Lời nói đầu

Tài liệu này được thực hiện sau một thời gian làm việc về Tính toán lưới của nhóm nghiên cứu Grid Computing tại trung tâm Tính toán hiệu năng cao - trường Đại học Bách Khoa Hà Nội.

Trong tài liệu này, chúng tôi xin trình bày về những hiểu biết về Tính toán lưới thu nhận được sau quá trình nghiên cứu, trình bày đặc tả chi tiết về hệ thống BKGrid2005 và các thành phần trong nó. Nội dung của tài liệu được chia làm 9 chương:

Chương 1: Tổng quan về tính toán lưới. Chương này giới thiệu các khái niệm cơ bản nhất về tính toán lưới, giúp người đọc có một cái nhìn toàn cảnh về tính toán lưới, về những khả năng mà tính toán lưới hứa hẹn sẽ mang lại cũng như các thành phần của lưới.

Chương 2: Giới thiệu chung về hệ thống BKGrid2005. Chương này nêu lên cái nhìn tổng quan về hệ thống BKGrid2005 đã được triển khai, từ các thành phần trong hệ thống cho đến kịch bản sử dụng hệ thống.

Chương 3, 4, 5, 6, 7, 8, 9: Chi tiết các thành phần trong hệ thống BKGrid2005. Đây là chương mô tả chi tiết từ việc đặt vấn đề, lựa chọn hướng tiếp cận cho đến việc thiết kế, xây dựng từng môđun cụ thể trong hệ thống.

MỤC LỤC

Danh mục hình vẽ

Chương 1: Tổng quan về tính toán lưới

1.1. Khái niệm chung

1.1.1. Giới thiệu

Ngày nay, với sự phát triển vượt bậc của khoa học kỹ thuật và công nghệ, đã xuất hiện những bài toán trong nhiều lĩnh vực đòi hỏi sức mạnh tính toán mà một máy tính riêng lẻ không thể đảm trách. Xuất phát từ những nhu cầu đó, các kỹ thuật tính toán song song, tính toán phân tán đã được đề xuất và đã phần nào đáp ứng được các yêu cầu này. Tuy nhiên, tham vọng của con người không dừng lại ở đó. Họ muốn một sức mạnh tính toán lớn hơn, với khả năng chia sẻ tài nguyên giữa mọi người trên phạm vi toàn cầu, khả năng tận dụng các phần mềm cũng như tài nguyên vật lý phân tán cả về mặt địa lý. Các tổ chức giải quyết vấn đề này bằng hai cách:

- Đầu tư thêm trang thiết bị, cơ sở hạ tầng tính toán (mua thêm máy chủ, máy trạm, siêu máy tính, cluster...). Tuy nhiên cách làm này hết sức tốn kém.
- Có một cách làm khác hiệu quả hơn đó là phân bố lại hợp lý các nguồn tài nguyên trong tổ chức hoặc thuê thêm các nguồn tài nguyên từ bên ngoài (tất nhiên là với chi phí rẻ hơn nhiều so với việc đầu tư cho cơ sở hạ tầng tính toán). Thực tế cho thấy có một phần lớn các nguồn tài nguyên của chúng ta đang được sử dụng lãng phí: các máy để bàn công sở thường chỉ hoạt động khoảng 5% công suất, ngay cả các máy chủ cũng có thể chỉ phải hoạt động với 20% công suất. Việc tận dụng hiệu quả các nguồn tài nguyên này có thể mang lại một sức mạnh tính toán khổng lồ

Cách giải quyết thứ hai này chính là mục tiêu của tính toán lưới. Tính toán lưới hướng đến việc chia sẻ và sử dụng hiệu quả các nguồn tài nguyên thuộc về nhiều tổ chức trên một quy mô rộng lớn (thậm chí là quy mô toàn cầu). Chính các công nghệ mạng và truyền thông phát triển mạnh mẽ trong những năm qua đã biến những khả năng này dần trở thành hiện thực. Các nghiên cứu về tính toán lưới đã và đang được tiến hành là nhằm tạo ra một cơ sở hạ tầng lưới cho phép dễ dàng chia sẻ và quản lý các tài nguyên đa dạng và phân tán trong môi trường lưới.

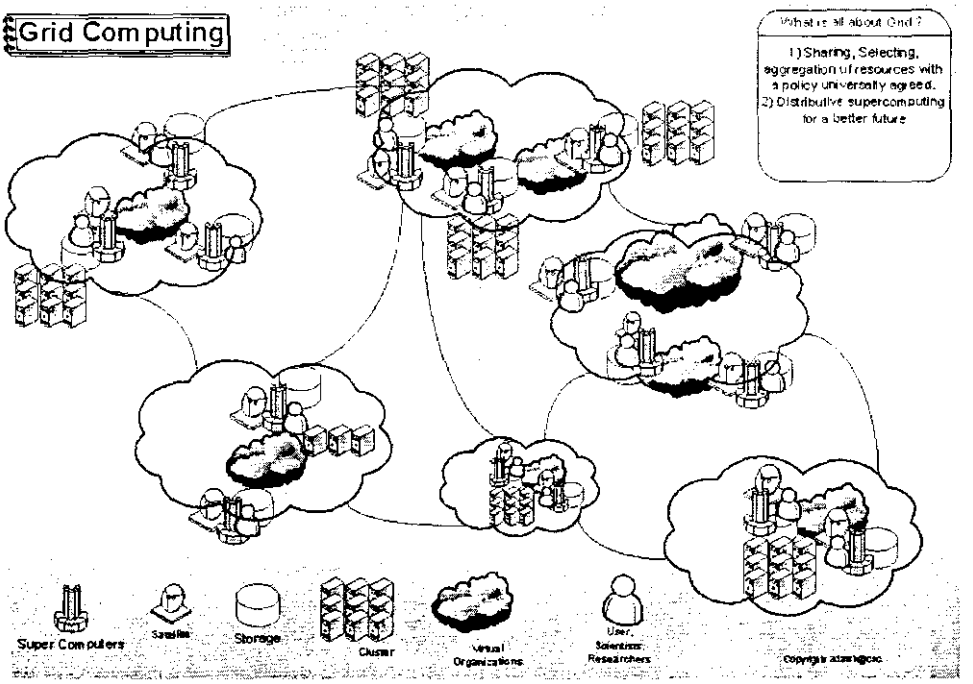
Như vậy, tính toán lưới, hiểu một cách đơn giản là sự phát triển tiếp theo của tính toán phân tán. Mục đích là tạo ra một máy tính ảo lớn mạnh từ một tập lớn các hệ thống không đồng nhất nhằm nâng cao khả năng tính toán, chia sẻ các tài nguyên khác nhau.

Các nhà khoa học tại Argone National Labs thuộc đại học Chicago (Mỹ) là những người đầu tiên đề xuất ý tưởng về tính toán lưới. Thuật ngữ lưới ở đây xuất phát từ lưới điện (electricity grid), ngụ ý rằng bất cứ một thiết bị tương thích nào đều có thể gắn vào trong lưới và được xếp ở một mức tài nguyên nào đó mà không cần quan tâm đến nguồn gốc của tài nguyên

đó. Trong tương lai, tính toán lưới có thể cung cấp cho người sử dụng các dịch vụ đóng vai trò như là dịch vụ cơ sở hạ tầng mà chúng ta sử dụng hàng ngày như điện, nước, giao thông...

Các công nghệ mạng và truyền thông phát triển mạnh mẽ trong những năm qua đã biến những khả năng này dần trở thành hiện thực. Các nghiên cứu về tính toán lưới đã và đang được tiến hành là nhằm tạo ra một cơ sở hạ tầng lưới cho phép dễ dàng chia sẻ và quản lý các tài nguyên đa dạng và phân tán trong môi trường lưới. Hai nhóm gồm Globus Alliance (được sự tài trợ của một vài trường đại học tại Mỹ như đại học Chicago, đại học Berkeley...) và Global Grid Forum (các thành viên bao gồm các hãng lớn như IBM, Sun, MicroSoft...) là các trung tâm nghiên cứu đáng chú ý hiện nay. Các nhóm này đã tạo ra các chuẩn mã nguồn mở và các giải pháp phần mềm cho công nghệ mới mẻ này. Đó là một nền tảng (framework) để các thành phần trong lưới có thể giao tiếp được với nhau. Trong đó:

- Globus Alliance tạo ra bộ công cụ Globus Toolkit (GT) mã nguồn mở, bao gồm các thư viện phần mềm và các dịch vụ cho phép người phát triển tạo ra các ứng dụng lưới. Thư viện của GT cung cấp các hàm đảm bảo vấn đề như an ninh, cơ sở hạ tầng thông tin, quản lí tài nguyên lưới, tính tin cậy, tính khả chuyển...
- Global Grid Forum quản lí các tiến trình chuẩn cho việc đặc tả kiến trúc các dịch vụ lưới OGSA (Open Grid Services Architecture) và OGSI (Open Grid Services Infrastructure). Các chuẩn OGSA, OGSI và bộ cung cụ Globus Toolkit giúp cho các nhà phát triển triển khai một cách thuận tiện các giải pháp tính toán lưới trong nhiều lĩnh vực nghiên cứu chuyên sâu ở Mỹ và châu Âu như: dự án tìm kiếm các tín hiệu ngoài trái đất SETI (Search for Extraerrestrial Intelligence), dự án về nghiên cứu bản đồ gen người, dự án IPG (Information Power Grid) của NASA... Đó là những ứng dụng tiêu biểu tạo sự thành công ban đầu của tính toán lưới trong giai đoạn nghiên cứu.



Hình 1-1: Minh họa về tính toán lưới

Hình 1-1 là một ví dụ về lưới, như một mạng liên kết các tài nguyên phân tán về mặt địa lý, các tài nguyên rất phong phú, đa dạng, bao gồm tập các siêu máy tính, các thiết bị truyền thông vệ tinh, các kho lưu trữ, các cluster tính toán hiệu năng cao , các tổ chức ảo liên kết trong lưới. Người dùng trong lưới cũng hết sức đa dạng, từ các người dùng thông thường, cho tới các người dùng chuyên dụng, có kiến thức sâu về chuyên môn như các nhà nghiên cứu, các nhà khoa học... Và lưới chính là sự tập hợp, chia sẻ, chọn lựa các nguồn tài nguyên này thông qua một chính sách thống nhất, phân phối các siêu máy tính và các hệ cluster để đạt hiệu năng tốt hơn trong tương lai.

Các thách thức mà công nghệ tính toán lưới đang phải giải quyết bao gồm:

- *Các tài nguyên hết sức đa dạng, không đồng nhất.* Tài nguyên ở đây được hiểu theo nghĩa hết sức tổng quát. Đó có thể là các tài nguyên phần cứng: tài nguyên tính toán, tài nguyên lưu trữ, các thiết bị đặc biệt khác...; các tài nguyên phần mềm: các CSDL, các phần mềm đặc biệt và đắt giá...; các đường truyền mạng...

Các tài nguyên này có thể rất khác nhau về mặt kiến trúc, giao diện, khả năng xử lý... Việc tạo ra một giao diện thống nhất cho phép khai thác và sử dụng hiệu quả các nguồn tài nguyên này là hoàn toàn không dễ dàng.

Ban đầu tính toán lưới được đặt ra chủ yếu là để tận dụng các nguồn tài nguyên tính toán nhưng hiện nay mục tiêu của nó đã được mở rộng sang rất nhiều nguồn tài nguyên khác như đã kể trên.

- *Các tài nguyên không chỉ thuộc về một tổ chức mà thuộc về rất nhiều tổ chức tham gia lưới.* Các tổ chức phải tuân thủ một số quy định chung khi tham gia vào lưới còn nhìn chung

là hoạt động độc lập tức là các tài nguyên này đều có quyền tự trị. Các tổ chức khác nhau thường có chính sách sử dụng hay cho thuê tài nguyên của họ khác nhau và do vậy cũng gây khó khăn cho việc quản lý.

- Các tài nguyên phân tán rộng khắp về mặt địa lý do vậy phải có các cơ chế quản lý phân tán.

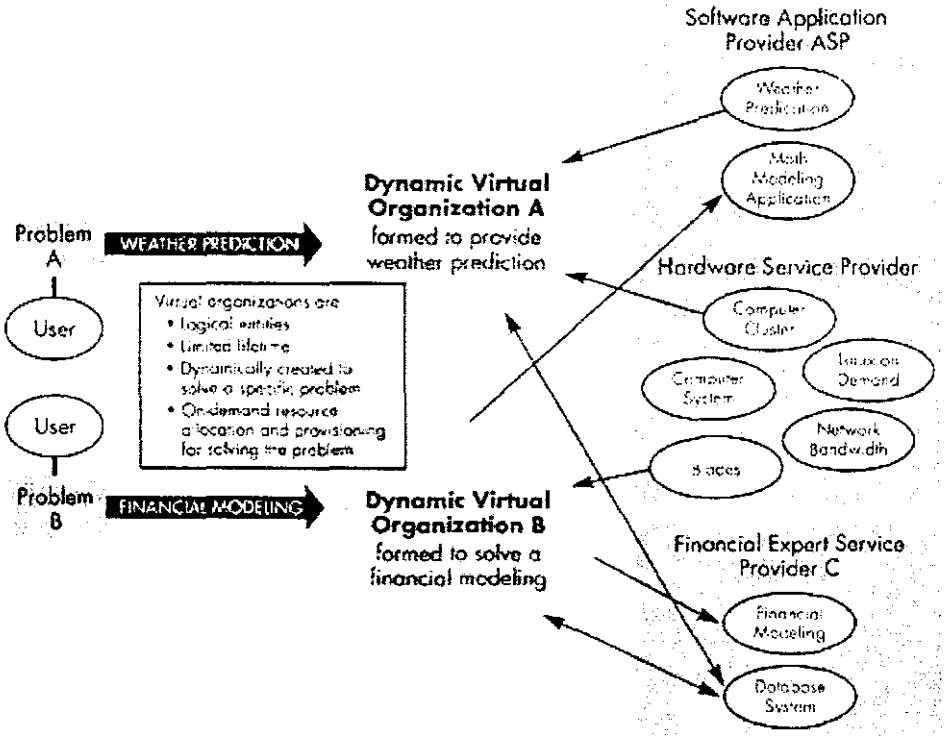
- *Đảm bảo an toàn thông tin* cho một môi trường phức tạp như môi trường lưới là rất khó khăn trong khi đây là một trong những điểm ưu tiên hàng đầu.

Theo Ian Foster, một hệ thống lưới là hệ thống có 3 đặc điểm chính sau:

- Phối hợp các tài nguyên phân tán từ nhiều miền quản trị khác nhau.
- Sử dụng các chuẩn mở và các giao thức mở.
- Mang lại cho người dùng chất lượng dịch vụ không tầm thường.

Riêng điểm thứ 2 là một điểm rất đáng lưu ý. Vì lưới là một môi trường thu hút nhiều tổ chức tham gia nên không thể coi nhẹ vai trò của các chuẩn mở và các giao thức mở, cũng giống như việc sử dụng các chuẩn này đã giúp cho mạng Internet bùng nổ mạnh mẽ trong những năm 90 của thế kỉ trước.

Khái niệm tổ chức ảo cũng là một khái niệm rất quan trọng trong tính toán lưới. Tổ chức ảo là một tổ chức được lập ra động để giải quyết một vấn đề nào đó. Thành phần của tổ chức ảo bao gồm rất nhiều tài nguyên thuộc về nhiều tổ chức (thực) khác nhau trong môi trường lưới và cùng hoạt động vì một mục tiêu chung. Tùy theo mức độ của vấn đề cần giải quyết mà các tổ chức ảo có thể rất khác nhau về quy mô, phạm vi hoạt động, thời gian sống. Hình dưới đây là một minh họa về tổ chức ảo. Có một người dùng cần giải một bài toán lớn về dự báo thời tiết, anh ta thành lập I tổ chức ảo bằng cách thuê một số nguồn tài nguyên khác nhau từ một vài tổ chức khác nhau. Tương tự như vậy, một người dùng cần giải một bài toán về dự báo tài chính, anh ta cũng thành lập một tổ chức ảo để giải quyết bài toán này.



Hình 1-2: Tổ chức ảo

Để hiểu rõ hơn về công nghệ tính toán lưới, trong phần tiếp theo chúng ta cùng điểm lại một số công nghệ tính toán đã và đang tồn tại và so sánh chúng với công nghệ tính toán lưới.

1.1.2. So sánh tính toán lưới với một số mô hình tính toán khác

1.1.2.1. World Wide Web (Web computing).

WWW hiện nay đang phát triển mạnh mẽ và được sử dụng rộng khắp. Sử dụng các chuẩn mở và các giao thức mở (TCP, HTTP, XML, SOAP), WWW có thể được sử dụng để xây dựng các tổ chức ảo tuy nhiên nó thiếu một số đặc tính quan trọng như các cơ chế chứng thực một lần, ủy nhiệm, các cơ chế phối hợp sự kiện...

1.1.2.2. Các hệ thống tính toán phân tán (Distributed computing systems).

Các công nghệ tính toán phân tán hiện tại bao gồm CORBA, J2EE, và DCOM rất thích hợp cho các ứng dụng phân tán tuy nhiên chúng không cung cấp một nền tảng phù hợp cho việc chia sẻ tài nguyên giữa các thành viên của tổ chức ảo. Một số khó khăn có thể kể ra trong việc khai phá tài nguyên, đảm bảo an ninh và xây dựng động các tổ chức ảo. Thêm nữa việc tương tác giữa các công nghệ này cũng gặp phải khó khăn. Tuy nhiên cũng đã có một số nghiên cứu nhằm mở rộng những công nghệ này cho môi trường lưới. VD Java JINI.

1.1.2.3. Các nhà cung cấp dịch vụ ứng dụng và dịch vụ lưu trữ (Application and Storage service provider).

Các nhà cung cấp ứng dụng và dịch vụ lưu trữ thường cung cấp cho người dùng một số ứng dụng cụ thể nào đó cũng như không gian lưu trữ. Người dùng tương tác với nhà cung cấp

dịch vụ thường là thông qua mạng riêng ảo (VPN) hoặc các đường truyền dành riêng và vì vậy nên loại bỏ được rất nhiều nguy cơ về an toàn bảo mật. Do vậy khi nói về các loại dịch vụ này thì ngữ cảnh của chúng cũng hẹp hơn tính toán lưới rất nhiều.

1.1.2.4. Các hệ thống tính toán ngang hàng (Peer – to – peer Computing systems).

Tính toán ngang hàng cũng là một lĩnh vực của tính toán phân tán. Một số hệ thống tính toán ngang hàng phổ biến hiện nay là SETI@home hay các mạng ngang hàng chia sẻ tập tin như Napster, Kazaa, Morpheus và Gnutella. Những điểm khác biệt chính giữa tính toán ngang hàng và tính toán phân tán là:

- Cộng đồng người sử dụng mà chúng hướng tới. Tính toán lưới có cộng đồng người sử dụng có thể nhỏ hơn tuy nhiên tập trung nhiều vào các ứng dụng và có yêu cầu cao hơn về an ninh cũng như tính toàn vẹn của ứng dụng. Trong khi đó các hệ thống mạng ngang hàng có thể có số người sử dụng rất lớn bao gồm cả các người dùng đơn lẻ và các tổ chức tuy nhiên không đòi hỏi cao về an ninh và mô hình chia sẻ tài nguyên cũng đơn giản hơn.
- Môi trường lưới liên kết các nguồn tài nguyên mạnh hơn, đa dạng hơn và chặt chẽ hơn.

1.1.2.5. Công nghệ tính toán hiệu năng cao truyền thống (High performance computing).

Như đã nói ở trên, để giải quyết những bài toán lớn người ta có thể đầu tư cho cơ sở hạ tầng tính toán. Để giải quyết những bài toán rất lớn và phức tạp người ta phải nghiên cứu xây dựng những hệ thống siêu tính toán. Các hướng nghiên cứu trong tính toán hiệu năng cao chủ yếu bao gồm:

- Nghiên cứu chế tạo những siêu máy tính tuần tự đơn bộ vi xử lý với tốc độ rất cao. Cách làm này gặp phải các giới hạn về vật lý như độ truyền dẫn của bán dẫn, tốc độ điện từ, nhiễu điện từ... nên không thể tăng mãi được.
- Để khắc phục khó khăn trên người ta nghiên cứu chế tạo các siêu máy tính song song bao gồm rất nhiều bộ xử lý hoạt động song song trên một bảng mạch chủ. Cách làm này đòi hỏi phải có những phần mềm thích hợp để tận dụng năng lực tính toán của hệ thống ví dụ: hệ điều hành song song phân tán, trình biên dịch song song, ngôn ngữ lập trình song song...
- Tuy nhiên việc nghiên cứu chế tạo ra các siêu máy tính nói chung mới chỉ được thực hiện ở các nước phát triển và giá thành của một hệ thống siêu máy tính như vậy (bao gồm cả phần cứng lẫn phần mềm hệ thống, công cụ phát triển) có thể lên đến hàng triệu đôla. Do vậy đối với các nước nghèo, các nước đang phát triển có thể dùng cách thứ ba, đó là xây dựng các siêu máy tính bằng cách kết nối nhiều máy tính thông thường với nhau thành một cluster. Cách làm này cho phép tạo ra các siêu máy tính giá rẻ. Bằng cách này viện CDAC của Ấn Độ đã tạo ra được siêu máy tính xếp thứ 173 trong số top 500 siêu máy tính của thế giới. Hiện nay trung tâm HPC của trường Đại học Bách Khoa Hà Nội cũng đã nghiên cứu thành công một siêu máy tính dạng cluster với 32 nút tính toán.

Như vậy khi nói đến các cluster ta thấy ngữ cảnh của chúng cũng hẹp hơn tính toán lưới rất nhiều thể hiện ở các điểm sau:

- Trước hết các cluster thuộc về một tổ chức, các tài nguyên của cluster là các máy tính thường là đồng nhất về mặt kiến trúc, hệ điều hành, khả năng xử lý... phân bố trên một giới hạn địa lý rất hẹp (ví dụ trong một phòng máy) và được liên kết với nhau bởi các mạng LAN tốc độ cao (cỡ vài trăm Mbits – Gbits) do vậy có độ tin cậy rất cao. Trái lại lưới bao gồm tập hợp rất nhiều tài nguyên đa dạng không đồng nhất thuộc nhiều tổ chức phân tán về mặt địa lý được liên kết với nhau bởi các mạng diện rộng có tốc độ thấp hơn và mức độ tin cậy kém hơn.
- Trong cluster các vấn đề về an ninh không được coi trọng. Vấn đề chủ yếu là làm sao để có được hiệu năng cao và không cần thiết phải hy sinh một phần hiệu năng vì các thủ tục an ninh (vì cluster chỉ thuộc về một tổ chức và chỉ nằm trong một phòng máy). Trong khi đó vấn đề an ninh là một trong những vấn đề quan trọng nhất mà tính toán lưới phải giải quyết.
- Trong cluster người ta có thể làm mọi thứ để đổi lấy hiệu năng. vd tối ưu hóa phần cứng phần mềm, sửa đổi nhân hệ điều hành hoặc viết hẳn những hệ điều hành chuyên biệt, thay đổi giao thức... Tính toán lưới không đặt vấn đề hiệu năng lên hàng đầu mà chủ yếu tập trung vào việc làm sao phân bổ hiệu quả các nguồn tài nguyên. Và tất nhiên khi đã đạt được mục tiêu đó thì tính toán lưới cũng mang lại hiệu năng rất cao, thậm chí hơn nhiều lần các siêu máy tính hiện có và điều đó chủ yếu là do quy mô cực kỳ rộng lớn của nó.

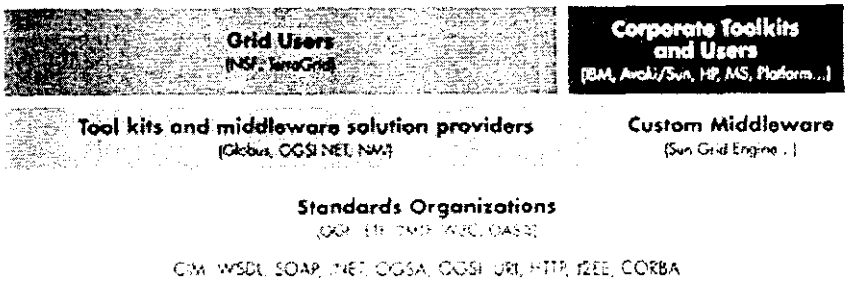
Cuối cùng cũng cần phải nói rằng tính toán lưới không phải là chìa khóa vạn năng dùng để giải quyết mọi vấn đề. Nó được dùng để bổ trợ chứ không phải là thay thế hoàn toàn các công nghệ tính toán hiện tại. Các công nghệ tính toán đã tồn tại cũng đã giải quyết từng phần các yêu cầu của tính toán lưới đặt ra (các yêu cầu về chia sẻ tài nguyên phân tán) tuy nhiên tính toán lưới hiện nay chính thức giải quyết các vấn đề đó một cách tập trung và bài bản hơn với một ngữ cảnh rộng hơn rất nhiều.

1.1.3. Các tổ chức tham gia vào quá trình phát triển của tính toán lưới

Các tổ chức tham gia vào quá trình phát triển của tính toán lưới có thể chia ra làm bốn nhóm lớn sau:

1.1.3.1. Các tổ chức phát triển các chuẩn cho lưới.

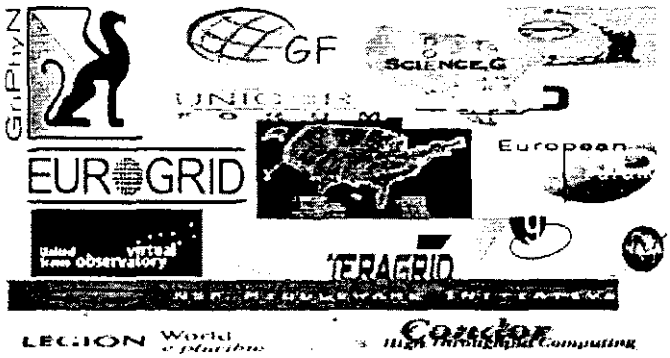
Đại diện cho nhóm này là diễn đàn lưới toàn cầu (GGF – Global Grid Forum) và các tổ chức chuẩn hóa quốc tế khác như OASIS (Organization for the Advancement of Structured Information Standards), W3C (World Wide Web Consortium), IETF (the Internet Engineering Task Force) và DMTF (the Distributed Management Task Force). Hiện nay một trong những hoạt động chính của GGF là phát triển chuẩn dịch vụ lưới OGSA.



Hình 1-3: Phân loại các tổ chức

1.1.3.2. Các tổ chức phát triển các bộ công cụ, framework và các middleware.

Bao gồm nhiều trường đại học, các viện nghiên cứu. Các tổ chức này đã cho ra đời rất nhiều các bộ công cụ phát triển lưới như Legion, Condor, Nimrod, Unicore, Globus...



Hình 1-4: Một số tổ chức tiêu biểu

1.1.3.3. Các tổ chức xây dựng và sử dụng các giải pháp lưới.

Có thể kể ra một số lưới tiêu biểu trên thế giới như Nasa Information Power Grid của Nasa, Science Grid của bộ quốc phòng Mỹ, dự án EuroGrid của liên minh châu Âu với nhiều lưới con như Bio Grid, Metro Grid, Computer-Aided Engineering (CAE) Grid, High Performance Center (HPC) Research Grid, Terra Grid của NSF...

1.1.3.4. Các tổ chức đưa công nghệ lưới vào các sản phẩm thương mại.

Trong nhóm này có nhiều đại gia trong ngành công nghiệp máy tính như IBM, SUN, HP,... Các hãng này đưa ra nhiều giải pháp khác nhau dựa trên nền công nghệ tính toán lưới. Hiện nay trên thế giới đã có sự phân biệt giữa công nghệ tính toán lưới mang tính hàn lâm và công nghệ tính toán lưới trong doanh nghiệp.

1.1.4. Một số bộ công cụ phát triển lưới chính hiện nay

Hiện nay trên thế giới có nhiều bộ công cụ phát triển hỗ trợ việc xây dựng lưới ở nhiều mức độ khác nhau. Tiêu biểu là:

1.1.4.1. Globus Toolkit

Globus là một dự án nghiên cứu gồm nhiều tổ chức tham gia với mục tiêu ban đầu là tạo cơ sở hạ tầng và các dịch vụ cấp cao cho một lưới tính toán tuy nhiên hiện nay nó đã mở rộng

phạm vi thành cơ sở hạ tầng cho phép chia sẻ nhiều loại tài nguyên đa dạng. Globus Toolkit là bộ phần mềm nguồn mở bao gồm các dịch vụ và các thư viện hỗ trợ cho việc quản lý, khai phá tài nguyên. Ngoài ra, bộ phần mềm này còn hỗ trợ hệ thống quản lý tệp và dịch vụ bảo mật. Cho đến nay, đây là bộ công cụ mạnh nhất và được sử dụng rộng rãi nhất để thiết kế và cài đặt các ứng dụng lưới. Điểm ưu việt là Globus Toolkit hỗ trợ hoàn toàn các giao thức OGSA và OGSI. Đồng thời, mô hình lập trình mà Globus Toolkit đưa ra là mô hình hướng dịch vụ. Globus Toolkit đã phát triển sang đến phiên bản 4.0, tuy nhiên, hiện nay phiên bản 3.2 là ổn định nhất và đang được ứng dụng rộng rãi hơn cả. Trong hệ thống tính toán lưới mà Trung tâm tính toán hiệu năng cao đại học Bách Khoa Hà Nội triển khai là phiên bản 3.2.

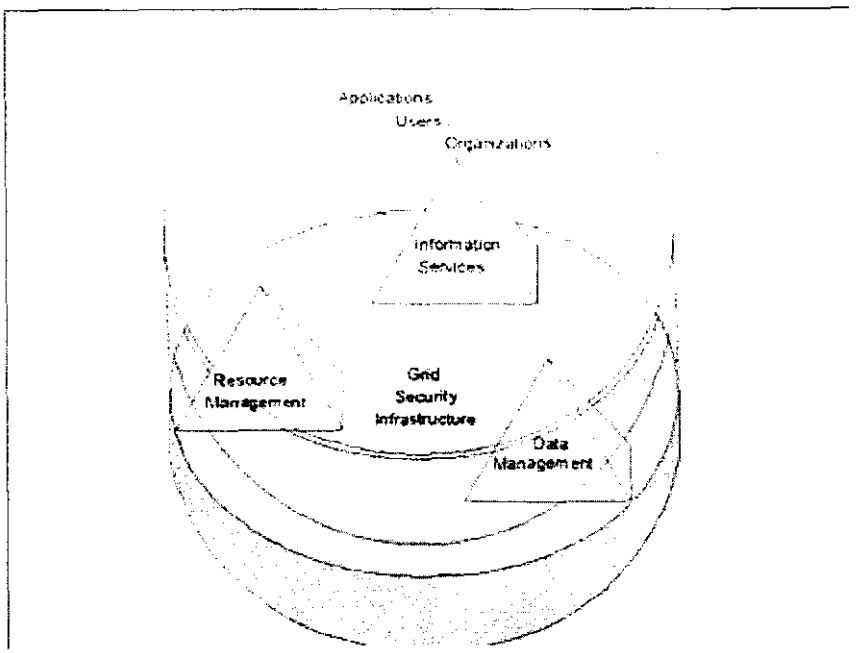
- Kiến trúc Globus Toolkit 3.2

GT3 hỗ trợ các tác vụ sau: chứng thực một lần, chứng thực và chứng quyền, đệ trình công việc, quản lý, tìm kiếm và cấp phát tài nguyên và truyền dữ liệu. Ngoài ra, GT3 còn cung cấp các API (Application Programming Interface) và bộ công cụ phát triển hệ thống (SDK - System Development Kit). Ba chức năng chính của bộ công cụ GT3 được ví dụ ba kim tự tháp, được xây dựng trên cơ sở hạ tầng bảo mật (GSI - Grid Security Infrastructure).

Quản lý tài nguyên: thực hiện các công việc sau bao gồm cấp phát tài nguyên, đệ trình công việc để thực hiện công việc từ xa và nhận kết quả trả về. Cuối cùng là quản lý trạng thái công việc và tiến trình. Các dịch vụ thông tin: cung cấp các hỗ trợ cho việc thu nhập thông tin trong lưới và truy vấn các thông tin này, dựa trên giao thức LDAP (Lightweight Directory Access Protocol).

Quản lý dữ liệu: hỗ trợ việc truyền tệp giữa các máy trong lưới và quản lý việc truyền tệp.

Dịch vụ thông tin: cung cấp đầy đủ thông tin liên quan đến các tài nguyên lưới (số lượng nút tài nguyên, tình trạng tài nguyên...) cũng như các thông tin về các dịch vụ mà lưới cung cấp.



Hình 1-5: Ba kim tự tháp trong Globus Toolkit 3.2

Mặc dù ba kim tự tháp nói trên chưa phải đã đáp ứng đầy đủ tất cả các yêu cầu để có thể xây dựng một dịch vụ lưới. Nhưng đó là những thành phần cơ bản, từ đó hỗ trợ cho người thiết kế ứng dụng có thể triển khai ứng dụng trên lưới.

- Các dịch vụ hỗ trợ bởi Globus Toolkit 3.2

Tầng kết nối

GT3.2 hỗ trợ dịch vụ bảo mật GSI (Grid Security Infrastructure). Đây là một trong những dịch vụ cơ bản đảm bảo sự tồn tại của lưới. GSI hỗ trợ các yêu cầu khắt khe về bảo mật trong lưới như cơ chế đăng nhập một lần (single sign on), cơ chế uỷ quyền (delegation), chứng thực đa phương (mutual authorization)...

Ngoài ra, GT3.2 còn tích hợp một dịch vụ truyền thông bảo mật là GridFTP. Giao thức truyền thông GridFTP là sự mở rộng của giao thức FTP và được GT3.2 hỗ trợ dưới dạng dịch vụ. Các dịch vụ ở tầng trên có thể sử dụng giao thức này để thực hiện các tác vụ truyền dữ liệu an toàn, độ tin cậy cao.

Tầng tài nguyên

GRIP (Grid Resource Information Protocol) dùng để định nghĩa giao thức về các thông tin tài nguyên chuẩn và các dạng thông tin liên quan. Ngoài ra còn có giao thức đăng kí tài nguyên trạng thái mềm GRRP (Grid Resource Registration Protocol) để đăng kí tài nguyên với các GIIS (Grid Index Information Servers).

Giao thức GRAM (Grid Resource Access and Management) dùng để cấp phát các tài nguyên tính toán và quản lí việc thực thi trên các tài nguyên.

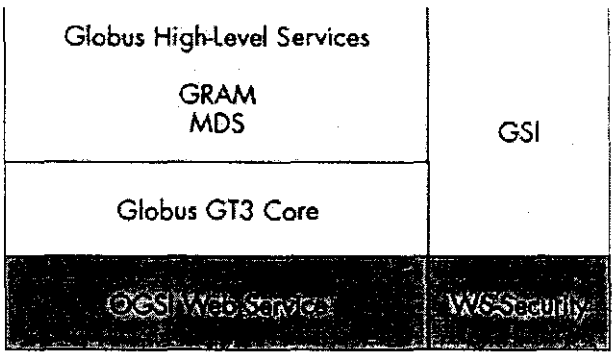
Giao thức LDAP (Lightweight Directory Access Protocol) để truy cập thông tin thư mục. Ngoài ra, Globus Toolkit còn định nghĩa các API phía khách (client) của C, Java cho các giao

thức trên. Các API phía chủ (server) cũng được cung cấp để tạo thuận tiện cho việc tích hợp các nguồn tài nguyên khác nhau vào trong lưới. Ví dụ như GRIS (Grid Resource Information Service) thực thi các hàm LDAP phía chủ để hiển thị các thông tin tài nguyên chuyên dụng theo yêu cầu.

GSS (Generic Security Services) là dịch vụ để xác định, chuyển giao và phân tích các giấy uỷ quyền cho phép thay thế các dịch vụ bảo mật ở tầng kết nối.

Tầng kết hợp

Ngoài các dịch vụ trên Globus Toolkit hỗ trợ dịch vụ GIIS chuyên hiển thị các thông tin về tài nguyên xác định thông qua giao thức thông tin LDAP, GRIS để lấy trạng thái tài nguyên và GRRP để đăng kí tài nguyên. Bên cạnh đó còn có dịch vụ nhân bản dữ liệu và quản lí việc nhân bản dữ liệu trong môi trường lưới (Replication). Các thư việc DUROC (Updated Request Online Coallocator) cung cấp nhiều API đảm bảo cho việc động cấp phát các tài nguyên.



Hình 1-6: Các dịch vụ cơ bản của GT

Globus được thiết kế theo kiến trúc phân tầng với tầng dưới cùng chính là các dịch vụ Web. Tầng tiếp theo là các dịch vụ lõi và tầng trên cùng là các dịch vụ cấp cao cung cấp các chức năng sử dụng cho người dùng. Có thể thấy xuyên suốt kiến trúc của GT là các cơ chế bảo mật cũng dựa trên nền tảng bảo mật của dịch vụ Web.

1.1.4.2. Legion

Legion là một dự án phát triển middleware cho lưới do đại học Virginia phát triển. Bộ công cụ Legion tookit được phát hành lần đầu tiên vào năm 1997. Đây là một bộ công cụ phát triển lưới hướng đối tượng. Năm 1998 công ty Avaki được thành lập đưa bộ công cụ này vào sử dụng cho mục đích thương mại.

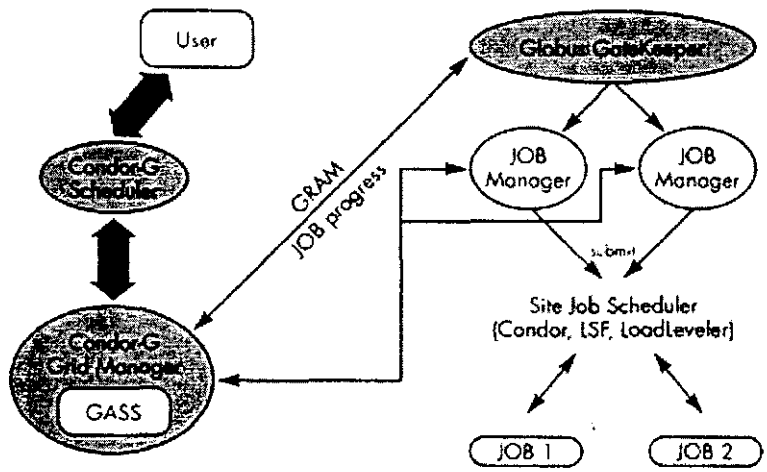
1.1.4.3. Condor

Condor là một công cụ hỗ trợ khả năng lưu trữ của các máy trạm phục vụ cho tác vụ tính toán. Condor phù hợp với các bài toán bao gồm các công việc không cần liên tác với nhau. Có thể xem Condor như là hệ thống quản lý tải chuyên dụng cho các công việc tính toán. Giống như các hệ thống xử lý theo lô, Condor cung cấp cơ chế xếp hàng công việc, chính sách lập lịch, biểu đồ ưu tiên và quản lý tài nguyên. Phụ thuộc vào công việc do người sử

dụng đệ trình là tuần tự hay song song, Condor đặt chúng vào hàng đợi, sau đó lựa chọn công việc để thực thi dựa voà chính sách ưu tiên, tình trạng công việc... và quản lý toàn bộ quá trình thực thi công việc.

Condor được sử dụng để quản lý một cụm máy tính hay các nút tính toán được nhóm lại với nhau. Ngoài ra, Condor còn có cơ chế tìm kiếm để đáp ứng giữa yêu cầu tài nguyên (công việc) và khả năng đáp ứng (máy tính). Bên cạnh bộ công cụ Condor tập trung vào việc hỗ trợ khả năng tính toán của các tài nguyên có tính tương đồng và được nhóm lại với nhau còn có phiên bản Condor-G cung cấp dịch vụ quản lý tài nguyên cho các ứng dụng lưới. Phần mềm này là sự kết hợp giữa các giao thức quản lý tài nguyên của Globus (GRAM, dịch vụ chỉ mục) và phương thức quản lý tài nguyên mang tính nội bộ của Condor. Hình vẽ 1.6 mô tả dạng sử dụng đơn giản của Condor-G kết hợp với Globus.

Như trên hình 1-7, Condor-G chứa một máy chủ GASS Server, dùng để chuyển công việc ra hay vào trung tâm tính toán. Bộ quản lý Condor-G sử dụng GRAM để lấy các thông tin tiến trình công việc thực hiện.



Hình 1-7: Chạy công việc từ xa sử dụng Condor-G

Condor-G được sử dụng trong các tổ chức thương mại hay khoa học. Tiêu biểu như các dự án Grid Physics Network (GriPhyN), International Virtual Data Grid Loboratory (VDGL).

1.1.4.4. Nimrod

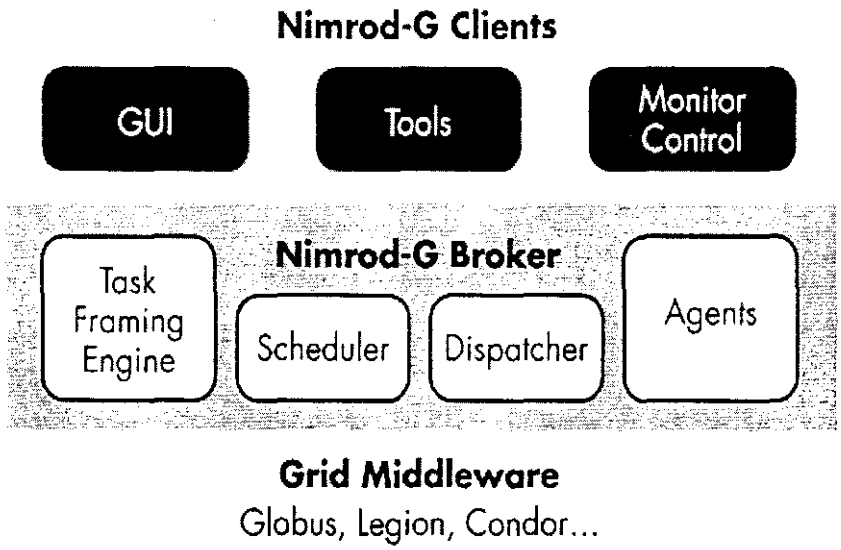
Nimrod cung cấp giao diện người dùng để mô tả các bài toán dạng "xử lý theo tham số" (parameter sweep) và trả lại kết quả của các công việc độc lập nhau được đệ trình bởi hệ thống quản lý tài nguyên.

Nimrod-G tương tự như Condor-G, tích hợp Nimrod và Globus làm cho các tài nguyên đa miền được coi như thuộc cùng một miền. Phần mềm này cung cấp ngôn ngữ mô tả để biểu diễn các tham số cần thiết để thực hiện công việc. Hệ thống bao gồm hệ quản lý tài nguyên và các thuật giải lập lịch công việc. Người sử dụng có thể quan sát quá trình thương lượng

gữa bộ môi giới tài nguyên và các dịch vụ trong thời gian chạy theo yêu cầu về chất lượng dịch vụ của họ. Kiến trúc Nimrod bao gồm:

Nimrod client, cho người sử dụng công cụ tạo ra các ứng dụng dạng "xử lý theo tham số", hướng dẫn và quản lý và tuỳ biến ứng dụng.

Bộ môi giới tài nguyên, có chứa bộ máy quản lý tác vụ TFE (Task Farming Engine), một bộ lập lịch làm nhiệm vụ khai phá tài nguyên, thương lượng và lập lịch, bộ khởi động và các tác tử (Agent) quản lý công việc trên các tài nguyên.



Hình 1-8: Kiến trúc Nimrod-G

Như vậy, Nimrod-G làm đơn giản quá trình thực hiện công việc trên lưới thông qua việc quản lý các yêu cầu của người sử dụng và các đáp ứng của tài nguyên trong lưới bằng các dịch vụ tài nguyên khác nhau. Nimrod hỗ trợ các giải thuật lập lịch quan trọng như:

+ Tối ưu hoá theo chi phí: sử dụng tài nguyên có giá rẻ nhất.

+ Tối ưu hoá theo thời gian: chạy song song các công việc.

+ Tối ưu hoá theo chi phí – thời gian: tương tự như giải thuật tối ưu hoá theo thời gian nhưng nếu có nhiều công việc có cùng chi phí, công việc nào có thời gian thực hiện ít hơn sẽ được ưu tiên.

Bước thời gian đặt trước: tương tự như giải thuật tối ưu hoá theo thời gian nhưng phải đảm bảo các công việc chưa được xử lý phải được đặt trước.

1.1.4.5. Unicore

Unicore là dự án nghiên cứu lưới được tài trợ bởi bộ giáo dục Đức. Hiện nay có một dự án mang tên GRIP (GRid Interoperability Project) đã được thực hiện từ 2002 nhằm mang lại khả năng tương tác giữa Unicore và Globus.

1.2. Những ứng dụng cơ bản

Trong mục này, chúng ta sẽ điểm kĩ hơn về những ứng dụng cơ bản của tính toán lưới để thấy được lợi ích mà công nghệ mới này hứa hẹn đối với người sử dụng:

1.2.1. Khai thác những nguồn tài nguyên chưa được sử dụng đúng mức

Ứng dụng cơ bản nhất của tính toán lưới chính là chạy một ứng dụng hiện hữu trên một máy khác. Công việc được yêu cầu có thể được chạy trên một máy rỗi đâu đó trong mạng lưới. Cần ít nhất hai điều kiện tiên quyết để thực hiện yêu cầu này. Thứ nhất, ứng dụng phải có khả năng thực hiện từ xa; thứ hai, máy từ xa phải đáp ứng được những yêu cầu về phần cứng, phần mềm, và/hoặc những yêu cầu về tài nguyên khác cho ứng dụng.

Theo thống kê, trong đa số các tổ chức, phần lớn các tài nguyên tính toán không được tận dụng đúng mức. Đa số các máy desktop chỉ bận rộn trong khoảng 5% thời gian, và thậm chí, trong một vài tổ chức, server cũng thường rất nhàn rỗi. Tính toán lưới cung cấp khả năng khai thác triệt để những tài nguyên chưa được tận dụng đúng mức và như vậy đã làm tăng đáng kể hiệu quả khai thác các tài nguyên này.

Trong thực tế, các máy có thể để trống một dung lượng đĩa khá lớn. Tính toán lưới có khả năng tập hợp các không gian nhớ trống này thành một một bộ nhớ ảo lớn hơn, tạo ra sự cải thiện hiệu năng và độ tin cậy cao hơn. Thực vậy: giả sử một ứng dụng cần đọc rất nhiều dữ liệu; dữ liệu này có thể được tự động nhân bản tại nhiều điểm chiến lược trên lưới. Như vậy, nếu ứng dụng phải xử lý trên một máy từ xa trong lưới, dữ liệu đã ở ngay đó hoặc gần đó và do vậy không cần phải di chuyển đến điểm xa xôi đó. Điều này tạo ra những lợi ích về hiệu năng, đồng thời, việc sao chép dữ liệu như vậy có thể được sử dụng như những bản sao lưu khi bản gốc bị hư hại hoặc không sẵn sàng.

Một chức năng khác của lưới là cân bằng việc sử dụng tài nguyên. Khi có một ứng dụng đòi hỏi quá nhiều tài nguyên thì rõ ràng một hệ thống đơn lẻ không đủ sức để đáp ứng nó. Giải pháp ở đây là chúng có thể được chuyển đến những hệ thống chưa được sử dụng đúng mức, nói cách khác là ứng dụng đó có thể được phân tán ra từng phần để hoàn thành trên nhiều hệ thống khác nhau. Tóm lại, lưới có thể đảm bảo tính nhất quán để cân bằng tải trên một tập rộng lớn các tài nguyên.

1.2.2. Khả năng thực hiện tính toán song song

Khả năng tính toán song song là một trong nững đặc điểm lôi cuốn nhất của tính toán lưới. Ngoài nhu cầu khoa học thuần túy, sức mạnh tính toán song song đang tạo nên một bước tiến bộ mới trong các lĩnh vực khác nhau của đời sống như: y sinh học, mô hình tài chính, thăm dò dầu mỏ, làm phim hoạt hình, và nhiều lĩnh vực khác.

Yêu cầu chung của việc thực hiện tính toán song song là các ứng dụng được thiết kế sao cho nó có thể được chia nhỏ thành các phần chạy độc lập với nhau. Một ứng dụng có yêu

cầu tính toán lớn trên lưới có thể được xem như nhiều "công việc con" nhỏ hơn mà mỗi công việc con này có thể được thực hiện trên một máy nào đó trong lưới. Mức độ liên quan giữa các công việc con càng thấp thì ứng dụng càng tối ưu. Một ứng dụng thực hiện song song sẽ kết thúc nhanh hơn gấp 10 lần nếu nó sửu dụng gấp 10 lần số bộ vi xử lý.

Việc song song hoá các công việc đặt ra rất nhiều vấn đề. Trước tiên, cần có giải thuật phù hợp để chia ứng dụng ra thành từng phần cho nhiều CPU. Vấn đề thứ hai là nếu các thành phần được chia ra không độc lập hoàn toàn có thể gây ra sự tương tranh về tài nguyên. Ví dụ: nếu tất cả các công việc con cần phải đọc ra và viết vào một file chung hoặc CSDL chung, những giới hạn truy nhập file hay CSDL đó sẽ trở thành nhân tố gây ra sự tương tranh. Sự tương tranh còn tiềm ẩn trong việc truyền thông điệp giữa các công việc con, khả năng truyền thông mạng, những giao thức đồng bộ hoá, băng thông vào ra cho các thiết bị lưu trữ.

Một đóng góp quan trọng khác của tính toán lưới là cho phép và đơn giản hoá sự tương tác giữa các lĩnh vực, hệ thống, lĩnh vực rộng hơn. Trước đây, tính toán phân tán hứa hẹn cho sự hợp tác này, và ở một chừng mực nào đó đã đạt được thành công. Tính toán lưới cung cấp những tiêu chuẩn quan trọng cho phép các hệ thống không đồng nhất có thể làm việc với nhau để hình thành một hệ thống tính toán ảo lớn với những nguồn tài nguyên ảo khác nhau. Người dùng của lưới có thể được tổ chức động vào trong một số tổ chức ảo, mỗi tổ chức có thể có những yêu cầu chính sách khác nhau. Những tổ chức ảo này có thể chia sẻ tài nguyên với nhau và tập hợp lại trong một lưới lớn hơn.

Một ưu điểm nổi bật của sự hợp tác là việc chia sẻ về dữ liệu. Những file hoặc cơ sở dữ liệu (CSDL) có thể mở rộng thông suốt qua nhiều hệ thống, như vậy nó sẽ có khả năng khai thác được nhiều hơn bất kỳ hệ thống đơn lẻ nào. Tuy vậy, việc chia sẻ tài nguyên không chỉ giới hạn về dữ liệu mà còn đặt ra đối với các nguồn tài nguyên khác như các thiết bị, phần mềm, các dịch vụ, bản quyền,... Các tài nguyên này được *ảo hoá* để tạo cho chúng khả năng vận hành đồng dạng hơn giữa những người tham gia vào lưới.

Những người tham gia vào lưới có thể là thành viên của một vài tổ chức thực hay ảo. Khi tham gia vào lưới, họ buộc phải tuân theo những chính sách bảo mật, những chính sách về giải quyết quyền ưu tiên,...

1.2.3. Chia sẻ những nguồn tài nguyên đặc biệt

Ngoài việc chia sẻ CPU và những tài nguyên lưu trữ, một lưới còn có thể cung cấp các dịch vụ cho phép truy nhập tới các thiết bị đặc biệt, phần mềm, bản quyền và những dịch vụ khác.

Ví dụ: một người dùng cần tăng thêm băng thông để thực hiện tìm kiếm, khai phá dữ liệu, công việc có thể được chia ra cho các máy trong lưới có kết nối độc lập tới Internet. Khi đó, toàn bộ khả năng tìm kiếm được nhân lên vì mỗi máy đều có một kết nối riêng biệt tới

Internet. Tất nhiên, nếu những máy này đã chia sẻ kết nối tới mạng thì băng thông tăng lên không đáng là bao.

Một vài máy có thể chia sẻ những phần mềm có bản quyền đắt đỏ. Để thực hiện điều này, công việc sẽ được gửi đến máy có chứa phần mềm đó, sau khi được thực hiện sẽ trả về kết quả cho máy yêu cầu. Nói cách khác, đây là việc chia sẻ giấy phép phần mềm.

Một vài máy trên lưới có thể có những thiết bị đặc biệt, và lưới sẽ đóng vai trò chia sẻ các thiết bị đó. Ví dụ: một vài máy trên lưới có thể kết nối với những kính hiển vi điện tử lớn. Để dùng nó từ xa, một mẫu thử được gửi tới chủ nhân của kính hiển vi. Sau đó, người dùng có thể điều khiển máy từ xa, thay đổi góc nhìn phối cảnh cho đến khi lấy được hình ảnh mong muốn.

1.3. Các thành phần của một hệ thống tính toán lưới

1.3.1. Thành phần quản lý

Bất kì một lưới nào cũng cần có một thành phần quản lý. Trước hết, thành phần này theo dõi các tài nguyên đang sẵn dùng và thành viên nào đang có mặt trong hệ thống. Các thông tin này rất quan trọng trong việc gán công việc cho máy nào trong lưới.

Tiếp đó là thành phần đo lường xác định dung lượng của từng nút mạng và tỉ lệ tài nguyên được sử dụng tại một thời điểm bất kì. Điều này là cơ sở cho việc lập lịch cho các tiến trình lưới. Nó cũng cho thấy tình trạng của lưới, cảnh báo tới người sử dụng về khả năng có thể xảy ra thất bại do thiếu tài nguyên, đụng độ hay các nguyên nhân khác. Một thủ tục khác cũng cần đến các thông tin này là các thống kê về sử dụng hệ thống, việc chi trả cho hệ thống khi chạy các phần mềm trên lưới.

1.3.2. Phần mềm donor

Mỗi máy tính thành viên đóng góp tài nguyên vào hệ thống đều cần có quá trình đăng nhập trước khi được xem như một thành viên chính thức. Thường thì sẽ có các thủ tục định danh và thẩm quyền để thực hiện, các thủ tục này sẽ giúp tạo lập một tài khoản cho máy thành viên cũng như người chủ của máy đó.

Một số hệ thống lưới tự động đăng nhập trong khi một số khác sử dụng ngay việc đăng nhập vào hệ điều hành trên máy địa phương. Trong kiểu thứ hai, hệ thống đối chiếu định danh người dùng sẽ quyết định quyền của người sử dụng đối với các máy khác nhau trong hệ thống lưới. Những quyền hạn này thường được quyết định bởi người quản trị hệ thống. Anh ta sẽ lập ra CSDL về người dùng và cả vào nơi được bảo vệ cẩn mật.

Trong một số hệ thống lưới, người ta không có một cơ chế định danh hay thẩm quyền nào, và người dùng nào cũng có thể đệ trình công việc để thực hiện trên lưới. Các hệ thống như vậy có ưu điểm là dễ cài đặt, các phần mềm không phức tạp. Tuy vậy, khi quy mô của hệ

thống được mở rộng thì điều này là cực kì nguy hiểm bởi nó sẽ dễ dàng bị hacker tấn công và hậu quả thật khó lường trước.

Hệ thống lưới có khả năng thông báo tới tất cả các máy thành viên rằng có một tài nguyên mới được kết nối vào hệ thống. Máy đệ trình phải tiến hành một số các hoạt động theo dõi, đo lường tỉ lệ tài nguyên được sử dụng trên một máy, máy nào ở trạng thái nghỉ,... Các thông tin này được truyền đến hệ thống quản lý lưới phục vụ cho công việc lập lịch sau này.

Vấn đề quan trọng nhất vẫn là khả năng tiếp nhận công việc và thực hiện công việc của một phần mềm được cài đặt trên một máy nào đó khi được gán nhiệm vụ. Quá trình được mô tả như sau: tại một trạm nào đó trên lưới, người sử dụng đệ trình một công việc và yêu cầu thực hiện trên lưới. Phần mềm quản lý phải có khả năng chọn ra máy thực hiện, liên lạc với máy đó để gửi công việc cần thực hiện. Phần mềm đệ trình phải có khả năng nhận được file chạy hoặc chọn bản copy trên máy sẵn, tiếp đó file chạy được thực hiện trên máy đệ trình và kết quả được trả về cho người yêu cầu. Hệ thống tiên tiến còn cho phép điều chỉnh động ưu tiên cho các công việc, dừng chúng lại khi cần thiết và sau đó có thể khởi động tiếp tục quá trình trên một máy khác. Các hoạt động này thường căn cứ vào tải hiện thời của hệ thống, các ưu tiên thay đổi trên lưới.

1.3.3. Phần mềm đệ trình

Thông thường người ta có thể dùng bất cứ máy thành phần nào trong lưới để tiến hành đệ trình một công việc nào đó. Tuy vậy, trong một số hệ thống, việc đệ trình các công việc được thực hiện bởi một số thành phần cài đặt trên một số máy gọi là nút đệ trình hay máy khách đệ trình. Khi lưới được xây dựng dựa trên các tài nguyên chuyên dụng hơn tài nguyên thường thì các thành phần đệ trình thường được cài đặt trên máy của người dùng hay các trạm làm việc.

1.3.4. Quản lý phân tán

Các lưới được cài đặt trên phạm vi rộng lớn thường có cấu trúc hình cây hay các dạng thủ hình khác phù hợp với dạng kết nối. Theo đó, các máy tính được kết nối thông qua mạng LAN tạo thành các cluster. Các lưới có thể được tạo nên từ các cluster của các cluster này. Vì thế đòi hỏi phải có các thủ tục quản lý các lưới phân tán sao cho đạt hiệu quả tính toán cao nhất. Các thao tác trên lưới cũng như các thủ tục chia sẻ dữ liệu và lập lịch phải phù hợp với cấu hình của lưới. Ví dụ: bộ lập lịch trung tâm sẽ không gán trực tiếp công việc cho một máy cụ thể mà gán cho một cluster ở dưới nó, sau đó bộ lập lịch của cluster này mới thực hiện gán công việc cho một máy cụ thể trong cluster đó. Tương tự như thế khi một máy đệ trình một công việc nào đó: công việc này sẽ được chuyển tới bộ quản lí của cluster chứa máy thành phần, sau đó được tiếp tục chuyển lên bộ quản lí cao hơn để từ đó thực hiện lập lịch cho công việc này.

1.3.5. Bộ lập lịch

Phần lớn các lưới tính toán đều có các phần mềm lập lịch, các phần mềm này có nhiệm vụ chọn ra các máy thành phần để thực thi các công việc được đệ trình tại một máy nào đó. Cơ chế lập lịch đơn giản nhất là cơ chế round-robin, tức là hệ thống sẽ chọn máy tiếp theo có các tài nguyên đáp ứng nhu cầu của công việc để thực thi. Tuy vậy trong các hệ thống tiên tiến thì các cơ chế lập lịch phức tạp và hiệu quả hơn sẽ được áp dụng.

Một số bộ lập lịch thực hiện gán ưu tiên cho từng công việc. Điều này được thực hiện bằng cách duy trì các hàng đợi công việc, mỗi hàng đợi chứa một tập các công việc với cùng mức ưu tiên. Khi một máy đã thực hiện xong công việc của mình, nó sẽ chọn tiếp một công việc ở hàng đợi có mức độ ưu tiên cao nhất. Cơ chế này sẽ được kết hợp với một số quy định khác dựa trên cơ sở là các tài nguyên người sử dụng .... Ví dụ như một tài nguyên trong tổ chức có thể bị hạn chế sử dụng trong khoảng thời gian nào đó trong ngày.

Bộ lập lịch phải dựa vào các thông tin như tải của lưới, các thông tin được đo lường như tỉ lệ tài nguyên được sử dụng để có thể quyết định xem máy thành phần có bận không trước khi đệ trình một công việc. Cũng như tổ chức của mạng, bộ lập lịch cũng có thể được tổ chức theo cấu trúc cây. Ví dụ: bộ siêu lập lịch (meta-scheduler) sẽ đệ trình một công việc tới một bộ lập lịch ở mức dưới chứ không phải là một máy thành phần cụ thể.

Các bộ lập lịch tiên tiến hơn còn có các chức năng theo dõi quá trình thực hiện của các công việc trong lưới, từ đó có thể quản lí được luồng công việc của toàn bộ hệ thống. Nếu một công việc bị ngừng do thiếu tài nguyên hệ thống, bộ lập lịch tốt phải có nhiệm vụ đệ trình lại công việc tại một nơi khác trong hệ thống. Tương tự như vậy: khi tiến trình rơi vào trạng thái lặp vô hạn và đạt tới thời gian giới hạn thì công việc đó sẽ không được thực hiện lại nữa. Thường thì mỗi công việc sẽ có một mã trả về khi kết thúc. Điều này sẽ giúp thực hiện các hoạt động sau này (ví dụ có phải thực hiện lại hay không).

Việc đặt trước các tài nguyên để phục vụ cho việc thực hiện các công việc sau này được thực hiện bởi các hệ thống đặt trước chuyên dụng. Đây thực chất không phải bộ lập lịch thông thường, mà trước hết nó như bộ lịch công tác quy định các công việc cụ thể trong một khoảng thời gian nhất định và bảo vệ các tài nguyên được sử dụng khỏi sự chiếm dụng của các tiến trình khác. Nó còn có khả năng dừng các công việc đang thực hiện khi tới thời gian được đặt từ trước.

1.3.6. Các thành phần truyền thông

Hệ thống lưới có thể có các phần mềm giúp các tiến trình có thể liên lạc với nhau. Ví dụ: trong trường hợp một công việc được chia thành nhiều công việc nhỏ khác nhau. Các công việc này được thực hiện riêng lẻ trên lưới. Tuy vậy, có thể các công việc này phải sử dụng kết quả của công việc khác. Điều này dẫn đến việc các công việc phải có khả năng liên kết với nhau. Các phần mềm truyền thông sẽ giúp một tiến trình có khả năng liên lạc với các tiến

trình khác, gửi và nhận các dữ liệu cần thiết. Trong hệ thống có thể sử dụng chuẩn giao diện MPI (Message Passing Interface) và một số chuẩn khác để thực hiện các liên kết này.

1.3.7. Các thành phần quản lý, theo dõi và đo lường

Như ta đã đề cập ở trên các hoạt động lập lịch đòi hỏi phải có các thông tin về hệ thống hiện tại như các máy nào sẵn dùng, tỉ lệ tài nguyên được sử dụng, tải của các nút lưới.... Thông thường các phần mềm donor sử dụng một số công cụ của hệ điều hành hoặc trực tiếp đo các thông số này. Các phần mềm này đôi khi còn được gọi là "sensor tải". Các thông tin này không những có giá trị với việc lập lịch mà nó còn đo lường được khả năng sử dụng lưới. Các thông số này có thể dự báo cho người quản trị thấy được xu hướng của lưới và các thiết bị có thể cần thiết phải thêm vào hệ thống.

Các thông tin đo lường có thể tiết kiệm cho mục đích kế toán, là cơ sở cho thủ tục môi giới tài nguyên, quản lý ưu tiên dễ dàng hơn. Người ta có thể thể hiện các thông tin đo lường dưới các dạng biểu diễn khác nhau sao cho trực quan đối với người sử dụng nhất.

1.4. Sử dụng lưới

1.4.1. Sử dụng lưới dưới góc độ người dùng

1.4.1.1. Đăng ký và cài đặt phần mềm lưới

Để sử dụng lưới tính toán, người sử dụng ban đầu phải đăng ký tham gia vào hệ thống lưới và cài đặt các phần mềm lưới cần thiết trên máy của anh ta. Người dùng cũng có thể chọn đăng ký với vai trò một máy donor cho hệ thống.

Để đăng ký vào hệ thống yêu cầu một thủ tục định danh với mục đích an toàn. Người sử dụng có thể đăng ký một account mới mà chỉ mình mới có thể sử dụng được. Các thủ tục này không nên thực hiện thông qua mạng máy tính (tương tự quá trình đăng ký một người sử dụng mới trong một mạng LAN).

Sau khi các thủ tục định danh được thực hiện, người sử dụng nhận được các phần mềm lưới để cài đặt trên hệ thống của mình với vai trò là người khai thác hệ thống hay người cung cấp tài nguyên cho hệ thống lưới. Các phần mềm này thường được thiết kế sao cho người dùng dễ thao tác nhất (chỉ cần là người sử dụng máy tính bình thường không cần có kiến thức sâu về hệ thống). Các thông tin cần thiết như thời điểm có thể sử dụng tài nguyên hệ thống, hạn chế quyền truy nhập... cũng có thể được cung cấp bởi người sử dụng khi máy tính của anh ta được cài đặt dưới dạng máy donor. Người sử dụng cũng cần thông báo cho người quản trị biết các ID của anh ta đã tồn tại trong hệ thống trên các máy khác.

1.4.1.2. Đăng nhập hệ thống

Sau khi thực hiện các thủ tục đăng ký, hầu hết các lưới đều yêu cầu người sử dụng phải đăng nhập trước khi sử dụng và khai thác hệ thống. Người sử dụng sẽ dùng định danh của anh ta khi đăng kí hệ thống. Một số các hệ thống khác sử dụng các ID khác với các ID người

sử dụng đăng nhập hệ điều hành trên máy địa phương. Thủ tục đăng nhập hệ thống thường đơn giản và thuận tiện cho người sử dụng, tuy nhiên nó phải đảm bảo rằng một người sử dụng không được đăng nhập trên nhiều máy khác nhau trên cùng hệ thống. Đối với người sử dụng, hệ thống lưới như một máy tính ảo hơn là tập hợp của nhiều máy tính bình thường.

Sau khi đăng nhập hệ thống người dùng có thể truy vấn và đệ trình các công việc của mình trên lưới. Một số mạng lưới có thể cho phép người sử dụng truy vấn và đệ trình công việc của mình mặc dù họ chưa đăng nhập hoặc thậm trí chưa đăng kí vào lưới.

1.4.1.3. Truy vấn và đệ trình các công việc

Người sử dụng có thể thực hiện các truy vấn để kiểm tra xem hệ thống có bận không và mức độ bận như thế nào, có thể biết tiến trình công việc của mình thực hiện như thế nào, và tìm kiếm các tài nguyên trên hệ thống. Các hệ thống lưới có thể cung cấp các công cụ dưới dạng đồ họa hay dòng lệnh cho phép người sử dụng thực hiện các truy vấn này. Việc sử dụng công cụ dòng lệnh cho phép người sử dụng nhập các kịch bản có thể thực hiện được để thể hiện yêu cầu của mình. Ví dụ người sử dụng có thể viết một kịch bản theo một quy tắc nhất định (dạng như SQL) để tìm kiếm các tài nguyên cần thiết.

Việc đệ trình một công việc cho lưới thực hiện bao gồm ba phần chính.

Thứ nhất các dữ liệu đầu vào và có thể một chương trình chạy hay một kịch bản được truyền tới máy để thực hiện công việc. Việc truyền các dữ liệu này được gọi là tạo kịch bản. Ở một dạng khác các dữ liệu và chương trình chạy đã được cài đặt trên một máy và được truy nhập thông qua hệ thống file mạng.

Thứ hai, thực hiện công việc. Các phần mềm lưới chạy trên máy donor thực hiện các chương trình chạy hoặc các file kịch bản dưới thẩm quyền của người sử dụng. Nó có thể sử dụng ID chung trên máy đó hoặc sử dụng chính ID của người submit công việc, phụ thuộc vào công nghệ lưới đang sử dụng. Một số hệ thống lưới tiên tiến còn có khả năng bảo vệ tức là không cho các tiến trình này phá hoại các tài nguyên trên máy donor. Quyền truy nhập hệ thống file cũng được hạn chế.

Cuối cùng, kết quả thực hiện được trả về cho người đệ trình công việc.

1.4.1.4. Cấu hình dữ liệu

Các dữ liệu truy nhập bởi các công việc trên lưới được bố trí vào và ra bởi hệ thống lưới. Tuy nhiên dựa vào kích thước và số lượng công việc sử dụng mà nó có thể phải di chuyển nhiều trên hệ thống. Chính vì điều này ta phải có một chính sách phân bố và sắp xếp dữ liệu cho hợp lý.

Ví dụ: một công việc có thể phải thực hiện nhiều lần trên một máy nào đó thì nên copy các dữ liệu mà công việc đó truy cập về máy đó để tránh chi phí truyền thông. Việc phân bố dữ liệu có thể được thực hiện nhờ hệ thống file mạng vì hệ thống này có khả năng chuyển dữ

liệu một cách hiệu quả từ một điểm trung tâm mỗi khi chương trình được thực hiện. Điều này là đúng trừ khi hệ thống lưới sử dụng công nghệ cache hay tự động nhân bản dữ liệu.
Một số nghiên cứu trong việc phân bổ và chia sẻ dữ liệu trên lưới đã giúp việc tận dụng lưới hiệu quả hơn và tránh được hiện tượng tắc nghẽn đường truyền bởi hiệu ứng thắt cổ chai (bottleneck).

1.4.1.5. Theo dõi và phục hồi

Người sử dụng sau khi đệ trình công việc có thể truy vấn xem công việc của mình được thực hiện như thế nào. Khi mà số lượng công việc trở nên rất lớn thì sẽ gặp khó khăn khi biểu diễn ra màn hình đồ họa. Khi đó mạng lưới có thể sử dụng công cụ biểu đồ để biểu diễn các độ đo trung bình của tiến trình. Việc xem một thủ tục nhỏ có được thực hiện đúng hay không lại càng gặp nhiều khó khăn hơn.
Trong một hệ thống để phục vụ cho việc lập lịch, hệ thống thường đưa ra các mức độ phục hồi cho các thủ tục bị thất bại. Các nguyên nhân thất bại thường do:
Lỗi lập trình: Thủ tục được kết thúc bởi một số lỗi chương trình.
Lỗi phần cứng hay nguồn điện: Máy tính hay thiết bị ngừng hoạt động.
Lỗi đường truyền: Đường truyền bị đứt hay nghẽn.
Tốc độ quá chậm: Chương trình có thể rơi vào vòng lặp vô hạn, hoặc đụng độ với các tiến trình khác.
Trong thực tế, không phải lúc nào ta cũng biết được nguyên nhân thất bại của công việc là do bản thân ứng dụng hay do cơ sở hạ tầng lưới. Các bộ lập lịch thường được thiết kế để phân loại các lỗi và tự động đệ trình lại các công việc này trên một nơi nào đó trong lưới với hy vọng sẽ thành công. Trong một số hệ thống, người dùng được thông báo về sự thất bại của công việc và họ phải quyết định xem có đưa ra lệnh chạy lại công việc đó hay không.

1.4.1.6. Đặt trước tài nguyên

Để nâng cao chất lượng thực hiện công việc, người dùng có thể thực hiện một số các thao tác đặt trước tài nguyên với sự độc quyền sử dụng hay một độ ưu tiên cao.Một hệ thống tương tự như các lịch làm việc được sử dụng.

1.4.2. Sử dụng lưới dưới góc độ người phát triển hệ thống

1.4.2.1. Lập kế hoạch

Người quản trị hệ thống phải có hiểu biết sâu sắc về yêu cầu của tổ chức để có thể chọn các công nghệ lưới cho phù hợp với các yêu cầu này. Trước khi triển khai một hệ thống lưới cho một tổ chức lớn, nên thực hiện xây dựng một hệ thống nhỏ hơn, học cách cài đặt và quản trị nó sau đó mới tiến hành thực hiện trên quy mô lớn, với các hệ thống thực.

1.4.2.2. Cài đặt

Bước đầu tiên hệ thống lưới được chọn phải được cài đặt với một cấu hình phù hợp. Các máy này được kết nối qua mạng với các máy khác bằng đường truyền có băng thông đủ

mạnh. Một điều tối quan trọng khi cài đặt lưới là khả năng vượt qua lỗi, tức là khi có một máy bị ngưng hoạt động do một lý do nào đó thì hệ thống vẫn có thể tiếp tục thực hiện các công việc của mình. Các máy có thể được cấu hình và kết nối để tiện lợi cho việc phục hồi và sao lưu. Các hệ thống khóa công khai phải được sao lưu thường xuyên, các khóa bí mật được bảo vệ sao cho không ai khác ngoài chủ của nó có thể truy nhập được.

Sau khi cài đặt, các máy được cấu hình với một địa chỉ mạng và các định danh tương ứng. Người quản trị có thể sử dụng quyền root để truy nhập máy quản lý hệ thống. Trong một số hệ thống người sử dụng có thể có quyền root đối với cả các máy donor, để khi cần thiết có thể cài đặt các phần mềm lưới cần thiết. Các phần mềm cài đặt trên các máy donor có thể cần cấu hình để có khả năng tìm được máy quản lý lưới và có thể chứa cả hệ thống khóa công khai của hệ thống lưới. Các phần mềm này có thể được truyền tới máy donor bằng các giao thức FTP hay bằng các phương tiện vật lý đơn giản khác như đĩa CD.

Khi lưới đi vào hoạt động, các phần mềm ứng dụng cùng các dữ liệu cần thiết có thể được cài đặt trên các máy donor. Các phần mềm này có thể bị hạn chế sử dụng bằng các giấy phép (license). Trong một số hệ thống có thể có các công cụ trợ giúp việc quản lý các giấy phép này trên phạm vi toàn lưới. Các công cụ này một mặt giúp ta khai thác đúng các phần mềm, mặt khác có các biện pháp sử dụng hiệu quả hơn các tài nguyên của lưới.

1.4.2.3. Quản trị người dùng và các máy donor

Nhiệm vụ bắt buộc phải thực hiện của người quản trị đó là quản lý các thành viên của lưới, có thể là các máy tính cung cấp tài nguyên và cả người dùng lưới nữa. Người quản trị có nhiệm vụ quản lý quyền truy nhập và sử dụng tài nguyên của người dùng thông thường. Các máy donor cũng có các quyền truy nhập nhất định và cũng cần phải quản lý. Các công việc được thực hiện trên lưới được thực hiện với một ID nhất định của người sử dụng đệ trình công việc đó. Như vậy các quyền truy nhập của công việc này phải khớp với quyền truy nhập và sử dụng tài nguyên của người sử dụng đã đệ trình nó.

Khi một người sử dụng đăng ký vào lưới hệ thống phải cung cấp các định danh và thêm vào cơ sở dữ liệu người sử dụng. Cơ sở dữ liệu này được quản trị bởi người quản trị hệ thống, có thể sẽ được truyền tới các máy khác trong hệ thống mỗi khi được cập nhật.

Tương tự như vậy, các thao tác xóa người sử dụng và các máy tính trong hệ thống cũng được thực hiện bởi người quản trị.

1.4.2.4. Các hoạt động thẩm quyền, định danh

Cần phải khẳng định rằng việc đảm bảo một cơ chế an toàn bảo mật ở mức cao nhất đối với lưới tính toán là đặc biệt quan trọng bởi vì nó không chỉ thực hiện di chuyển dữ liệu mà còn thực hiện các chương trình ứng dụng trên môi trường lưới. Đây là môi trường rất tốt cho sự tấn công của virus và các chương trình ngựa Trojan.

Dưới đây là một số hoạt động cần thực hiện các hoạt động định danh và thẩm quyền:

Thực hiện các hoạt động thẩm quyền chính xác.

Tạo mới, hủy bỏ các giấy chứng nhận.

Bảo vệ các máy chủ chứa các thông tin về thẩm quyền và định danh.

Đảm bảo không bị trùng lặp các tên của người sở hữu giấy phép.

Phục vụ thủ tục tạo chữ kí trong giấy chứng nhận.

Các hoạt động đăng nhập hệ thống.

Sử dụng kĩ thuật mã hóa theo khóa công khai.

1.4.2.5. Quản lý tài nguyên, dữ liệu

Một nhiệm vụ khác của người quản trị hệ thống đó là quản lý tài nguyên. Công việc này bao gồm việc tạo lập quyền sử dụng các tài nguyên cũng như theo dõi việc sử dụng các tài nguyên và các hoạt động thanh toán. Các thống kê về tỉ lệ sử dụng tài nguyên sẽ giúp có những đánh giá về xu hướng của hệ thống, các phần cứng cần lắp đặt thêm và các phần không sử dụng tới có thể giảm bớt để tăng hiệu quả sử dụng hệ thống.

Một số các thành phần trong hệ thống yêu cầu thiết lập các ưu tiên cho các tiến trình và người sử dụng tùy từng điều kiện và hoàn cảnh cụ thể mà người quản trị thực hiện việc cấu hình các ưu tiên này cho hợp lý. Điều này cũng được thực hiện tương tự trong các bộ phận quản lí giấy phép phần mềm.

1.4.2.6. Chia sẻ dữ liệu

Đối với các hệ thống lưới nhỏ thì việc chia sẻ dữ liệu có thể dễ dàng thực hiện được thông qua các giao thức truyền file thông thường hay sử dụng hệ thống file mạng. Tuy nhiên khi phạm vi của lưới tăng lên thì điều này chưa thực sự phù hợp. Khi đó yêu cầu đối với lưới là làm sao cho người sử dụng không phụ thuộc vào bất cứ một kho dữ liệu nào trên lưới, vì thế để nâng cao hiệu quả của lưới người quản trị phải có chiến lược sao lưu, nhân bản dữ liệu hợp lý.

1.4.3. Sử dụng lưới dưới góc độ người phát triển

Các ứng dụng của tính toán lưới có thể chia thành ba loại:

Các ứng dụng không thực hiện được với các máy đa bộ xử lý nhưng có thể được thực hiện trên nhiều máy khác nhau.

Các ứng dụng đã được thiết kế để chạy với hệ thống lưới đa bộ xử lý.

Các ứng dụng cần thay đổi và viết lại để khai thác các tài nguyên tính toán lưới.

Trong đó loại ứng dụng cuối cùng được quan tâm bởi các nhà phát triển ứng dụng lưới.

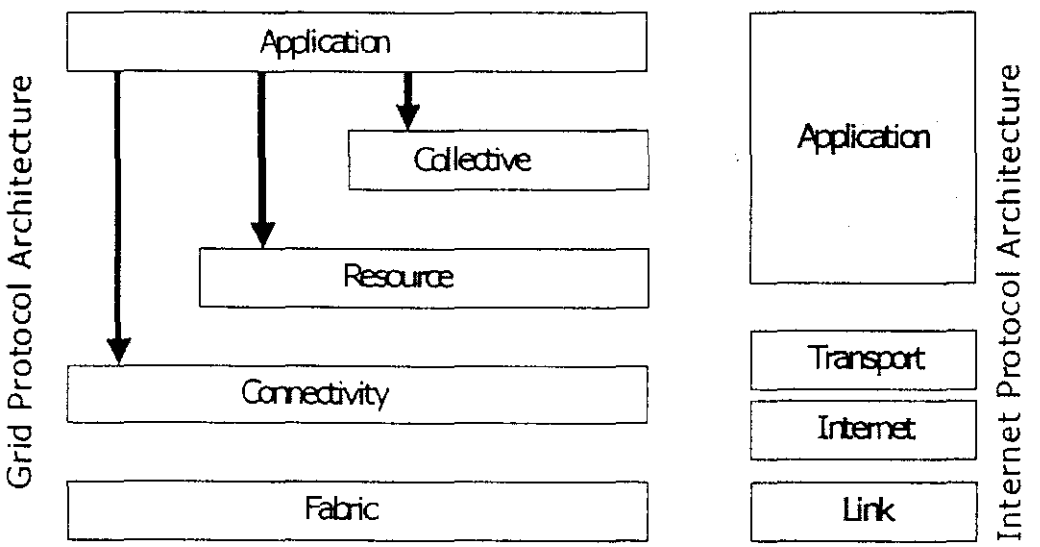
1.5. Kiến trúc chung của lưới

Trong mục này, chúng ta sẽ xem xét kiến trúc lưới với các thành phần hệ thống cơ bản, mục tiêu, chức năng của các thành phần này, và sự tương tác giữa các thành phần với nhau.

Để định nghĩa kiến trúc lưới, chúng ta bắt đầu từ một giả thiết cho rằng: có thể thiết lập các quan hệ chia sẻ giữa bất kì thành viên tiềm năng nào để đảm bảo của các thao tác hiệu quả cho tổ chức ảo.

Khả năng liên tác (Interoperability) là vấn đề trung tâm cần được bàn đến. Trong môi trường mạng, khả năng liên tác có nghĩa là các giao thức thông thường. Do vậy, kiến trúc lưới trước hết là *kiến trúc giao thức*, với các giao thức định nghĩa nhữnh kỹ thuật cơ bản giúp người sử dụng và tài nguyên của tổ chức ảo có thể đàm phán, thiết lập, quản lý, khai thác các quan hệ chia sẻ. Một kiến trúc mở dựa trên chuẩn sẽ làm cho khả năng mở rộng, khả năng liên tác, khả năng di chuyển và chia sẻ mã nguồn thuận tiện hơn. Các giao thức chuẩn sẽ giúp chúng ta dễ dàng trong việc định nghĩa các *dịch vụ chuẩn* nâng cao chất lượng hệ thống. Chúng ta cũng có thể xây dựng các APIs và SDKs để cung cấp sự trừu tượng hóa trong lập trình cần thiết cho việc tạo ra một lưới hữu dụng. Công nghệ và kiến trúc này cùng với nhau tạo thành một cái gì đó giống như middleware. Việc hình thành nên middleware là thực sự có ý nghĩa trong môi trường mạng phức tạp và không đồng nhất, vì nó giúp tạo ra khả năng độc lập giữa những người tham gia vào hệ thống, nhất là phía người sử dụng tài nguyên.

Như vậy, bản chất của kiến trúc lưới bao gồm các điểm chính đi từ khả năng liên tác, các giao thức và các dịch vụ chuẩn đến các APIs, SDKs. Với bản chất này, kiến trúc lưới được đề xuất trong được chia thành các tầng như trong hình 1-9. Trong đó, các thành phần trong một tần có một số đặc điểm chung và có thể được xây dựng trên khả năng của các tầng thấp hơn.



Hình 1-9: Kiến trúc phân tầng của lưới so sánh với kiến trúc phân tầng Internet

1.5.1. Tầng Fabric

Tầng Fabric trong lưới cung cấp các tài nguyên được gián tiếp truy nhập chung bởi các giao thức lưới; ví dụ như: các tài nguyên tính toán, các hệ thống lưu trữ, các bảng danh mục liệt kê, các tài nguyên mạng và các cảm biến. Mỗi tài nguyên có thể là một thực thể logic như là

hệ thống file, các cụm máy tính, hay là một máy tính phân tán đem góp chung. Trong những trường hợp này, việc thực thi trên tài nguyên có thể bao gồm những giao thức trong, nhưng những điều này không được xem xét trong kiến trúc lưới.

Các thành phần của tầng Fabric thực hiện các thao tác trên các tài nguyên địa phương cụ thể (vật lý hoặc logic) tương ứng với các thao tác chia sẻ ở các mức cao hơn. Các chức năng thực hiện tại tầng Fabric một mặt phụ thuộc vào nhau một cách chặt chẽ và tinh tế, mặt khác thì hỗ trợ các thao tác chia sẻ. Các chức năng của tầng Fabric càng tốt thì càng cho phép tạo nên nhiều thao tác chia sẻ tinh vi; tại cùng một thời điểm, nếu chúng ta đặt ra ít yêu cầu cho các phần tử Fabric thì quá trình triển khai cơ sở hạ tầng của lưới sẽ đơn giản đi.

Kinh nghiệm cho thấy: đối với tài nguyên, ít nhất cũng phải thực hiện được các kỹ thuật để cho biết cấu trúc, trạng thái và khả năng của chúng, bên cạnh đó là các kỹ thuật quản lý tài nguyên cung cấp việc kiểm soát chất lượng dịch vụ. Sau đây là danh sách một số đặc điểm và khả năng của một số kiểu tài nguyên cụ thể:

Tài nguyên tính toán: đó là các kỹ thuật cần thiết để khởi đầu chương trình và để giám sát cũng như điều khiển việc thực hiện tiến trình xuất kết quả. Các kỹ thuật quản lý cho phép điều khiển các tài nguyên được cấp cho trong các tiến trình cũng rất hữu hiệu, ví dụ như các kỹ thuật đặt chỗ trước.

Tài nguyên lưu trữ: đó là các kỹ thuật cần cho việc sắp đặt và lấy file, cũng như các kỹ thuật đọc và ghi một tập con của file và/hoặc các chức năng lựa chọn hay thu nhỏ đối với các dữ liệu ở xa. Các kỹ thuật quản lý tài nguyên cho phép điều khiển các tài nguyên cấp cho chuyển giao dữ liệu (như: không gian đĩa, băng thông, CPU).

Tài nguyên mạng: đó là các kỹ thuật quản lý đối với các tài nguyên được cấp cho chuyển giao trên mạng.

Các kho mã nguồn: đây là một dạng đặc biệt của các tài nguyên lưu trữ. Nó cần các kỹ thuật để quản lý phiên bản của mã nguồn và mã nguồn.

Các bảng danh mục (catalog): đây là một dạng đặc biệt của các tài nguyên lưu trữ.

1.5.2. Tầng Connectivity

Tầng này xác định các giao thức lõi về kết nối và chứng thực cho các giao dịch trên mạng. Các giao thức kết nối cho phép trao đổi dữ liệu giữa các tài nguyên của tầng Fabric. Giao thức chứng thực được xây dựng trên các dịch vụ kết nối để cung cấp các cơ chế bảo mật an toàn trong việc thẩm tra người dùng và tài nguyên.

Các giao thức kết nối bao gồm các giao thức vận chuyển (transport), tìm đường (routing) và quản lý tên (naming). Các giao thức này giả sử là được xây dựng trên các giao thức có sẵn TCP/IP (trong tương lai có thể có thêm các giao thức mới).

Các chính sách chứng thực trong môi trường các VO có những đặc tính sau :

Single sign on: Người dùng chỉ phải chứng thực một lần duy nhất để truy cập đến các tài nguyên khác nhau.

Delegation: Chương trình của người dùng được thừa hưởng tất cả các quyền hạn có thể có của người dùng để truy cập các tài nguyên. Đồng thời, tùy điều kiện mà nó có thể ủy thác một tập con các quyền cho một chương trình khác.

Tích hợp với các giải pháp bảo mật địa phương: Thực tế là các cơ chế bảo mật của lưới không thay thế toàn bộ các giải pháp bảo mật địa phương đã có mà cho phép tạo các ánh xạ vào môi trường địa phương đã có.

Các quan hệ tin tưởng giữa người dùng: Quyền truy cập các tài nguyên được cung cấp không gắn với từng người dùng cụ thể mà dựa trên mối quan hệ tin cậy lẫn nhau. Nếu người dùng A có quyền truy cập ở site S1 và S2, đồng thời S1 và S2 tin cậy lẫn nhau thì A có thể truy cập từ S1 sang S2 mà không cần phải chứng thực với S2 và ngược lại.

1.5.3. Tầng Resource

Tầng này dựa trên các giao thức truyền thông và bảo mật của tầng Connectivity để xây dựng các giao thức dùng trong việc quản lý tài nguyên như thỏa thuận an toàn, khởi tạo, theo dõi, điều khiển các chi phí trong các hoạt động chia sẻ. Các giao thức tầng này chỉ quan tâm đến việc tương tác với từng tài nguyên cụ thể.

Việc thực thi của các giao thức này sẽ gọi các hàm của tầng Fabric để truy cập các nguồn tài nguyên.

Hai lớp giao thức chính:

Information protocols (các giao thức thông tin) được sử dụng để thu được các thông tin về cấu trúc và trạng thái của tài nguyên như cấu hình, mức tải hiện thời và các chính sách sử dụng khác...

Management protocols (các giao thức quản lý) sử dụng để truy cập các tài nguyên chia sẻ, tạo các tiến trình, truy cập dữ liệu... Nó phải đảm bảo sự nhất quán dữ liệu, tính toán các chi phí, đồng thời hỗ trợ khả năng theo dõi tình trạng và điều khiển các hoạt động.

1.5.4. Tầng Collective

Trong khi tầng Resource tập trung vào sự tương tác với từng tài nguyên thì tầng này chứa đựng các giao thức và dịch vụ quản lý chung cho các nguồn tài nguyên. Các giao thức này được xây dựng dựa trên một số lượng nhỏ các giao thức nền tảng của các tầng Resource Connectivity. Chính vì vậy nó có thể thực thi nhiều dịch vụ khác nhau từ các giao thức nền tảng này.

Directory Services (các dịch vụ thư mục) cho phép tìm ra các nguồn tài nguyên và các đặc tính của nó. Một dịch vụ thư mục này có thể cho phép truy xuất các tài nguyên theo tên, kiểu...

Co-allocation, *scheduling* (lập lịch), *brokering services* (các dịch vụ môi giới) cho phép yêu cầu cấp tài nguyên cho các mục đích cụ thể và lập lịch thực thi các công việc dựa trên các tài nguyên được cấp phát.

Monitoring and diagnostics services (các dịch vụ theo dõi và chẩn đoán) cho phép theo dõi các sai hỏng, các cuộc tấn công xâm phạm, quá tải...

Data replication services (các dịch vụ nhân bản dữ liệu): cho phép tăng hiệu năng của hệ thống.

Grid-enabled programming systems : các hệ thống hỗ trợ việc lập trình trong môi trường tính toán lưới.

Workload management systems and collaboration framework cho phép quản lý các luồng công việc.

Software discovery services: cho phép chọn lựa các nền tảng thực thi tùy theo các tham số của bài toán.

Community authorization servers: có những server chuyên để cấp phép và cung cấp các chính sách bắt buộc dựa trên những giao thức về quản lý tài nguyên (trong tầng Resource) và an toàn bảo mật đề ra trong tầng Connectivity.

Community acccounting and payment services: cho phép xác định các chi phí và các giới hạn của các nguồn tài nguyên.

Các dịch vụ hợp tác

1.5.5. Tầng ứng dụng

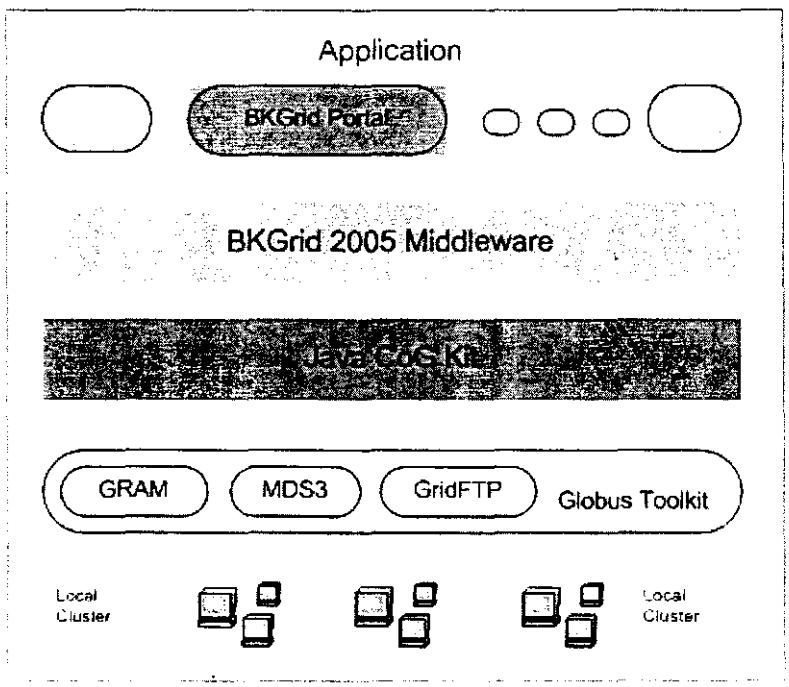
Đây là tầng trên cùng trong kiến trúc phân tầng tính toán lưới. Các ứng dụng lưới này được xây dựng trên cơ sở triệu gọi các hàm, các dịch vụ được cung cấp bởi các tầng phía dưới. Vì vậy, ở từng tầng này ta phải thiết kế và cài đặt các dịch vụ, hàm cụ thể cho các thao tác như quản lí tài nguyên, truy cập dữ liệu, tìm kiếm tài nguyên... để sao cho người sử dụng lưới cảm thấy hoàn toàn trong suốt. Người sử dụng yêu cầu chạy ứng dụng, nhận về kết quả mà không hề biết ứng dụng được chạy ở đâu trên hệ thống lưới, sử dụng tài nguyên gì, ở đâu. Vì vậy, hệ thống lưới được coi như một máy tính ảo được kết hợp bởi các tài nguyên khác nhau.

Chương 2: Giới thiệu chung về hệ thống BKGrid2005

Hệ thống BKGrid 2005 được xây dựng dựa trên kiến trúc phân tầng, các thành phần đều hướng dịch vụ, phục vụ những yêu cầu cụ thể, riêng biệt của người dùng. Một trong những điểm mạnh của hệ thống là có tính khả mở, có giao diện tiếng Việt thân thiện người dùng, và cung cấp một số tính năng cơ bản, đặc trưng của tính toán lưới, như một môi trường bảo mật an toàn cho người dùng lưới thông qua cổng điện tử portal, khả năng đệ trình các công việc từ xa để tận dụng hiệu năng tính toán của hệ cluster trong lưới, khả năng quản trị tri thức và dữ liệu, cụ thể là giải quyết một lưới các bài toán về khai phá dữ liệu, và đặc biệt là kiến trúc agent đa cấp lai với dịch vụ lưới cho phép mở rộng các tài nguyên lưới trong môi trường phân tán.

2.1. Giới thiệu

BKGrid 2005 được xây dựng như một phần mềm trung gian (middleware), bao gồm một tập các dịch vụ cơ sở ở tầng trên của bộ công cụ phát triển lưới Globus Toolkit 3.2.



Hình 2-1: BKGrid 2005 và nền tảng phát triển

Như trên hình vẽ, BKGrid 2005 phát triển trên nền tảng Globus Toolkit, là bộ công cụ để phát triển ứng dụng lưới, cung cấp một số dịch vụ cơ bản cho hệ thống như dịch vụ truyền file

GridFTP, dịch vụ quản lý tài nguyên phân tán GRAM, dịch vụ tìm kiếm và khai phá thông tin MDS3. Ngay phía trên, hệ thống sử dụng thư viện lập trình ứng dụng Java CogKit, để tương tác với các dịch vụ cơ sở mà Globus Toolkit đã cung cấp.

Hệ thống bao gồm hai thành phần chính: thành phần trung gian BKGrid 2005 Middleware cung cấp một số dịch vụ mở rộng như các dịch vụ khai phá dữ liệu, dịch vụ môi giới và quản lý tài nguyên mức cao hơn, dịch vụ bảo mật và thành phần BKGrid Portal là cổng giao tiếp các dịch vụ hệ thống cung cấp với người dùng.

BKGrid 2005 phát triển cũng dựa trên nền tảng của BK GridPortal năm ngoái, nhưng nó có những đặc điểm khác biệt cơ bản sau:

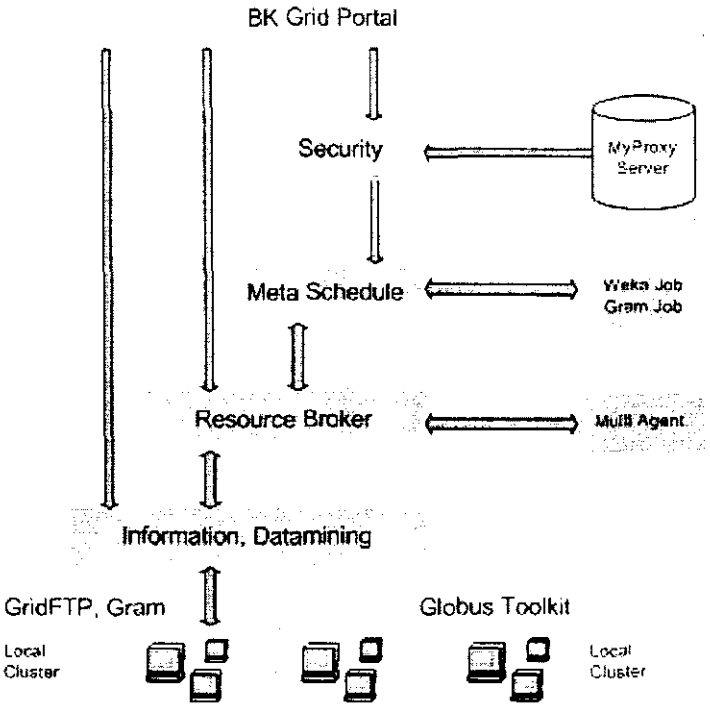
Hệ thống sẽ xây dựng trên công nghệ nền tảng Java, các thành phần đều hướng dịch vụ, tuân theo các chuẩn của dịch vụ lưới. Ngoài ra, giao diện ứng dụng của dịch vụ được sử dụng công nghệ portal, với các chuẩn portlet, dịch vụ portlet (portlet service), cung cấp một môi trường làm việc vừa thống nhất, lại vừa riêng biệt cho từng người dùng.

Hệ thống sử dụng công nghệ agent, với mô hình agent đa cấp để thương lượng và thu thập thông tin của các tài nguyên trong môi trường phân tán. Ngoài ra, agent còn được tích hợp trong các dịch vụ lưới trở thành các dịch vụ lai agent. Đây là nét mới về mặt công nghệ so với BKGrid 2004.

Hệ thống có một số chức năng mới: các dịch vụ lưới giải quyết một lớp các bài toán về khai phá dữ liệu, khả năng đệ trình công việc từ xa với các công việc đơn hoặc song song theo chuẩn MPI.

2.2. Kiến trúc hệ thống

Dưới đây là kiến trúc của BKGrid 2005 được thể hiện theo sự phân bố của các luồng thông tin dữ liệu:



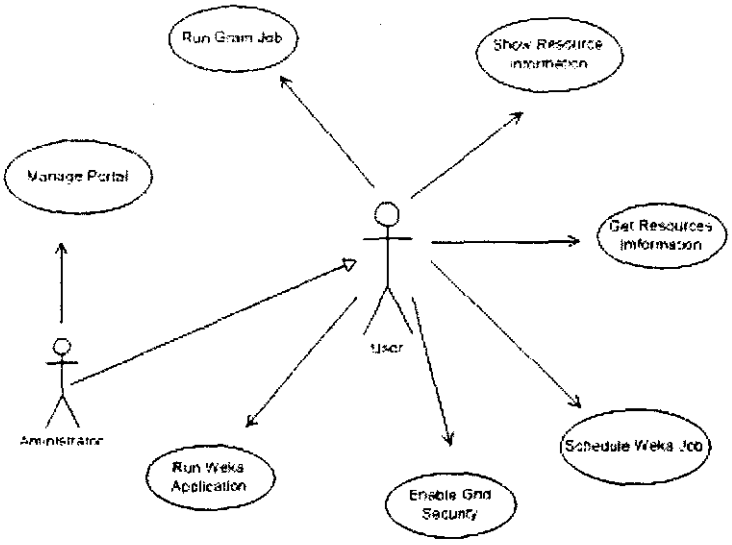
Hình 2-2: Kiến trúc BKGrid 2005 và các luồng thông tin

Dựa trên quan điểm người dùng, các luồng dữ liệu trong hệ thống được phân tầng, thể hiện sự chuyên nghiệp hóa về chức năng của từng thành phần nghiệp vụ. Cụ thể là:

- *Tầng dưới cùng:* là các hệ cluster cục bộ, với hiệu năng tính toán cao, chính là các tài nguyên tính toán của lưới. Ngay phía trên là bộ công cụ phát triển lưới Globus Toolkit.
- *Tầng thông tin và khai phá dữ liệu:* cung cấp các khả năng thông tin và khai phá dữ liệu trong các tài nguyên cục bộ, có thể cung cấp trực tiếp cho người dùng qua portal hoặc một số thông tin cơ bản về dịch vụ và hệ thống cho các tầng phía trên.
- *Tầng môi giới tài nguyên:* sử dụng công nghệ agent với kiến trúc của các hệ đa agent, thương lượng với các tài nguyên trong lưới để lấy các thông tin về hệ thống, dịch vụ mà phía dưới cung cấp
- *Tầng lập lịch mức cao:* tiếp nhận yêu cầu người dùng, sử dụng các thông tin do tầng môi giới tài nguyên cung cấp, lựa chọn tài nguyên phù hợp để thực hiện công việc. Công việc thực hiện có thể là về yêu cầu khai phá dữ liệu (weka job), hay đệ trình các công việc từ xa (Gram job), tầng lập lịch phải đảm nhận theo dõi công việc, phục hồi chống lỗi trước khi thông báo kết quả cho người dùng.
- *Tầng bảo mật:* trực tiếp hỗ trợ bảo mật cho tầng lập lịch, truy xuất các giấy ủy nhiệm người dùng qua máy chủ MyProxy, kiểm tra tính hợp lệ của người dùng trước khi cho phép người dùng đệ trình công việc xuống dưới tầng lập lịch.
- *BKGrid Portal:* phía trên cùng là cổng dịch vụ lưới BKGrid Portal, cung cấp giao diện đồ họa cho người dùng sử dụng các dịch vụ mà hệ thống cung cấp.

2.3. Các chức năng

Để hiểu rõ hơn về các chức năng, các thành phần riêng rẽ của hệ thống, sau đây là biểu đồ Use Case (UC) của BKGrid 2005:



Hình 2-3: Biểu đồ UseCase của BKGrid 2005

Trên biểu đồ Use Case, hệ thống sẽ gồm các chức năng cơ bản sau:

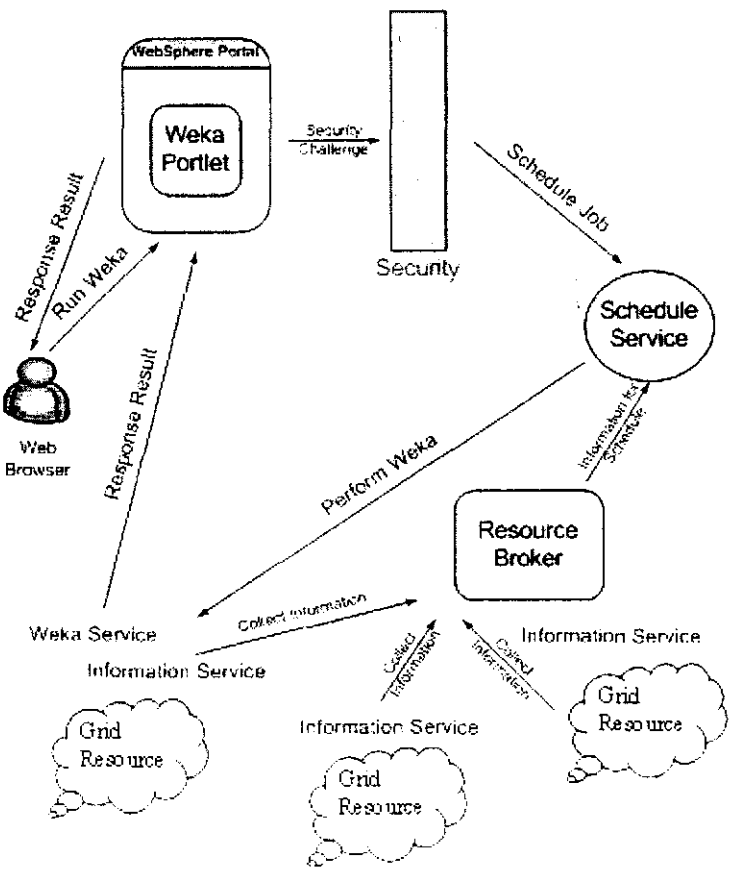
- *Chức năng bảo mật (UC Enable Grid Security):* có nhiệm vụ trao chuyển giấy ủy nhiệm hợp lệ thông qua portal, tích hợp bảo mật portal với bảo mật lưới. Dịch vụ cho phép người dùng sử dụng giấy chứng nhận của nhiều nhà CA, cho phép người dùng sử dụng nhiều giấy ủy nhiệm, cung cấp một cơ sở dữ liệu ổn định về thông tin giấy ủy nhiệm và người dùng tương ứng... và quan trọng nhất là cung cấp cho người dùng giấy ủy nhiệm hợp lệ để có thể thông hành trên môi trường lưới.
- *Chức năng thông tin (UC Show Resource Information):* Dịch vụ này sẽ thông tin cụ thể, chi tiết về tài nguyên và dịch vụ lên portal: số nút trong một tài nguyền, dung lượng, bộ nhớ,... Ngoài ra, dịch vụ cũng phải cung cấp cho bộ môi giới tài nguyên để tiến hành lập lịch.
- *Chức năng môi giới (UC Get Resources Information):* Dịch vụ này sẽ đảm nhận trách nhiệm thương lượng về chính sách bảo mật giữa các miền tài nguyên trong lưới, sau đó đưa những thông tin cụ thể, chi tiết, tổng quan về cho bộ lập lịch tiến hành công việc.
- *Chức năng lập lịch (UC Schedule Weka Job*): Dịch vụ sẽ sử dụng các thông tin do bộ môi giới cung cấp, sau đó tiến hành lập lịch để lựa chọn tài nguyên phù hợp nhất trong lưới cho người dùng khi yêu cầu dịch vụ lưới. Ngoài ra, một thành phần Client của dịch vụ cũng phải thể hiện được tiến trình theo dõi tình trạng công việc đang thực hiện (active, inactive, pending). Và một phần quan trọng nữa là đưa ra thông tin của người sử dụng dịch vụ để sau

này tiện cho việc thanh toán. Trong khi lập lịch, cũng cần chú ý tới tham số chất lượng dịch vụ QoS.

- *Chức năng Weka (UC Run Weka Application):* Đây là ứng dụng chủ chốt do portal cung cấp, tất cả các dịch vụ khác đều phục vụ cho ứng dụng này. Dịch vụ sẽ cài đặt những giải thuật cơ bản về Datamining dựa trên Weka, thể hiện phần client của mình lên giao diện portal để người dùng dễ sử dụng nhất. Các kết quả thực hiện được ở đầu ra cũng sẽ thể hiện cụ thể, sinh động để dàng thông tin cho người dùng. Nhiệm vụ của dịch vụ này là phải tương tác với dịch vụ lập lịch.

Tất cả các chức năng này đều do hệ thống BKGrid 2005 cung cấp, mỗi người dùng lưới có giấy chứng nhận hợp lệ là sử dụng được dịch vụ. Tuy nhiên, một chức năng rất quan trọng, đó là *chức năng quản lý người dùng và phân quyền (UC Manage Portal),* thì do bộ công cụ GridSphere cung cấp, sử dụng công nghệ RBAC - điều khiển truy nhập máy chủ dựa trên phân quyền (Roll-based Acess Control). Chức năng này được thực hiện trên đặc quyền của người quản trị hệ thống, các người dùng bình thường không được phép thay đổi giao diện, thành phần chức năng trên portal.

Để hiểu rõ hơn về tương tác giữa các thành phần trong hệ thống, xin đưa ra một kịch bản hoạt động trong UC Run Weka Application:

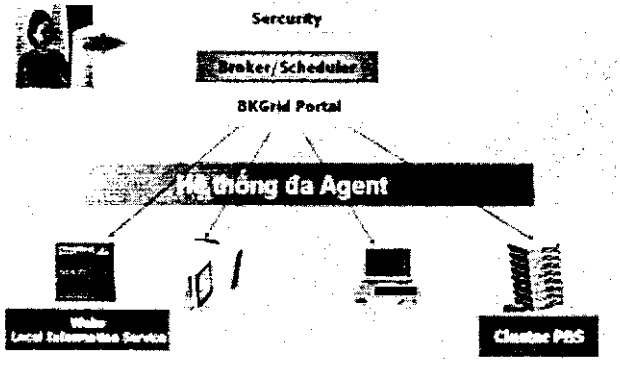


Hình 2-4: Kịch bản tương tác trong UC Run Weka Application

UC Run Weka Application là một trong những chức năng chính của hệ thống, cung cấp cho người dùng lưới một ứng dụng về DataMining. Chính vì thế mà kịch bản hoạt động của thành phần này là khá phức tạp, và liên quan đến hầu hết các thành phần khác trong hệ thống. Trên hình vẽ, có thể thấy hoạt động của UC này như sau:

- Khi người dùng có nhu cầu chạy ứng dụng Weka, thông qua một trình duyệt Web, họ sẽ đăng nhập vào portal, cung cấp các tham số cần thiết để thực hiện ứng dụng .
- Portal sẽ kiểm tra tính hợp lệ của người dùng trong lưới, sau đó chuyển yêu cầu đến bộ lập lịch.
- Bộ lập lịch sẽ lấy các thông tin cần thiết lập lịch từ bộ môi giới tài nguyên, thông tin tài nguyên được bộ môi giới tập hợp từ rất nhiều tài nguyên khác nhau, thông qua dịch vụ thông tin.
- Sau khi đã có đủ thông tin cần thiết, bộ lập lịch tiến hành lập lịch, lựa chọn một tài nguyên phù hợp nhất để thực hiện ứng dụng Weka thông qua dịch vụ Weka trong tài nguyên.
- Cuối cùng, dịch vụ Weka trả về kết quả thực hiện cho portal, rồi trả cho người dùng qua trình duyệt Web.

2.4. Hoạt động của hệ thống



Hình 2-5: Hoạt động của BKGrid 2005

Mô tả hoạt động của BKGrid 2005

1. Người dùng đăng nhập vào hệ thống thông qua portal.
2. Sau đó, muốn sử dụng các dịch vụ và tài nguyên lưới, phải qua dịch vụ bảo mật. Người dùng sẽ được cung cấp một giấy ủy nhiệm lưới từ máy chủ MyProxy
3. Tiếp đó người dùng sẽ đệ trình các yêu cầu của mình tới bộ lập lịch và môi giới tài nguyên.
4. Bộ lập lịch sẽ tương tác với một hệ đa Agent để tìm kiếm và thương lượng với các tài nguyên, sau đó lựa chọn một hoặc một tập các tài nguyên phù hợp để thực hiện các yêu cầu đó.

5. Tại các tài nguyên sẽ triển khai một tập các dịch vụ cơ bản phục vụ yêu cầu người dùng như: dịch vụ khai phá liệu, dịch vụ tính toán hiệu năng cao.

2.5. Các thành phần

2.5.1. Cổng dịch vụ lưới - BK Grid Portal

Cổng dịch vụ lưới Portal là cổng kết nối dịch vụ giữa người dùng và nhà cung cấp dịch vụ, được phát triển như một phần mềm trên mạng Intenet để cung cấp các chức năng cần thiết theo hướng người dùng. Việc sử dụng công nghệ Portal cho phép tạo môi trường làm việc riêng biệt cho từng người dùng, ở đây người dùng có thể tùy chọn một số thay đổi môi trường làm việc của mình, đồng thời tách biệt các chức năng dịch vụ riêng biệt từ phía máy chủ và tái sử dụng các thành phần chức năng của Web.

GridPortal là sự kết hợp giữa công nghệ lưới và Portal, cung cấp một cổng giao diện thống nhất cho các dịch vụ lưới, môi trường bảo mật và khả năng truy nhập các tài nguyên lưới thông qua trình duyệt Web. Hiện nay trên thế giới đang phát triển rất nhiều loại GridPortal, phục vụ cho các mục đích khoa học, nghiên cứu, mô phỏng, truyền thông và đặc biệt là các dịch vụ tính toán phân tán hiệu năng cao và lưu trữ dữ liệu. Hệ thống BKGrid 2005 phát triển dựa trên phần mềm nguồn mở Gridsphere Portal, một kiến trúc portal mới được phát triển bởi các tác giả Jason Novotny, Michael Russell và Oliver Wehrens tại học viện công nghệ Abert Einstein – Germany. Điểm mới của Gridsphere là việc tích hợp công nghệ GridPortlet cung cấp giao diện cho các dịch vụ lưới cơ bản, làm nền tảng cho phát triển các ứng dụng lưới.

BKGrid 2005 đã thừa kế các tính năng ưu việt của GridPortal này, như công nghệ điều khiển truy nhập dựa trên phân quyền RBAC, sử dụng các thẻ giao diện thuần Java Bean phát triển các Portlet làm giao diện cho dịch vụ lưới theo chuẩn JSR 168 của SUN. Thông qua Gridsphere Portal, hệ thống cung cấp các dịch vụ Porlet cơ bản về thông tin người dùng sau khi sử dụng các dịch vụ lưới, các dịch vụ quản lý và ánh xạ các thông tin bảo mật của các thực thể lưới vào trong hệ cơ sở dữ liệu ổn định HsqlDB trong hệ thống portal, cung cấp môi trường làm việc độc lập và thân thiện tiếng Việt cho mỗi người dùng.

2.5.2. Dịch vụ bảo mật

Bảo mật là một trong những khía cạnh quan trọng trong xây dựng lưới; điều này xuất phát từ bản chất của môi trường lưới là tập hợp đa tài nguyên, đa người dùng, đa miền quản trị. Xây dựng một chính sách bảo mật toàn cục cho lưới không chỉ quan tâm đến chính sách bảo mật liên miền mà cần phải tính đến các cơ chế chính sách và cơ sở hạ tầng bảo mật của địa phương của các vùng, nút là thành viên lưới. Điều này thể hiện được tính kế thừa cao, giảm bớt chi phí khi phải thay lại toàn bộ các cơ sở hạ tầng bảo mật cục bộ để phù hợp với lưới.

Hiện tại, lưới đang phát triển theo hướng dịch vụ, hướng người dùng và phục vụ các yêu cầu người dùng, người dùng có thể khởi tạo ra các dịch vụ mới mà không cần có sự can thiệp của nhà quản trị, ngoài ra các dịch vụ này còn có thể tương tác với nhau. Một yêu cầu được đặt ra là phải có cơ chế định danh các thực thể lưới (người dùng, dịch vụ, tài nguyên lưới và các tiến trình), cấp quyền cho các dịch vụ mà không ảnh hưởng tới các cơ chế bảo mật địa phương.

Các thực thể lưới được mở rộng không chỉ giới hạn các tài nguyên trong một tổ chức ảo mà ở giữa các tổ chức ảo với nhau. Như vậy đòi hỏi phải có một mô hình bảo mật hướng người dùng (user-driven security model), cho phép người dùng tạo ra các thực thể và các miền chính sách để liên kết tài nguyên trong các tổ chức ảo.

Dịch vụ bảo mật trong BKGrid 2005 được xây dựng ở tầng trên cùng của BKGrid 2005, không đi sâu vào cơ sở hạ tầng bảo mật phức tạp của các dịch vụ lưới ở phía dưới mà tập trung cung cấp hạ tầng bảo mật ở tầng trên, là thành phần trung gian để trung chuyển giữa bảo mật lưới và bảo mật portal. Nhiệm vụ chính của dịch vụ là cung cấp và trao chuyển các giấy ủy nhiệm lưới theo chuẩn X509 của người dùng, thiết lập một môi trường bảo mật an toàn, thuận tiện cho người dùng qua cổng điện tử portal, đồng thời bảo đảm đầy tính chứng thực và thẩm quyền, cũng như các khả năng bảo mật khác của các tài nguyên lưới phía dưới. Dưới đây là một số chức năng chính của dịch vụ bảo mật:

Quản lý vòng đời: Dịch vụ có khả năng cập nhật động thời gian sống của giấy uỷ nhiệm, một khoảng thời gian nhất định, giấy ủy nhiệm sẽ tự hủy nếu người dùng quên bỏ kích hoạt nó.

Cấp lại giấy ủy nhiệm: Do giấy ủy nhiệm cho người dùng có thời gian sống là hạn chế, khi người dùng sử dụng một dịch vụ tính toán nào đó với bộ dữ liệu lớn, thời gian xử lý lâu thì dịch vụ bảo mật phải có cơ chế làm tươi lại giấy ủy nhiệm, không làm ảnh hưởng tới công việc của người trên lưới.

Quản lý kho lưu trữ: Rõ ràng hệ thống lưu trữ rất nhiều giấy ủy nhiệm người dùng để ủy quyền lên portal, nên việc quản lý kho lưu trữ là rất quan trọng. Các giấy ủy nhiệm người dùng được mã hóa trong kho lưu trữ, để cho dù kho lưu trữ có bị tổn thương, kẻ địch vẫn phải mất một thời gian để giải mã, và khi đó giấy ủy nhiệm người dùng có thể sẽ hết hạn. Dịch vụ cung cấp khả năng cho phép người dùng lựa chọn các hệ thống kho lưu trữ trên các miền phân tán khác nhau về mặt địa lý qua công nghệ phân tán của dịch vụ lưới.

Quản lý giấy ủy nhiệm đa người dùng: người dùng có thể có nhiều giấy chứng nhận khác nhau, từ các nhà thẩm quyền CA khác nhau. Dịch vụ cung cấp khả năng lưu trữ tất cả các giấy chứng nhận của người dùng, đưa thông tin về công việc người dùng muốn thực hiện, lựa chọn giấy ủy nhiệm đúng đắn cho mỗi phần việc, sau đó trả lại giấy ủy nhiệm cho người dùng.

Dịch vụ bảo mật là thành phần đầu tiên và cơ bản của hệ thống, thông qua nó người dùng mới có thể triệu gọi các dịch vụ khác và truy nhập vào tài nguyên lưới, giống như một nhà

cung cấp các giấy visa để cho phép thông hành qua các nước khác nhau. Do kiến trúc mở và khả năng kế thừa các nghiệp vụ, các dịch vụ khác có thể thông qua dịch vụ bảo mật để kiểm tra tính hợp lệ của người dùng trong hệ thống lưới, biết được họ là ai và có quyền hạn gì, cũng như ghi lại các thông tin người dùng để sử dụng cho mục đích thanh toán cũng như bảo mật sau này.

2.5.3. Dịch vụ thông tin

Dịch vụ thông tin hệ thống là một thành phần quan trọng trong hệ thống giúp cung cấp đầy đủ thông tin về các tài nguyên và dịch vụ trong hệ thống để giúp các bộ lập lịch và phân phối tài nguyên có thể hoạt động hiệu quả. Trong môi trường lưới, các thông tin được cung cấp bao gồm:

Thông tin tĩnh về host: Địa chỉ IP, Tên và phiên bản hệ điều hành, Nhà cung cấp, phiên bản, tốc độ bộ vi xử lí, số lượng bộ vi xử lí,Tổng dung lượng bộ nhớ vật lí, tổng dung lượng bộ nhớ ảo, ...

Thông tin động về host: Thông tin về tải CPU,...

Thông tin về hệ thống lưu trữ: Tổng dung lượng đĩa, Dung lượng đĩa còn trống,...

Thông tin về tài nguyên tính toán: Nhà cung cấp CPU, loại CPU, tốc độ,....Loại bộ nhớ, dung lượng, tốc độ, ...Dung lượng đĩa ...

Thông tin về tài nguyên mạng: Băng thông mạng, Độ trễ Latency, Topology, Các giao thức,...

Thông tin động: Bộ nhớ vật lí còn trống, Bộ nhớ ảo còn trống

Thông tin về các dịch vụ lưới: dịch vụ thông tin sẽ tìm kiếm các dịch vụ để lấy các thông tin của các dịch vụ đó và cung cấp cho người dùng hoặc cho các ứng dụng khác trên lưới.

Dịch vụ thông tin hệ thống đóng vai trò như một giao diện cho phép người dùng hay các dịch vụ khác khai thác, theo dõi và sử dụng hiệu quả các nguồn tài nguyên. Trong bất kì môi trường lưới nào, thông tin về tài nguyên luôn dao động, phụ thuộc vào khả năng xử lí và chia sẻ dữ liệu. Khi dữ liệu được giải phóng, chúng có thể cập nhật trạng thái của chúng trong dịch vụ thông tin. Các phần mềm phía máy khách, bộ môi giới tài nguyên và các bộ quản lí tài nguyên có thể sử dụng các thông tin này để phục vụ các mục đích riêng của chúng. Về cơ bản, các chức năng của dịch vụ thông tin hệ thống bao gồm:

+ Tạo và quản lí các thông tin động về hệ thống trong từng tài nguyên cục bộ thông qua các bộ cung cấp thông tin.

+ Tập hợp các dữ liệu dịch vụ từ các thực thể dịch vụ khác nhau.

+ Đăng kí các thực thể dịch vụ lưới với trình chứa dịch vụ.

+ Cung cấp khả năng truy cập vào các thông tin tĩnh và thông tin động về các thành phần hệ thống cho người dùng cũng như cho các dịch vụ khác.

Dịch vụ thông tin là một trong các thành phần cốt lõi của tính toán lưới. Một mặt, dịch vụ thông tin trả lời trực tiếp các câu truy vấn từ bất kì ứng dụng nào, cung cấp các thông tin cho bộ định vị tài nguyên mặt khác là nguồn thông tin không thể thiếu được trong các bộ môi giới tài nguyên. Trong các hệ thống lưới thực tế, việc theo dõi trạng thái lưới là công việc thường xuyên của những người quản trị lưới điều này hoàn toàn có thể được trợ giúp bởi các dịch vụ thông tin từ đó người quản trị lưới sẽ có những chính sách nhằm điều chỉnh các tài nguyên trong lưới.

2.5.4. Dịch vụ môi giới tài nguyên

Agent là một công nghệ phân tán, cho phép tương tác, thương lượng với các tài nguyên một cách linh hoạt qua cơ chế truyền thông điệp (messages). Việc kết hợp các agent ngang hàng, khả năng phân cấp giữa các agent với nhau để tạo thành các hệ đa agent là một trong những đặc trưng nổi bật của agent, giúp cho việc tập hợp các tài nguyên không đồng nhất về mặt địa lý trong môi trường phân tán, để cùng phục vụ cho một mục đích chung.

Công nghệ Agent được sử dụng trong hệ thống BKGrid 2005, với chức năng chính là môi giới các tài nguyên trong lưới, thu thập các thông tin động cũng như thông tin tĩnh về dịch vụ và hệ thống qua dịch vụ thông tin, sau đó cung cấp cho bộ quản lý tài nguyên để tiến hành lập lịch. Ngoài ra một ứng dụng rất quan trọng khác của Agent là quản lý từng tài nguyên và dịch vụ đơn lẻ. Từng tài nguyên và dịch vụ sẽ có một Agent đại diện và Agent này sẽ chịu trách nhiệm thay mặt người sở hữu tài nguyên đó định ra các chính sách sử dụng, thương lượng với Agent người dùng. Agent quản lý tài nguyên cũng chịu trách nhiệm thực hiện một số tính năng nâng cao khác của môi trường lưới như tính năng đặt trước tài nguyên.

Đây được xem một thành công khá đáng kể về mặt công nghệ của nhóm phát triển BK Grid 2005 vì thực tế trên thế giới việc nghiên cứu ứng dụng Agent vào lưới cũng mới chỉ đang ở trong giai đoạn khởi đầu. Việc thiết kế các dịch vụ lai Agent (dịch vụ lưới tích hợp với công nghệ Agent) cho phép tạo ra các dịch vụ rất linh hoạt gọn nhẹ kết hợp được các ưu điểm của cả hai mô hình lập trình là mô hình Agent và mô hình dịch vụ lưới. Trong tương lai, một hướng phát triển của agent là kết hợp trong các dịch vụ bảo mật, thương lượng với các chính sách bảo mật địa phương, thực hiện việc cập nhật động các danh sách điều khiển truy nhập cục bộ, tạo điều kiện thuận lợi cho các nhà quản trị đưa ra các chính sách bảo mật toàn cục trong môi trường lưới.

2.5.5. Bộ lập lịch

Xét từ phía ứng dụng, bộ lập lịch và môi giới tài nguyên là một trong những thành phần trung tâm của BK Grid, bảo đảm chất lượng dịch vụ và hiệu năng hệ thống khi cung cấp cho người dùng. Nhiệm vụ của bộ lập lịch và môi giới tài nguyên bao gồm:

+ Khám phá tài nguyên.

+ Chọn tập tài nguyên phù hợp để thực hiện yêu cầu của người dùng theo những tiêu chí tối ưu mà người dùng đặt ra.

+ Theo dõi quá trình thực hiện ứng dụng và có những xử lý cần thiết khi có những tình huống bất thường xảy ra. Ví dụ: tài nguyên bị lỗi, có tài nguyên mới trong quá trình thực hiện.

+ Ghi lại những thông tin về lịch sử xử dụng của người dùng làm cơ sở cho việc thanh toán.

+ Hiển thị thông tin chi tiết về quá trình thực hiện ứng dụng

Trong BK Grid 2005 bộ lập lịch được tích hợp chặt chẽ với ứng dụng cụ thể là ứng dụng Weka. Có thể gọi bộ lập lịch này là bộ lập lịch ở mức người dùng (User level scheduler) hay là bộ lập lịch ở mức ứng dụng (Application level scheduler). Đặc điểm của các bộ lập lịch dạng này là tuy không mang tính tổng quát và phải được thiết kế cụ thể cho từng kiểu dạng ứng dụng tuy nhiên đó cũng chính là một ưu thế của chúng: do hiểu rõ và được thiết kế chuyên biệt cho từng ứng dụng nên chúng có điều kiện để thực hiện các phép tối ưu cho từng kiểu loại ứng dụng cụ thể. Là thành phần giữ vị trí trung tâm, các bộ lập lịch và môi giới tài nguyên có các mối liên hệ với dịch vụ thông tin hệ thống và các dịch vụ Weka.

Để cho các bộ lập lịch hoạt động hiệu quả thì yêu cầu đầu tiên là phải có đầy đủ thông tin về môi trường lưới. Yêu cầu này cũng đúng cho bất kỳ cơ chế quản lý tài nguyên nào dù là tập trung hay phân tán. Có lẽ khó ai có thể phủ nhận vai trò của thông tin trong công tác quản lý điều hành. Tuy nhiên như đã đề cập ở các phần trên, do tính chất rộng lớn và phức tạp của môi trường lưới mà việc có đầy đủ các thông tin một cách kịp thời là rất khó khăn do có quá nhiều yếu tố bất cập như đường truyền mạng, lỗi tài nguyên. Do vậy trong phạm vi của mình các dịch vụ thông tin sẽ cung cấp một cách đầy đủ nhất các thông tin về tài nguyên và dịch vụ trong phạm vi quản lý của mình để làm đầu vào cho các bộ lập lịch. Về phía các bộ lập lịch và môi giới tài nguyên cũng được thiết kế để có thể hoạt động trong môi trường thông tin kém đầy đủ và chính xác.

Các dịch vụ thông tin trong BK Grid được triển khai trên từng tài nguyên nhằm cung cấp các thông tin về chính tài nguyên đó cũng như các dịch vụ được triển khai trên nó. Các thông tin này có thể là thông tin tĩnh như kiểu loại, kiến trúc tài nguyên (máy kiến trúc nào, hệ điều hành gì), kiểu loại dịch vụ cung cấp (dịch vụ tính toán hay lưu trữ, cách thức sử dụng, giá cả, chất lượng dịch vụ...) hoặc có thể là các thông tin động như tải CPU, dung lượng bộ nhớ/đĩa cứng trống, các tham số về chất lượng phục vụ hiện thời của dịch vụ như số lượng người dùng, tốc độ xử lý...

Việc thu thập các thông tin trên toàn bộ các tài nguyên sẽ do một tập các Agent đảm nhiệm. Trong BK Grid bộ lập lịch và môi giới tài nguyên không tương tác trực tiếp với các dịch vụ thông tin mà thông qua hệ đa Agent này để lấy thông tin, như dịch vụ thông tin hệ thống (SystemService), nhóm các dịch vụ lưới Weka (ClassifierService, ClusterService, AsociationRuleService), ...

Người dùng khi sử dụng các dịch vụ khai phá dữ liệu của Weka sẽ không tương tác trực tiếp với các dịch vụ tính toán đầu cuối này mà sẽ thông qua đại diện của mình là bộ lập lịch và môi giới tài nguyên. Bộ lập lịch nhận các yêu cầu từ người dùng cùng một tập các dữ liệu đầu vào sau đó lựa chọn các dịch vụ tính toán phù hợp để thực hiện sao cho tối ưu được các tiêu chí mà người dùng đã đặt ra. Ví dụ: tối ưu hóa về giá cả, thời gian...Việc kết hợp các dịch vụ tính toán đầu cuối Weka lại sẽ mang lại một ứng dụng tinh xảo hơn và nhiều lựa chọn hơn cho người dùng Vd: cùng lúc sử dụng nhiều dịch vụ tính toán đầu cuối để xử lý dữ liệu, lựa chọn phương thức xử lý tối ưu cho từng loại công việc như tiết kiệm thời gian hay chi phí.

2.5.6. Các dịch vụ khai phá dữ liệu

Hệ thống BKGrid 2005 đã tiến hành lưới hóa các mô đun của ứng dụng Weka thành các dịch vụ lưới. Weka là một chương trình nguồn mở do trường đại học Wakailo, New Zealand phát triển với mục đích phục vụ cho bài toán khai phá dữ liệu và đặc biệt là khâu phân tích dữ liệu. Weka là một ứng dụng tương đối tốt trong lĩnh vực khai phá dữ liệu và đặc biệt là phù hợp khi triển khai trên lưới. Bởi lẽ, chương trình Weka được thực thi dưới dạng lô, tức là mỗi khi có yêu cầu khai phá dữ liệu, chương trình sẽ nhận đầu vào và thực hiện việc xử lý mà không có sự tương tác trực tiếp với người dùng, sau đó nếu thành công thì kết quả được trả về hoặc nếu thất bại thì thông báo lỗi cho người sử dụng. Thứ hai, do nhu cầu xử lý khối lượng rất lớn dữ liệu đầu vào nên chương trình Weka được thực thi trên máy đơn sẽ không có hiệu năng như mong muốn. Trên thực tế, chương trình này đang được phát triển tiếp theo hướng song song hóa. Chính vì vậy, hệ thống BKGrid 2005 sẽ lưới hoá chương trình weka để thể hiện các nghiên cứu về phát triển ứng dụng lưới. Các dịch vụ lưới cho Weka được chia thành hai nhóm:

+ *Nhóm các dịch vụ ở mức tài nguyên.* Đây là các dịch vụ cơ bản tương ứng với các thuật toán khai phá dữ liệu trong Weka: classifiers, clusters, filters.... Các dịch vụ mức tài nguyên được triệu gọi bởi các dịch vụ mức ứng dụng.

+ *Nhóm các dịch vụ mức ứng dụng.* Đây là các dịch vụ mà người sử dụng tương tác trực tiếp bằng cách cung cấp một tập rất lớn dữ liệu đầu vào nhằm mục đích khai phá dữ liệu. Sau khi nhận đầy đủ các yêu cầu từ người sử dụng như dữ liệu đầu vào, kiểu thuật toán, các ràng buộc về chi phí, thời gian thực hiện... dịch vụ sẽ lấy các thông tin về tài nguyên trên lưới, thương lượng và lập lịch thực hiện công việc. Để có thể xử lý tập dữ liệu đầu vào, các dịch vụ ở mức tài nguyên sẽ được kích hoạt. Trong quá trình xử lý, người sử dụng có thể tương tác trực tiếp với dịch vụ như huỷ yêu cầu, thêm yêu cầu hay thay đổi chiến lược tối ưu của họ. Chính vì vậy, dịch vụ phải có mức trừu tượng cao, người sử dụng không cần quan tâm đến chi tiết xử lý của dịch vụ. Dịch vụ này phải được thể hiện trên giao diện.

2.5.7. Dịch vụ tính toán

Bên cạnh các dịch vụ Weka, BK Grid 2005 còn cung cấp dịch vụ chạy ứng dụng từ xa. Người dùng chỉ việc viết sẵn ứng dụng theo chuẩn MPI sau đó upload lên portal của BK Grid, dịch vụ tính toán trên cluster sẽ chịu trách nhiệm điều khiển thực thi ứng dụng này và trả lại kết quả đã xử lý cho người dùng. Mô-đun đệ trình công việc lên lưới đưa ra nhằm đơn giản hóa quá trình thực hiện công việc cho người dùng. Mô-đun này cung cấp các chức năng sau:

+ Cung cấp giao diện Web thân thiện cho phép người dùng người dùng đệ trình công việc lên lưới.

+ Chạy được các ứng dụng đơn và các ứng dụng song song lập trình theo thư viện truyền thông điệp MPI.

+ Quản lý tình trạng các công việc do người dùng đệ trình lên lưới.

+ Lưu lại kết quả trên máy chủ và trả lại kết quả cho người dùng khi có yêu cầu.

Việc triệu gọi ứng dụng từ xa trở nên trong suốt đối với người dùng không chuyên về lưới nhờ việc giải phóng họ khỏi các công việc đặc thù của lưới như. Để sử dụng mô-đun này trước tiên người sử dụng phải thực hiện đăng nhập vào lưới, đưa các thông tin yêu cầu và ủy quyền cho lưới thực hiện công việc, mô-đun đệ trình công việc sẽ nhận các thông tin này, sử dụng dịch vụ lập lịch lưới thực hiện tìm kiếm thông tin về các tài nguyên lựa chọn tài nguyên thích hợp nhất để thực hiện công việc. Môđun đệ trình công việc thực hiện theo dõi tình trạng công việc và trả lại kết quả cho người dùng khi có yêu cầu.

2.6. Tham gia BKGrid2005

Quá trình triển khai kết hợp tính toán song song phân cụm vào tính toán lưới đòi hỏi người thực hiện phải có những kiến thức khá sâu về cả hai mô hình tính toán này. Đây thực sự là một điều khó khăn đối với những người mới làm quen với tính toán lưới và tính toán song song phân cụm. Shell script thực hiện cấu hình tự động được đưa ra nhằm đơn giản hóa vấn đề này, đối với cluster based-on PBS ta thực hiện thiết lập các thông số cho máy chủ PBS, tạo ra các hàng đợi định tuyến (routing queue) và các hàng đợi thực hiện công việc (execution queue) còn đối với grid based-on Globus ta thực hiện thiết lập các thông số để làm việc với cluster như shell thực hiện truy nhập, thông số kiểm tra hàng đợi, v.v... Do vậy nhóm phát triển BK Grid 2005 đã phát triển một công cụ cho phép cấu hình tự động hệ thống cluster PBS mới vào lưới. Các tính năng chính của mô-đun này là:

+ Thiết lập các hàng đợi thực thi công việc (execution queue) với các mức độ ưu tiên khác nhau để thực hiện các công việc khác nhau tùy theo khoảng thời gian tương ứng.

+ Thiết lập một hàng đợi định tuyến (routing queue) dùng để kiểm tra các công việc rồi đưa công việc vào các hàng đợi thực thi tương ứng.

+ Thực hiện thiết lập các thông số cho PBS_Server để nó hoạt động hiệu quả.

+ Cấu hình để Grid based-on Globus hiểu tài nguyên nó sẽ thực hiện là một cluster.

Chi tiết kĩ thuật về việc xây dựng và triển khai các thành phần trong hệ thống BKGrid2005 sẽ là nội dung của các chương tiếp theo.

Chương 3: Cổng dịch vụ lưới - BK Grid Portal

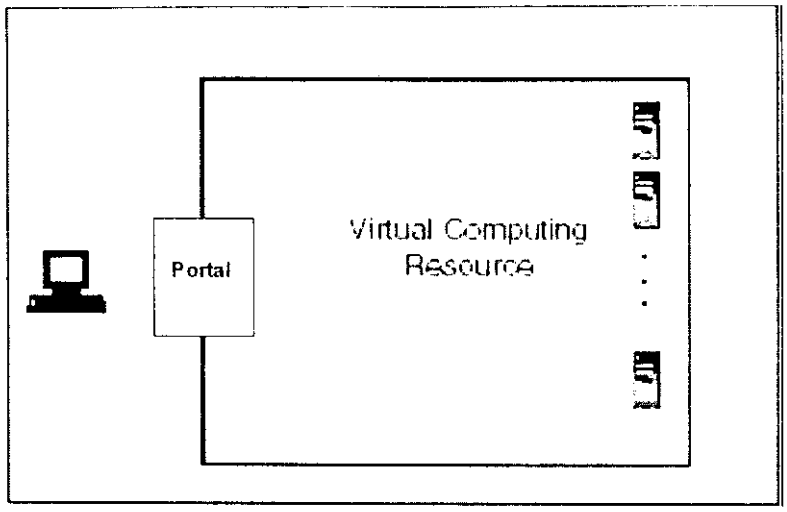
Cổng dịch vụ lưới Portal là cổng kết nối dịch vụ giữa người dùng và nhà cung cấp dịch vụ, được phát triển như một phần mềm trên mạng Intenet để cung cấp các chức năng cần thiết theo hướng người dùng. Việc sử dụng công nghệ Portal cho phép tạo môi trường làm việc riêng biệt cho từng người dùng, ở đây người dùng có thể tùy chọn một số thay đổi môi trường làm việc của mình, đồng thời tách biệt các chức năng dịch vụ riêng biệt từ phía máy chủ và tái sử dụng các thành phần chức năng của Web.

GridPortal là sự kết hợp giữa công nghệ lưới và Portal, cung cấp một cổng giao diện thống nhất cho các dịch vụ lưới, môi trường bảo mật và khả năng truy nhập các tài nguyên lưới thông qua trình duyệt Web. Hiện nay trên thế giới đang phát triển rất nhiều loại GridPortal, phục vụ cho các mục đích khoa học, nghiên cứu, mô phỏng, truyền thông và đặc biệt là các dịch vụ tính toán phân tán hiệu năng cao và lưu trữ dữ liệu. Hệ thống BKGrid 2005 phát triển dựa trên phần mềm nguồn mở Gridsphere Portal, một kiến trúc portal mới được phát triển bởi các tác giả Jason Novotny, Michael Russell và Oliver Wehrens tại học viện công nghệ Abert Einstein – Germany. Điểm mới của Gridsphere là việc tích hợp công nghệ GridPortlet cung cấp giao diện cho các dịch vụ lưới cơ bản, làm nền tảng cho phát triển các ứng dụng lưới.

BKGrid 2005 đã thừa kế các tính năng ưu việt của GridPortal này, như công nghệ điều khiển truy nhập dựa trên phân quyền RBAC, sử dụng các thẻ giao diện thuần Java Bean phát triển các Portlet làm giao diện cho dịch vụ lưới theo chuẩn JSR 168 của SUN. Thông qua Gridsphere Portal, hệ thống cung cấp các dịch vụ Porlet cơ bản về thông tin người dùng sau khi sử dụng các dịch vụ lưới, các dịch vụ quản lý và ánh xạ các thông tin bảo mật của các thực thể lưới vào trong hệ cơ sở dữ liệu ổn định HsqlDB trong hệ thống portal, cung cấp môi trường làm việc độc lập và giao diện tiếng Việt thân thiện cho mỗi người dùng.

3.1. Giới thiệu chung về Grid Portal

Nhiệm vụ của portal là cung cấp cho người dùng sự thuận tiện, dễ dàng để khai thác các dịch vụ lưới. Người sử dụng không cần quan tâm đến cấu trúc, độ phức tạp của lưới hay nói cách khác thông qua portal mà môi trường lưới là ảo đối với người sử dụng. Hiện tại GT3 chưa hỗ trợ portal. Vì vậy, khi thiết kế ứng dụng, người thiết kế có thể thêm thành phần này vào ứng dụng nếu như người sử dụng không có kiến thức về lưới hoặc chưa biết các thao tác với các thành phần trong lưới.



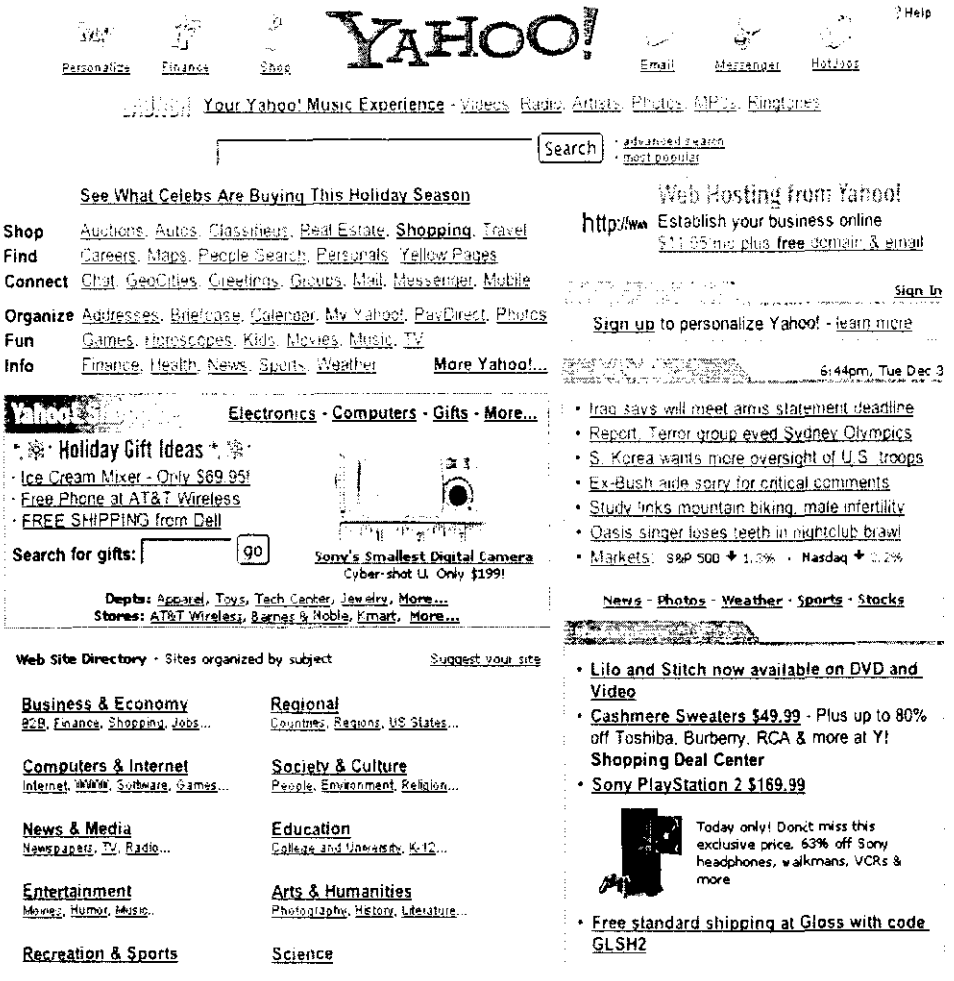
Hình 3-1: Mô hình portal trong lưới

Một giao diện lưới có thể được cấu trúc như giao diện Web tạo thuận tiện cho người sử dụng

3.1.1. Portal

Chúng ta có một định nghĩa tổng quát về portal như sau :

Portal là một ứng dụng chạy trên nền web mà có khả năng tích hợp và cá nhân hoá các ứng dụng, thông tin và các dịch vụ cộng tác. Portal cung cấp cho người dùng một điểm truy cập đơn tới các nguồn dữ liệu, nội dung và các dịch vụ đa dạng của một đơn vị nghiệp vụ (business) hay các tài nguyên trên mạng internet.

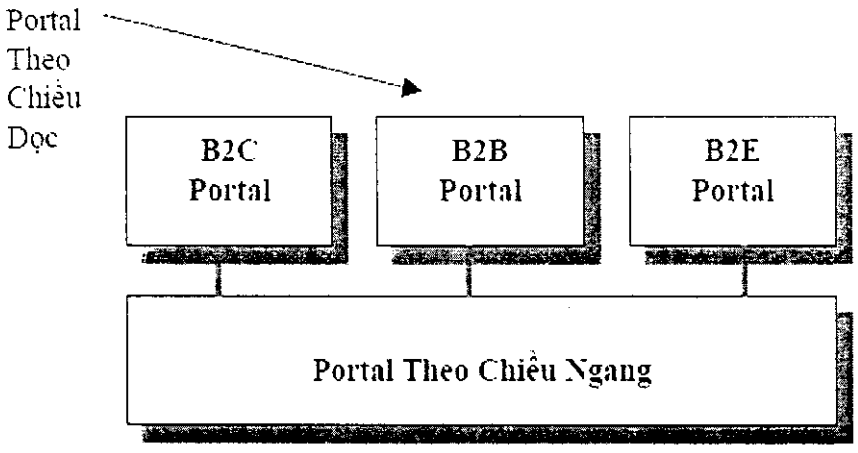


Hình 3-2: Ví dụ một Portal

3.1.1.1. Phân loại Portal

Phân loại Portal theo chiều ngang (Horizontal portals):Đây là loại Portal kết nối tới hầu hết các tài nguyên của các cơ quan, doanh nghiệp, tổ chức, cá nhân và cung cấp các dịch vụ chung để có thể tích hợp các Portal theo chiều dọc.

Phân loại Portal theo chiều dọc (Vertical portals): Loại Portal này tập trung phục vụ cho các nhu cầu của một bộ, một ngành, ban, hay một doanh nghiệp cụ thể, với các ứng dụng cụ thể phù hợp với cơ cấu tổ chức cũng như các thực trạng của cơ quan, tổ chức, doanh nghiệp đó.



Hình 3-3: Phân loại Portal

3.1.1.2. Tiêu chuẩn của một Portal

+ Quản trị dễ dàng: Thông qua một giao diện web thống nhất để thực hiện tất cả các công việc quản trị hệ thống như cấp quyền đăng nhập, phân nhóm người sử dụng... và ngay cả các tác vụ quan trọng như sao lưu dữ liệu, nhân bản và kiểm tra tính nhất quán của dữ liệu.

+ An toàn: Đạt tiêu chuẩn SSO (single sign-on- đăng nhập hệ thống 1 lần duy nhất), hệ thống portal phải giao tiếp tốt với cơ chế bảo mật LDAP hay cơ chế bảo mật tương tự khác. Có thể quản trị cả những người sử dụng đang làm việc tại các máy tính trên Internet, bên ngoài firewall.

+ Tùy biến dễ dàng: Người sử dụng có khả năng tự động cấu hình portal để chỉ cung cấp hình thức và số lượng thông tin mình cần, tùy thuộc vào vai trò, vị trí, nhu cầu thông tin của người sử dụng.

+ Chia sẻ thông tin: Người sử dụng trong portal phải có đầy đủ các công cụ như: thư tín điện tử, hội thảo nhóm... để trao đổi, chia sẻ thông tin với mọi người trong hệ thống.

+ Khả năng mở rộng hệ thống: Tiêu chuẩn này còn tùy thuộc vào từng sản phẩm portal. Nếu nhu cầu chỉ là một vài chức năng hỗ trợ kinh doanh thì các sản phẩm như SAP, Siebel là đạt yêu cầu. Nhưng nếu portal được tích hợp thêm các API hay tiêu chuẩn XML thì người sử dụng tự do xây dựng cơ chế cung cấp thông tin theo sở thích cá nhân.

+ Tính khả chuyển: Vì hệ thống sẽ phục vụ hàng ngàn đối tượng khác nhau nên sản phẩm portal phải hỗ trợ cơ chế xử lý phân luồng, mô hình dữ liệu phân tán, cân bằng động, nhân bản...

+ Sự hữu dụng: Các portal, nhất là portal hoạt động theo mô hình B-to-C (nhà kinh doanh-người tiêu dùng) do trình độ, nhu cầu của người sử dụng rất khác nhau, nên portal phải có thể tùy biến giao diện, hình thức thể hiện thông tin, có công cụ tìm kiếm thông tin hữu hiệu.

+ Giá cả: Giá thành một portal không chỉ bao gồm chi phí mua bản quyền phần mềm portal, mà còn phải tính cả các chi phí về đào tạo, tinh chỉnh hệ thống....

Ở góc độ hệ thống mạng nội bộ thì portal đảm nhận nhiệm vụ đảm bảo thông tin cho công tác chuyên môn. Ở góc độ khách hàng hay đối tác kinh doanh thì portal chính là nơi doanh nghiệp tiếp xúc, trao đổi và thực hiện giao dịch với đối tác, khách hàng của mình. Khi một doanh nghiệp nào đó có portal thì tất cả mọi giao dịch đều được thực hiện trên một giao diện duy nhất nhưng thông tin được thể hiện theo ý thích đến tận cá nhân người sử dụng. Và portal cũng làm cho quá trình trao đổi thông tin giữa các nhân viên trong một doanh nghiệp với khách hàng hay đối tác được nhanh chóng, thuận tiện và ít tốn kém.

Đó là lợi ích hứa hẹn của hệ thống portal, nhưng việc xây dựng một hệ thống portal hoàn toàn không nhẹ nhàng chút nào. Khác với việc xây dựng một web site, xây dựng một portal không chỉ là xây dựng cơ sở hạ tầng mà còn bao gồm việc xác định và lập danh sách các chức năng chuyên môn chính cần phải thực hiện trên portal, và đây chính là bước quan trọng nhất.

Một portal cơ bản không được làm yếu đi hay gây quá tải khả năng cung cấp thông tin của hệ thống, thông tin lại phải được cung cấp tùy theo chức năng, nhu cầu của từng cá nhân cụ thể. Bất cứ một người sử dụng nào, chỉ cần đăng nhập vào hệ thống một lần duy nhất là có thể giao tiếp và khai thác mọi thông tin theo yêu cầu chuyên môn của họ, không cần biết thông tin đó được lấy ở đâu, từ nguồn nào trong hệ thống.

Portal còn làm thay đổi khái niệm xây dựng, phân phối và phát triển một ứng dụng phần mềm. Người xây dựng, nhà phát triển portal phải nắm vững khái niệm về dịch vụ web là gì, như thế nào là một ứng dụng trên nền web và tuân thủ một loạt các tiêu chuẩn dành riêng cho mô hình portal như Microsoft .Net, Sun Java System, WSRP và hàng loạt những đặc tả về Java Portlet (JSR 168, 170, 188, 207). Việc tuân thủ tốt các tiêu chuẩn chung và thống nhất sẽ làm cho các ứng dụng có thể giao tiếp và trao đổi thông tin lẫn nhau trong một portal cho dù chúng được xây dựng hay tích hợp từ nhiều nguồn khác nhau (xem mục 'Tiêu chuẩn đặc trưng cho portlet, ứng dụng chức năng chạy trên portal' để biết thêm chi tiết). Hệ thống các tiêu chuẩn chung và phân chia một cách khoa học cũng giúp người xây dựng portal tự do tìm kiếm và chọn lựa các ứng dụng thích hợp với yêu cầu chuyên môn của mình, không cần quan tâm đến ai là người sản xuất ra phần mềm chức năng đó. Như vậy, thay vì tốn công, tốn tiền của cho công tác đào tạo, huấn luyện mỗi khi chuyển sang sử dụng công nghệ mới, thì từ nay người phát triển ứng dụng tập trung khai thác tối đa kiến thức, năng lực sẵn có của họ để xây dựng hệ thống ứng dụng.

Các giải pháp xây dựng portal hàng đầu hiện nay hoặc dựa trên tiêu chuẩn J2EE chạy trên máy chủ như WebSphere của IBM, hoặc theo mô hình của WebLogic của BEA, hoặc là chuẩn .Net của Microsoft. Ngoài ra cũng có thể sử dụng kết hợp các giải pháp với nhau. Nhưng cho dù portal được xây dựng từ giải pháp nào đi nữa thì nó cũng phải đảm bảo các

yếu tố như có khả năng tích hợp, khai thác tốt cơ chế bảo mật và mô hình cấu trúc trên hệ thống sẵn có của người sử dụng.

3.1.2. Grid Portal

Grid Portals xây dựng dựa trên mô hình Web portal quen thuộc, như là Yahoo hay Amazon, cung cấp các khả năng của lưới cho cộng đồng người dùng thông qua giao diện Web, cung cấp các khả năng truy nhập tới các dịch vụ và tài nguyên lưới.

Grid portal được xem là một môi trường giải quyết vấn đề, cho phép các nhà khoa học lập trình, truy nhập và thực thi các ứng dụng lưới phân tán từ một trình duyệt web hay từ các công cụ trên desktop khác. Trong một lớp của Grid portal - "cổng dịch vụ khoa học" các tri thức khoa học và các công cụ được thể hiện dưới dạng các ứng dụng khoa học chứ không phải dưới dạng các cổng dịch vụ tính toán phân tán phức tạp. Điều này cho phép các nhà khoa học có thể tập trung vào các vấn đề khoa học vì Grid đã trở thành một môi trường trong suốt đối với họ. Một phân lớp khác của Grid portal - "cổng dịch vụ người dùng" cung cấp cho người dùng trực tiếp các tài nguyên và các dịch vụ của lưới. Loại portal này yêu cầu người dùng phải có những hiểu biết sâu sắc về công nghệ tính toán lưới.

Một số dự án Grid Portals

- Grid-Port:

Frame work dựa trên Perl phát triển bởi Mary Thomas và Steve Mock tại San-Diego Supercomputing Center (SDSC)

- Grid Portal Development Toolkit (GPDK):

Phát triển bởi Jason Novotny tại Lawrence Berkeley National Laboratories (LBNL)

- Astrophysics Simulation Collaboratory (ASC):

Phát triển bởi Michael Russell tại University of Chicago, hiện tiếp tục phát triển bởi Greg Daues tại National Super-Computing Alliance (NCSA

3.1.3. Các yêu cầu đối với một Grid Portal

Từ phía người dùng, các yêu cầu chính đối với một hệ thống Grid portal chủ yếu liên quan tới việc truy cập tới các dịch vụ lưới.

3.1.3.1. Các dịch vụ bảo mật

Người dùng sẽ đăng nhập vào một portal bằng một trình duyệt web và sử dụng xác thực user-id và mật khẩu (passowrd)để xác thực. Mặc dù hiện nay có rất nhiều các công nghệ chứng thực mới và tốt hơn, nhưng sử dụng id và password vẫn là phương thức được áp dụng rộng rãi nhất. Khi hệ thống cần bảo mật ở mức độ cao hơn thì chúng ta có thể sử dụng các công nghệ khác như : thẻ thông minh (smart card), chứng thực bằng sinh trắc học,...

Một khi người dùng đã được xác thực sẽ có một công việc của portal server hoạt động như một proxy trong hầu hết các tương tác lưới. Chính vì vậy portal server phải có được một giấy chứng thực proxy để nó có thể đại diện cho người dùng. Cách thông thường để đạt được điều này là người dùng đệ trình một proxy lên một "MyProxy" server. Sau đó portal server sẽ lấy proxy từ MyProxy server và giữ nó trong suốt phiên giao dịch của người dùng. Quá trình tạo một cặp khóa quản lý người dùng và đệ trình giấy chứng thực proxy lên MyProxy server thực sự là rất xa lạ đối với đại đa số người dùng. Do đó chúng ta mong muốn có một dịch vụ ủy nhiệm độc lập để quản lý các giấy chứng thực và các cặp khóa cho người dùng. Hi vọng rằng một chương trình như vậy sẽ sớm được đề xuất. Một mảng nữa của bảo mật là quản lý việc nhận dạng nhóm. Lưới thường được tổ chức theo mô hình các tổ chức ảo (Virtual Organization) trong đó chứa một nhóm các cá nhân và các viện nghiên cứu, các tổ chức liên kết với nhau.

3.1.3.2. Quản lý file từ xa

Các công cụ để truy nhập các thư mục và các file là một yêu cầu chính đối với một portal. Các công cụ đơn giản cho GridFTP là cần thiết nhưng có thể các file được quản lý bởi một hệ thống dữ liệu ảo - nơi mà dữ liệu được phân loại, thực thi bởi các dịch vụ lưới nền (back-end grid services). Portal mà người dùng cần truy nhập các hệ thống và công cụ sẽ cần phải có công cụ để quản lý file từ xa.

3.1.3.3. Quản lý các công việc từ xa

Khả năng đệ trình các công việc lên lưới để thực thi và theo dõ là một yêu cầu kinh điển đối với một portal. Người dùng với những tài nguyên cụ thể sẽ muốn biết được hàng đợi công việc dùng những tài nguyên này và muốn tham khảo các công cụ lập lịch. Họ cũng muốn theo dõi việc thực thi các công việc và biết được những sai sót bằng cách đọc các biên bản ghi nhớ (log). Các ứng dụng lưới lớn thì sự thực thi của chúng thường được định nghĩa bằng các luồng công việc phức tạp. Công việc của portal không phải là thực thi hay quản lý các luồng công việc này. Sẽ có nhiều ngôn ngữ được đề xuất và sử dụng vào việc đó. Tuy nhiên việc lập trình các luồng công viêc và các hệ thống thực thi chúng được triển khai dưới dạng các dịch vụ và portal phải cung cấp được khả năng truy nhập đến chúng.

3.1.3.4. Truy nhập tới các dịch vụ thông tin

Một vai trò cũng rất quan trong của portal đó là khả năng truy nhập tới các thư mục và các công cụ chỉ mục. Mỗi một người dùng nên có một nơi lưu trữ riêng tư, lâu dài các tham chiếu tới các thông tin quan trọng họ lưu trữ trên lưới. Các cơ chế truyền tin và thông báo là trung tâm đối với việc quản lý các ứng dụng lưới. Người dùng có thể đăng ký với các dịch vụ thông báo và sẽ có các thông báo được lưu trữ, có thể truy nhập được thông quan các chỉ mục tìm kiếm.

3.1.3.5. Các giao diện ứng dụng

Đối với các portal khoa học thì các chi tiết của lưới nên trong suốt với người dùng và được dấu sau các giao diện ứng dụng. Người dùng cần phải có được khả năng chạy ứng dụng, cấu hình và điều khiển từ xa các ứng dụng như là cách họ đã quen thuộc khi sử dụng các ứng dụng desktop.

3.1.3.6. Truy cập cộng tác

Bất kỳ một tổ chức ảo nào cũng phải có khả năng truy nhập các tài nguyên chia sẻ. Điều này bao gồm khả năng sử dụng các công cụ cộng tác trong thời gian thực cũng như các cộng tác không đồng bộ.

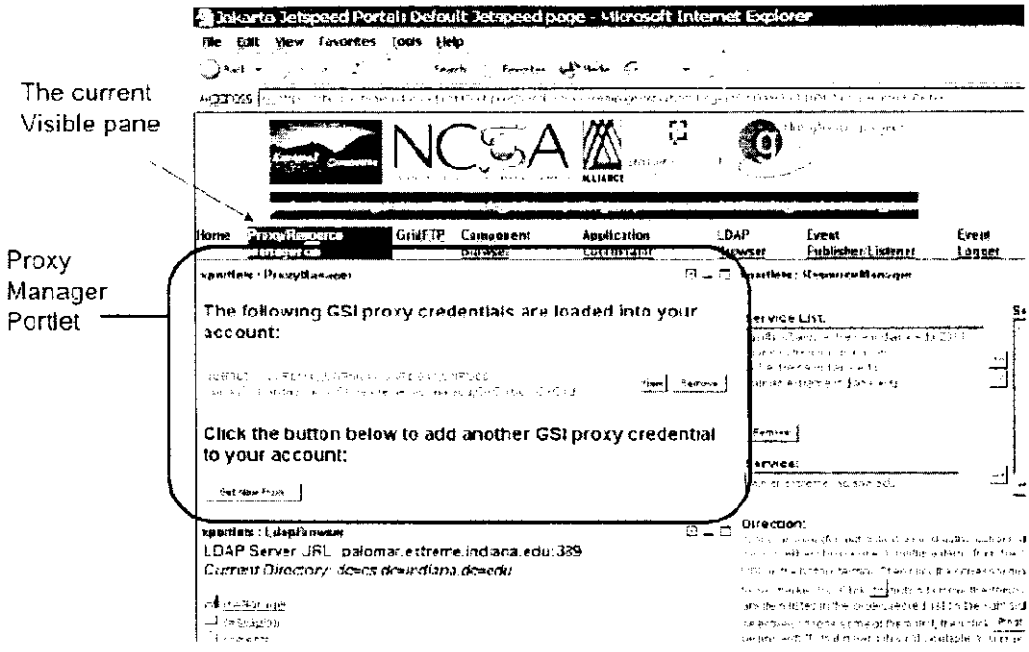
3.1.4. Kiến trúc của Grid portal theo hướng tiếp cận portlet

Kiến trúc của Grid được định nghĩa bằng một tập hợp các dịch vụ lưới từ xa. Kiến trúc của portal được phát triển dựa trên ý tưởng một portal server là một container cho các client của dịch vụ lưới. Các client này được thiết kế theo mô hình portlet.

Portlet là một chuẩn mở được đưa ra cho các bộ công cụ để phát triển Web Portal và Grid Portal, với mục đích để đóng gói, triển khai và chuyên nghiệp hóa các thành phần nghiệp vụ web. Portlets đưa ra API cho việc xây dựng các thành phần giao diện ảo trong nội dung trang Web và các nhà cung cấp dịch vụ. Mỗi portlet cung cấp một cửa sổ nhỏ trong một trang portal, và có thể có nhiều portlet trong một trang portal.

Portlet là một thành phần server điều khiển một của sổ nhỏ được cấu hình bởi người dùng trong một ô nhỏ (pane) của trình duyệt web. Portlet có đặc tả tiêu chuẩn và được hỗ trợ bởi IBM, SUN, BEA và Oracle. Ngoài ra cũng có một tiêu chuẩn được hỗ trợ bởi Apache trong dự án Jetspeed, Michigan Chef server và GridLab Gridsphere.

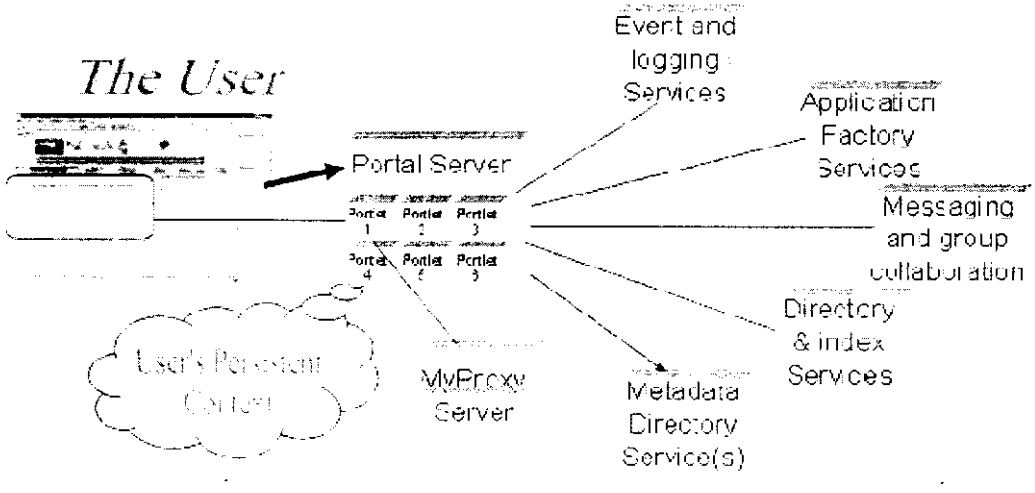
Trong những hệ thống này, portal server cung cấp cho người dùng một tập hợp các frame mà mỗi frame này chứa một hoặc nhiều portlet. Ngườid ùng có thể cấu hình các portal frame cho phù hợp với mình. Chính điều này đã tạo hướng tiếp cận hướng thành phần đối với việc xây dựng portal. Bằng việc cung cấp cho người dùng các porlet cơ bản (các porlet cung cấp thành phần tương tác của các ứng dụng và dịch vụ lưới), người dùng có thể quyết định làm thế nào để tổ chức một môi trường tốt nhất cho mình.



Hình 3-4: Ví dụ một porlet (Proxy Manger) được cấu hình trong frame, nằm cùng nhóm với các porlet khác dưới tiêu đề Proxy Resource Manager

Ngoài ra kiến trúc porlet còn có rất nhiều các ưu điểm. Mỗi một dịch vụ lưới có thể được kết hợp với một portlet riêng biệt. Do đó chúng ta có thể dễ dàng thêm các dịch vụ mới và đưa các nhóm porlet khác nhau vào portal.

Cuối cùng mỗi một người dùng có thể chọn lựa, cấu hình các porlet mà họ muốn và sự lựa chọn này trở thành một phần trong ngữ cảnh người dùng. Điều này được thể hiện trong hình vẽ sau :



Hình 3-5: Cái nhìn từ phía người dùng,portal như một tập các giao diện dịch vụ

3.2. Gridsphere Portal

Hệ thống BKGrid 2005 sử dụng Gridsphere Portal của các tác giả Jason Novotny, Michael Russell và Oliver Wehrens tại học viện Albert Einstein, Đức. Đây là một portal mã nguồn mở được sử dụng cho môi trường lưới. Gridsphere hoạt động như một ứng dụng web, đòi hỏi một máy chủ, ví dụ như Jakarta Tomcat.

3.2.1. Các đặc điểm của Gridsphere Portal

+ Hỗ trợ hầu hết các hệ CSDL chính bao gồm MySQL, Postgres, DB2, HsqlDB, ...

+ Hỗ trợ các chuẩn Portlet mới nhất hiện nay: JSR 168 Portlet API, Portlet API IBM's WebSphere 4.2, đồng thời dễ dàng phát triển và tương tác với "third-party portlets".

+ Sử dụng các visual beans và thư viện thẻ giao diện người dùng của Gridsphere (UI) để phát triển các portlet phức tạp, đặc tả các xếp đặt hiển thị thông qua XML một cách linh hoạt

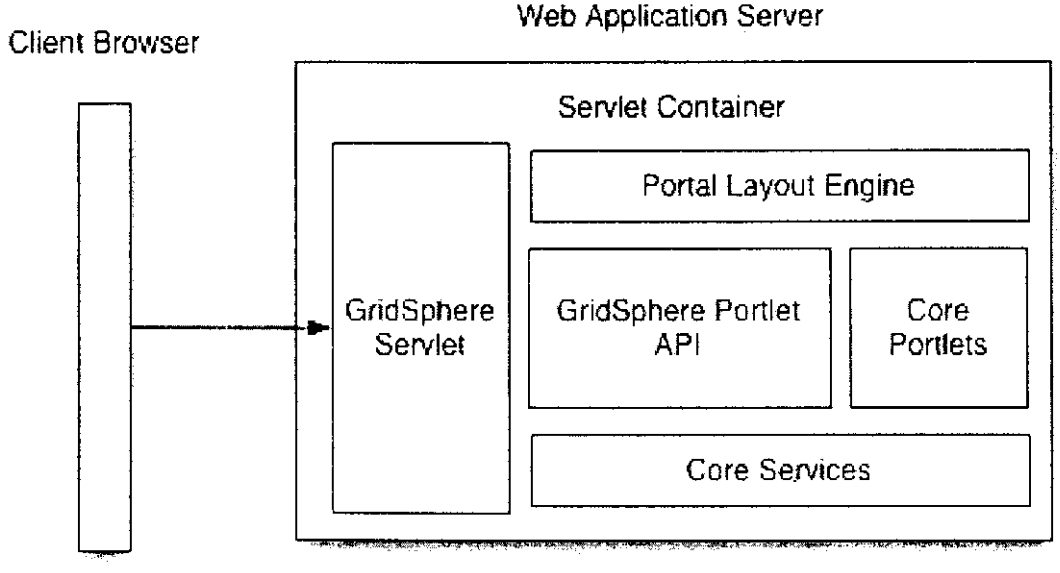
+ Cơ chế điều khiển truy nhập dựa trên phân quyền với các loại người dùng: guests, users, admins and super users.

+ Mô hình dịch vụ portlet (portlet service) cung cấp các chức năng có thể tái sử dụng cho nhiều portlet.

+ GridSphere core portlets bao gồm các tính năng như: đăng nhập, thoát, người dùng và điều khiển truy nhập

3.2.2. Mô hình Gridsphere

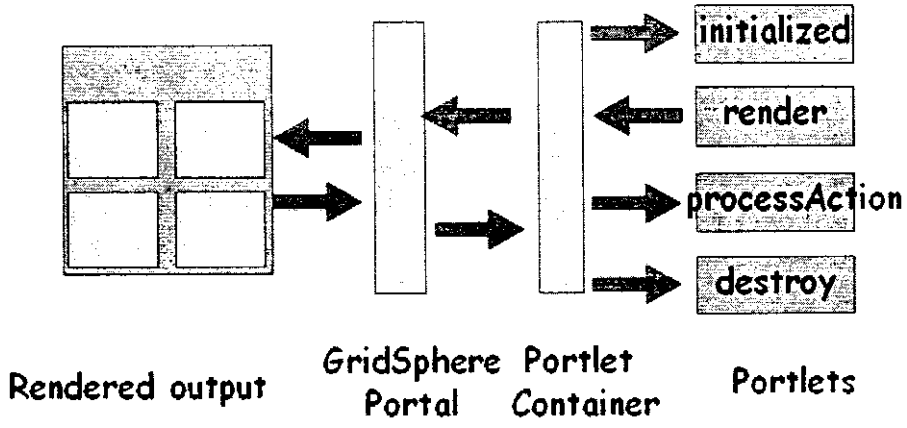
Gridsphere hoạt động như một ứng dụng Web, đòi hỏi môi trường trình chủ như Jakatar Tomcat.



Hình 3-6: Mô hình Gridsphere

3.3. Portlet Service (Dịch vụ portlet)

Grid Portlet hoàn toàn giống như các Portlet, nhưng thể hiện giao diện cho các dịch vụ lưới thông qua Portal. Chính vì đặc trưng này, khi phát triển các Grid Portlet, đòi hỏi phải có một số cấu hình ở máy chủ Web để hiểu được các đặc tả giao diện dịch vụ theo chuẩn GWSDL khi triệu gọi các dịch vụ lưới từ xa.



Hình 3-7: Vòng đời của Portlet trong Gridsphere Portal

Chương 4: Dịch vụ bảo mật

Dịch vụ bảo mật trong BKGrid 2005 được xây dựng ở tầng trên cùng của BKGrid 2005, không đi sâu vào cơ sở hạ tầng bảo mật phức tạp của các dịch vụ lưới ở phía dưới mà tập trung cung cấp hạ tầng bảo mật ở tầng trên, là thành phần trung gian để trung chuyển giữa bảo mật lưới và bảo mật portal. Nhiệm vụ chính của dịch vụ là cung cấp và trao chuyển các giấy ủy nhiệm lưới theo chuẩn X509 của người dùng, thiết lập một môi trường bảo mật an toàn, thuận tiện cho người dùng qua cổng điện tử portal, đồng thời bảo đảm đầy đủ tính chứng thực và thẩm quyền, cũng như các khả năng bảo mật khác của các tài nguyên lưới phía dưới. Đây là thành phần đầu tiên và cơ bản của hệ thống, thông qua nó người dùng mới có thể triệu gọi các dịch vụ khác và truy nhập vào tài nguyên lưới, giống như một nhà cung cấp các giấy visa để cho phép thông hành qua các nước khác nhau. Do kiến trúc mở và khả năng kế thừa các nghiệp vụ, các dịch vụ khác có thể thông qua dịch vụ bảo mật để kiểm tra tính hợp lệ của người dùng trong hệ thống lưới, biết được họ là ai và có quyền hạn gì, cũng như ghi lại các thông tin người dùng để sử dụng cho mục đích thanh toán cũng như bảo mật sau này.

4.1. Bảo mật trong môi trường lưới

Do đặc điểm hỗn tạp và không đồng nhất của các tổ chức và tài nguyên trong lưới, vấn đề bảo mật trong lưới là một trong những vấn đề được quan tâm hàng đầu. Có những vấn đề bảo mật mới chưa từng gặp trong các công nghệ bảo mật hiện tại cho hệ thống tính toán phân tán truyền thống. Ví dụ, các hệ thống tính toán song song đòi hỏi nhiều tài nguyên tính toán, dẫn tới nhu cầu phải thiết lập các mối quan hệ bảo mật, không đơn giản chỉ là với client và server, mà giữa hàng trăm tiến trình thực hiện trong không gian tập hợp nhiều miền quản trị. Ngoài ra, cần phải có các chính sách bảo mật liên miền cho lưới, các công nghệ điều khiển truy nhập giữa các miền khác nhau cũng phải được hỗ trợ.

Các ứng dụng và hệ thống lưới có thể đòi hỏi bất cứ chức năng nào trong các chức năng cơ bản của bảo mật như là: chứng thực, điều khiển truy nhập, toàn vẹn, bí mật. Khi phát triển kiến trúc bảo mật lưới, cũng cần phải lựa chọn giải pháp để đáp ứng được đòi hỏi của các đặc tính rất riêng của lưới:

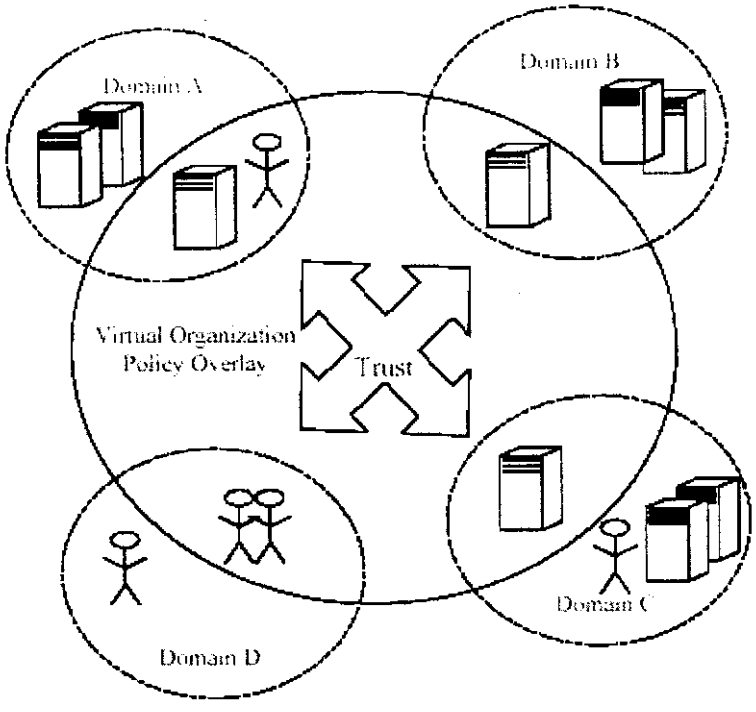
+ Đăng nhập một lần: Khi bắt đầu một tính toán đòi hỏi sử dụng tài nguyên, cho thuê tài nguyên hay truyền thông nội bộ, người dùng có thể được chứng thực, và sẽ không phải chứng thực trong các tính toán tiếp theo.

+ Giấy ủy nhiệm người dùng (mật khẩu, khóa bí mật, ...): phải được bảo vệ bằng các chính sách như mã hóa, hệ thống file bảo mật, phân quyền, ...

+ Tích hợp các giải pháp bảo mật địa phương: các giải pháp bảo mật liên miền phải tích hợp với các giải pháp bảo mật địa phương để đảm bảo độc lập của các thành viên lưới.
+ Cơ sở hạ tầng giấy ủy nhiệm, giấy chứng nhận thống nhất: Truy nhập liên miền đòi hỏi phải có một quy ước thống nhất để biểu diễn định danh của các thực thể lưới như là người dùng, tài nguyên,... Vì thế, cần có một chuẩn để mã hóa các giấy chứng nhận cho mục đích bảo mật. Hiện tại, X509 là chuẩn cho các giấy chứng nhận phổ biến trong môi trường lưới.
+ Hỗ trợ bảo mật nhóm truyền thông: Một tính toán có thể đòi hỏi một số các tiến trình, cùng cộng tác các hoạt động của chúng với nhau như là một nhóm. Tổ hợp các nhóm tiến trình sẽ thay đổi trong vòng đời của một tính toán. Vì thế, cần cung cấp bảo mật truyền thông nhóm động. Không có giải pháp bảo mật nào hiện tại hỗ trợ tính năng này, thậm chí là thư viện lập trình bảo mật GSS-API còn không cung cấp bảo mật nhóm.
+ Độc lập công nghệ: các chính sách bảo mật không phục vụ cho một công nghệ phát triển ứng dụng cụ thể nào. Hơn nữa, có thể cài đặt các chính sách bảo mật trong một phạm vi các công nghệ bảo mật, dựa trên cả kĩ thuật mã hóa công khai và phân phối khóa công khai.

4.1.1. Các thách thức bảo mật trong môi trường lưới

Các yêu cầu bảo mật lưới ở trên được định hướng để cung cấp các tổ chức ảo phân tán, rộng lớn để chia sẻ và sử dụng các nguồn tài nguyên đa dạng trong một mô hình thống nhất. Tuy nhiên, về khía cạnh bảo mật, các tài nguyên cũng như các thành phần khác tham gia lưới lại bị quản lý bởi các nội quy và các chính sách của một tổ chức truyền thống mà chúng là thành viên. Do vậy, để các tổ chức ảo truy nhập vào các tài nguyên trong các tổ chức truyền thống, chúng phải được thiết lập và cộng tác qua mối quan hệ tin tưởng hai bên, tồn tại giữa người dùng với các tổ chức truyền thống của họ và mối quan hệ giữa người dùng với các tổ chức ảo. Chúng ta không thể thiết lập quan hệ tin tưởng trực tiếp giữa các tổ chức truyền thống với tổ chức ảo hay các thành viên mở rộng của nó.
Cơ chế bảo mật lưới giải quyết các trở ngại này bằng cách cho phép có một tổ chức ảo thống nhất chung một phần chính sách của các tổ chức truyền thống (policy domain overlay). Cơ chế được thể hiện như hình vẽ:



Hình 4-1: Một tải chồng miền chính sách của tổ chức ảo đưa các miền chính sách phân tán vào trong một miền tin tưởng chung.

Các tài nguyên và các tổ chức đưa ra các điều khiển chính sách mở rộng (outsource policy) cho một bên thức ba, các tổ chức ảo (VOs), phối hợp các chính sách mở rộng trong một miền tin tưởng ổn định lâu dài, để cho phép chia sẻ tài nguyên và sử dụng. Giải pháp tải chồng các chính sách dẫn tới các chức năng chủ yếu sau mà bảo mật lưới phải thực hiện:

+ Hỗ trợ nhiều cơ chế bảo mật khác nhau: Các miền tài nguyên hay các tổ chức ảo thường đã có sự đầu tư đáng kể trong các cơ chế bảo mật và cơ sở hạ tầng bảo mật của địa phương họ. Do vậy mà thách thức lớn nhất chính là phải liên kết các công nghệ bảo mật trên các địa phương hơn là thay thế toàn bộ nó, như thế sẽ rất tốn kém và hoàn toàn không có tính kế thừa.

+ Khởi tạo động các dịch vụ: Người dùng có thể khởi tạo ra các dịch vụ mới mà không cần có sự can thiệp của nhà quản trị, ngoài ra các dịch vụ này còn có thể tương tác với nhau. Như vậy là phải có cơ chế định danh các thực thể lưới, cấp quyền cho các dịch vụ mà không ảnh hưởng tới các cơ chế bảo mật địa phương. Một ví dụ trong cơ sở hạ tầng bảo mật GSI, khi một dịch vụ lưới cung cấp cho người dùng, các định danh về người dùng sử dụng dịch vụ, định danh của dịch vụ, định danh của hệ thống mà dịch vụ đăng ký trên đó đều được xác định rõ ràng.

+ Thiết lập động các miền chứng thực tin tưởng (trust domain): Việc chứng thực không chỉ được thiết lập giữa người dùng và tài nguyên trong một tổ chức ảo mà còn mở rộng giữa các tổ chức ảo với nhau. Như vậy đòi hỏi phải có một mô hình bảo mật hướng người dùng

(user-driven security model), cho phép người dùng tạo ra các thực thể và các miền chính sách để liên kết tài nguyên trong các tổ chức ảo.

4.1.2. Các chính sách bảo mật trong môi trường lưới

Một số định nghĩa quan trọng :

+ Chủ thể (Subject): Là một thành viên của các hoạt động bảo mật, đối với môi trường lưới chủ thể thường là người dùng, tài nguyên hay các tiến trình thay mặt cho các tài nguyên đó.

+ Giấy ủy nhiệm (Credential): là một phần thông tin, dùng để cung cấp định danh cho chủ thể để xác định tên và vai trò của chủ thể đó trong lưới.

+ Thẩm quyền (Authentication): là tiến trình để chủ thể chứng minh định danh của mình cho đối tượng được yêu cầu. Thẩm quyền hai bên (bên yêu cầu và bên được yêu cầu) là quá trình hai bên thẩm quyền lẫn nhau, còn gọi là thẩm quyền đa phương (multual authentication).

+ Đối tượng (Object): là các tài nguyên được bảo vệ bởi một chính sách bảo mật địa phương cụ thể.

+ Chứng thực (Authorization): là tiến trình mà thông qua đó, ta xác định được một chủ thể có được phép truy nhập và sử dụng tài nguyên hay không.

+ Miền tin tưởng (trust domain): là cấu trúc quản lý mức logic, do một chính sách bảo mật ổn định, đơn lẻ mức địa phương nắm giữ, hay nói cách khác, nó là một tập các chủ thể và đối tượng được quản lý bởi đơn miền quản trị và chính sách bảo mật cục bộ.

Sau đây là các chính sách bảo mật giải quyết các yêu cầu thách thức trình bày ở phần trên.

- Môi trường lưới bảo mật đa miền: Do lưới là một tập hợp không đồng nhất của các người dùng và tài nguyên cục bộ, cho nên các chính sách bảo mật cục bộ dành cho các tài nguyên và người dùng cũng khác nhau, chính sách bảo mật lưới phải đảm bảo tích hợp được tất cả các tập hợp không đồng nhất này. Nói chung, môi trường lưới không hạn chế hay không ảnh hưởng tới các chính sách bảo mật địa phương, nhiệm vụ của chính sách bảo mật lưới là phải tập trung điều khiển các tương tác liên miền, ánh xạ các hoạt động liên miền vào trong các chính sách bảo mật địa phương. Ví dụ trong cơ sở hạ tầng GSI, các hoạt động liên miền được thực hiện bởi các chủ thể sở hữu một giấy chứng nhận lưới theo chuẩn X509. Trong từng miền cụ thể, các giấy chứng nhận này sẽ được ánh xạ tương ứng với một người dùng cục bộ nào đó thông qua một file ánh xạ (grid-mapfile), là một bản ghi chứa tên người dùng cục bộ và định danh của giấy chứng nhận đó.

- Hoạt động lưới hạn chế trong đơn miền quản trị: Mặc dù lưới là một tập đa miền quản trị, tuy nhiên các hoạt động đa miền lại phải tuân theo các chính sách bảo mật địa phương trên đơn miền quản trị. Nói cách khác, không có hoạt động bảo mật hay dịch vụ lưới nào được đưa vào các hoạt động địa phương thông qua các chính sách bảo mật của lưới.

- Các chủ thể toàn cục và cục bộ đều tồn tại: Tại mỗi đơn miền quản trị đều tồn tại hai chủ thể trên, và chính sách để ánh xạ từ một phần tử toàn cục vào phần tử cục bộ. Để làm ví dụ, mỗi người dùng đều có hai tên, một tên toàn cục để hoạt động trên tất cả các tài nguyên, và một tên cục bộ trên mỗi tài nguyên. Ánh xạ tên toàn cục vào tên cục bộ tạo khả năng đăng nhập một lần (sigle-sign-on) trên môi trường lưới. Trong cơ sở hạ tầng GSI, tên toàn cục chính là tên định danh của giấy ủy nhiệm X509, và tên cục bộ là tên người dùng trong hệ điều hành.
- Chứng thực đa phương: Hoạt động giữa các thực thể định vị trong các miền tin tưởng khác nhau đòi hỏi chứng thực đa phương, bảo đảm cho sự an toàn và bí mật của các hoạt động. Ví dụ trong dịch vụ truyền file GridFTP, cả client và server đều phải chứng minh định danh của mình trong lưới, client đòi hỏi server có định danh như mình mong muốn không, còn server sẽ kiểm tra danh sách các định danh client, xem client có quyền đăng nhập vào server để sử dụng dịch vụ truyền file không.
- Mỗi đối tượng toàn cục được ánh xạ vào đối tượng cục bộ được coi như chúng đã qua chứng thực địa phương trên đối tượng cục bộ đó.
- Tất cả các quyết định điều khiển được đưa ra đều là cục bộ hay dựa trên cơ sở của đối tượng cục bộ, hay không có một quyết định điều khiển nào là toàn cục, áp dụng cho tất cả các tài nguyên cục bộ. Ví dụ, một người dùng lưới có thể sử dụng dịch vụ truyền file GridFTP tại một tài nguyên này, nhưng tại các tài nguyên khác trong lưới, anh ta sẽ không có quyền truy nhập. Anh ta không thể ra lệnh truyền file cho toàn bộ các tài nguyên trong lưới.
- Có thể dùng chung tập giấy chứng nhận với các chương trình thay mặt cho cùng một tiến trình, chạy trên cùng một chủ thể trong cùng một miền tin tưởng. Như đã biết, tính toán lưới liên quan tới hàng trăm tiến trình chạy trên một tài nguyên đơn. Chính sách này cho phép mở rộng cho các ứng dụng song song có kích thước lớn, bằng cách tránh các yêu cầu phải tạo một giấy chứng nhận duy nhất cho mỗi tiến trình, mà cho phép các tiến trình song song này dùng chung một tập các giấy chứng nhận.

4.1.3. Kiến trúc bảo mật cho tính toán lưới

Phần này sẽ đưa ra một kiến trúc bảo mật dựa trên các ngữ cảnh bảo mật và chính sách bảo mật được nêu ra ở trên. Các thành phần được đề cập trong kiến trúc bảo mật cho tính toán lưới bao gồm: các thực thể, giấy chứng nhận và các giao thức.

4.1.3.1. Các thành phần trong kiến trúc bảo mật

+ Thực thể: trong môi trường lưới bao gồm tất cả các chủ thể và đối tượng trong các tính toán. Chủ thể sẽ bao gồm các người dùng, các tiến trình thực hiện tính toán, bởi vì mỗi tính toán bao gồm nhiều tiến trình, mỗi tiến trình lại thay mặt cho một người dùng cụ thể. Còn đối

tượng được hình dung như môi trường rộng lớn của các tài nguyên có sẵn: máy tính, kho dữ liệu, mạng, thiết bị hiển thị, ... để phục vụ cho tính toán.

+ Giấy chứng nhận: được đưa ra trong kiến trúc như là một giải pháp cho yêu cầu thống nhất các định danh của các thực thể lưới, cung cấp cho cả người dùng và tài nguyên. Cơ chế này giúp giảm bớt thời gian chứng thực khi thực hiện tính toán, tạo điều kiện thuận lợi nhất cho người dùng. Có hai loại giấy chứng nhận:

Giấy chứng nhận người dùng: được cấp quyền để thay mặt người dùng trong một thời gian hạn chế.

Giấy chứng nhận người dùng hoạt động thay cho người dùng, nhờ đó nó có những thuận lợi như: không yêu cầu người dùng phải có mặt trong thời gian tính toán, thay vào đó giấy chứng nhận của người dùng phải sẵn có trong suốt thời gian tính toán. Hơn nữa, thời gian sống của giấy chứng nhận được điều khiển bởi người dùng, giúp cho người dùng hoàn toàn chủ động trong quá trình tính toán mà không cần có mặt ở đó.

Giấy chứng nhận tài nguyên: chứng nhận cho một tài nguyên là thành viên hợp lệ của lưới, các chính sách bảo mật riêng trong tài nguyên đó tương thích với các chính sách bảo mật chung của lưới, có thể ánh xạ chính sách bảo mật chung của lưới vào trong chính sách bảo mật cụ thể của tài nguyên. Ví dụ, trong chính sách bảo mật của Globus Toolkit 3 (GT3), giấy ủy nhiệm tài nguyên GRIM (ánh xạ thực thể tài nguyên lưới - Grid Resource Identity Mapper) chứng minh tài nguyên này là thành viên của lưới, tuân theo các chính sách bảo mật của lưới. Ngoài ra, kiến trúc GRIM cho phép cơ chế ủy quyền truy cập tài nguyên (e.g. host) thông qua một chương trình setuid, cho phép tài nguyên này nằm lâu dài trong một host cụ thể, tuân theo các chính sách bảo mật trong host đó.

+ Các giao thức: là các quy ước được đưa ra cho hoạt động bảo mật trong môi trường lưới. Trong kiến trúc này, sử dụng một số giao thức giúp cho các hoạt động trên toàn bộ lưới: người dùng có thể đăng nhập, định vị tài nguyên, định vị các tài nguyên khác trong tiến trình thực hiện.

4.1.3.2. Mô tả kiến trúc

Một số ký hiệu:

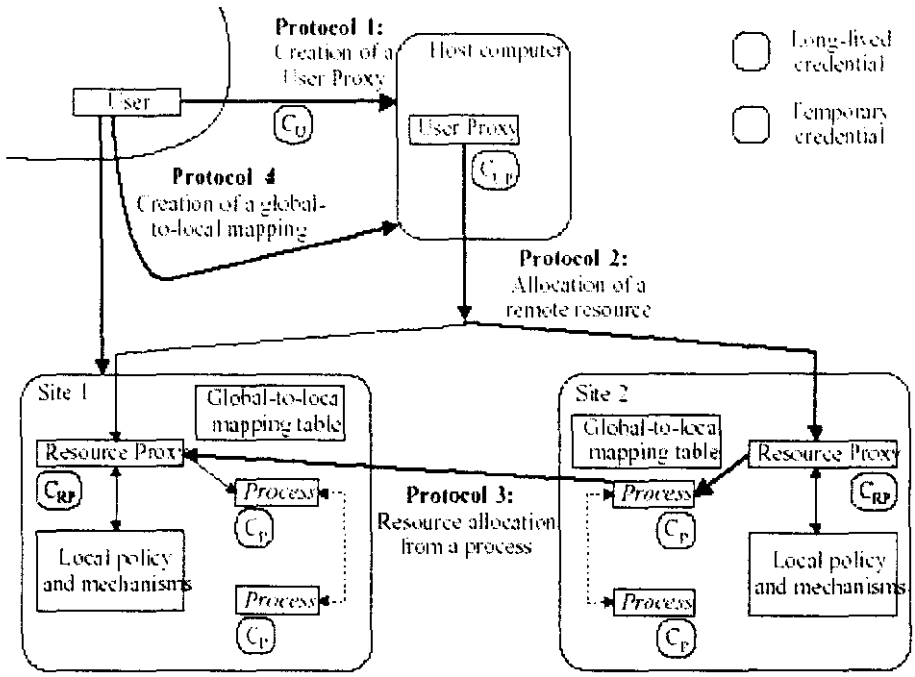
U, R, P: tương ứng là các người dùng, tài nguyên, và tiến trình.

UP, RP: giấy chứng nhận người dùng và giấy chứng nhận tài nguyên.

CX: Giấy ủy nhiệm của chủ thể X.

SigX(text): văn bản (text) được kí bởi chủ thể X.

Hình 1-3 mô tả kiến trúc bảo mật của lưới.



Hình 4-2: Mô hình kiến trúc bảo mật của tính toán lưới

Như trên hình vẽ, các hoạt động trong kiến trúc bao gồm

1. Người dùng thông qua giấy ủy nhiệm người dùng, đăng nhập hệ thống sử dụng giao thức 1

2. Định vị tài nguyên và khởi tạo các tiến trình qua giấy ủy nhiệm người dùng sử dụng giao thức 2.

3. Mỗi tiến trình có thể định vị các tài nguyên khác một cách trực tiếp sử dụng giao thức 3.

4. Khởi tạo bảng ánh xạ các định danh toàn cục vào trong các định danh cục bộ sử dụng giao thức 4.

Sau đây là tóm tắt các giao thức trên:

Giao thức 1: Đăng nhập hệ thống

1. Người dùng truy nhập vào máy tính mà ở đó giấy chứng nhận người dùng có thể được khởi tạo, sử dụng bất cứ dạng chứng thực nào được thực hiện trên máy tính.

2. Người dùng tạo ra giấy ủy nhiệm người dùng (CUP), sử dụng giấy chứng nhận người dùng UP để kí lên một bản ghi bao gồm: định danh người dùng, tên máy cục bộ, thời gian hợp lệ cho CUP và bất cứ thông tin nào được đòi hỏi bởi giao thức chứng thực sử dụng để cài đặt kiến trúc.

Cup = SigU (usr-id, host, start-time, end-time, auth-info)

Ở đây cần phân biệt giấy chứng nhận người dùng UP và giấy ủy nhiệm người dùng CUP. Giấy chứng nhận UP thay mặt cho người dùng cho các hoạt động trên lưới, tuy nhiên nó có một vài hạn chế: dễ bị tổn thương và không hạn chế thời gian hoạt động khi thay mặt người

dùng. Giải pháp được đưa ra là sử dụng giấy ủy nhiệm tạm thời CUP, do giấy chứng nhận UP kí lên với một khóa bí mật. Giấy ủy nhiệm CUP cũng có thể hoạt động thay mặt người dùng, nhưng có thời gian hoạt động hạn chế và một số ràng buộc bởi người dùng như: tên máy (nơi mà UP được phép hoạt động), tên các miền (nơi mà UP được phép khởi tạo tiến trình và sử dụng tài nguyên)

3. Một tiến trình giấy chứng nhận người dùng được khởi tạo cùng với CUP. Nó dựa trên chính sách bảo mật cục bộ để bảo vệ tính toàn vẹn của CUP.

Giao thức 2: Giao thức định vị tài nguyên

1. Các giấy chứng nhận UP và RP thẩm quyền lẫn nhau (xem khái niệm thẩm quyền đa phương) sử dụng giấy ủy nhiệm CUP và CRP. Một phần của tiến trình này, RP kiểm tra để đảm bảo CUP vẫn còn hợp lệ, chưa hết thời hạn.

2. UP chuyển các yêu cầu về định vị tài nguyên cho RP dưới dạng

SigUP(các yêu cầu định vị)

3. RP kiểm tra tính hợp lệ của UP đã yêu cầu định vị, bảo đảm các yêu cầu này đúng là do UP đã kí lên.

4. Nếu yêu cầu được tiếp nhận, RP tạo ra một bản ghi RESOURCE_CREDENTIAL chứa: tên người dùng, tài nguyên định vị cho người dùng đó, tên tài nguyên, ...

5. RP chuyển RESOURCE_CREDENTIAL một cách an toàn tới UP (thực hiện giống bước 1, UP chuyển yêu cầu định vị tới RP).

6. UP kiểm tra tính hợp lệ RESOURCE_CREDENTIAL, xác nhận và kí lên nó - Sigup(RESOURCE_CREDENTIAL), để tạo ra một giấy ủy nhiệm yêu cầu tài nguyên. Giấy ủy nhiệm này còn được gọi là giấy ủy nhiệm tiến trình CP.

7. UP chuyển CP một cách an toàn tới RP (thực hiện giống bước 1, UP chuyển yêu cầu định vị tới RP).

8. RP định vị tài nguyên, tiếp tục tiếp nhận các tiến trình CP mới do người dùng yêu cầu.

Giao thức 3: Giao thức định vị tài nguyên thông qua tiến trình người dùng

1. Tiến trình và UP của nó chứng thực lẫn nhau (xem chứng thực đa phương 1.2.2) thông qua CP và CUP.

2. Tiến trình phát ra một yêu cầu tới UP của nó, dưới dạng:

SigP {"định vị", các tham số yêu cầu định vị}

3. Nếu UP chấp nhận yêu cầu, nó sẽ phát ra một yêu cầu định vị tài nguyên tới RP cụ thể nào đó sử dụng giao thức 2, đã mô tả ở trên.

4. Kết quả định vị tài nguyên sẽ được UP kí lên, sau đó chuyển cho tiến trình yêu cầu.

Giao thức 4: Ánh xạ định danh toàn cục vào định danh cục bộ.

1a. UP chứng thực với RP

1b. UP phát ra một yêu cầu đã ký MAP-SUBJECT-UP dưới dạng các tham số: tên chủ thể tài nguyên, tên chủ thể toàn cục

2a. Người dùng đăng nhập vào tài nguyên sử dụng các phương thức chứng thực của tài nguyên và bắt đầu một tiến trình đăng ký ánh xạ.

2.b Tiến trình ánh xạ tài nguyên phát ra một yêu cầu MAP-SUBJECT-P tới RP, dưới dạng các tham số: tên chủ thể tài nguyên, tên chủ thể toàn cục.

1. RP đợi các yêu cầu MAP-SUBJECT-UP và MAP-SUBJECT-P với các tham số tương ứng.

2. RP bảo đảm rằng tiến trình đăng ký ánh xạ thuộc về chủ thể tài nguyên được xác định trong yêu cầu ánh xạ.

3. Nếu kiểm tra hợp lệ, RP sẽ thiết lập ánh xạ và gửi xác nhận tới tiến trình đăng ký ánh xạ và UP.

4. Nếu kiểm tra không hợp lệ trong khoảng thời gian MAP-TIMEOUT, RP sẽ loại bỏ yêu cầu và gửi xác nhận tới các thực thể đang chờ đợi.

5. Nếu xác nhận chưa được nhận trong khoảng thời gian MAP-TIMEOUT, yêu cầu coi như là không được chấp nhận.

Hoạt động chi tiết của các giao thức đều được thiết lập dựa trên thẩm quyền, kĩ thuật chữ kí điện tử và mã hóa. Tuy nhiên, kiến trúc không đề cập một kĩ thuật chữ kí điện tử riêng biệt nào, việc cài đặt cụ thể sử dụng giao diện lập trình ứng dụng chung GSS, do đó việc cài đặt và phát triển có thể sử dụng bất cứ công nghệ bảo mật riêng biệt nào.

Cơ sở hạ tầng lưới GSI, một thành phần quan trọng của bộ công cụ Globus Toolkit, với việc sử dụng giao diện lập trình ứng dụng dịch vụ bảo mật chung GSS-API, giao thức bảo mật tầng Socket SSL, ... là một cài đặt cụ thể của kiến trúc bảo mật trên, sẽ được trình bày ở phần tiếp theo.

4.2. Tổng quan về cơ sở hạ tầng bảo mật GSI

Cơ sở hạ tầng bảo mật lưới (GSI) là tập các giao thức, thư viện và công cụ, cho phép người dùng và các ứng dụng truy cập tài nguyên lưới một cách an toàn. Đây là thành phần quan trọng và cơ bản nhất trong các công cụ phát triển lưới cũng như các ứng dụng dựa trên lưới.

4.2.1. Các khái niệm cơ bản về an toàn bảo mật

Một trong các khái niệm nền tảng cơ bản trong lĩnh vực an toàn và bảo mật thông tin đó chính là mã mật (Cryptography). Cơ sở hạ tầng bảo mật trong tính toán lưới (GSI) được xây dựng hoàn toàn dựa trên các khái niệm cơ bản này. Phần này sẽ đưa ra những khái niệm lý thuyết cơ bản tổng quan về mã mật

Hệ thống mã mật được xem là bộ 5 {P, C, K, E, D}.

P Tập các bản rõ có thể có : một văn bản, thông báo, gồm nhiều câu được sản sinh từ văn phạm. P còn được gọi là không gian tin.

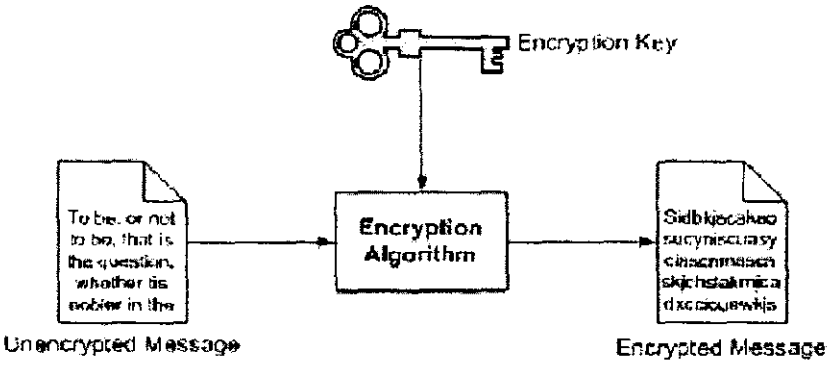
C Tập hữu hạn các bản mờ.

K Tập các khóa được xây dựng dựa trên cơ sở bộ chữ cái và các phép đại số trên các trường chứa bảng chữ cái đó.

E Tập các cơ chế mã có thể áp dụng đối với khóa đã chọn.

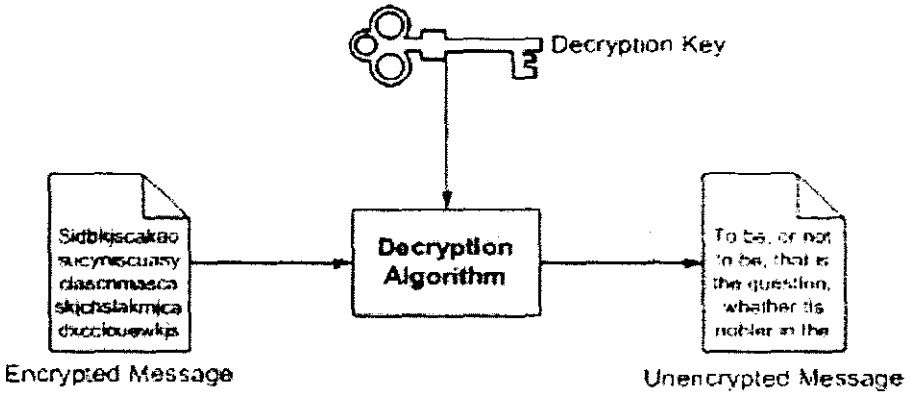
D Tập các cơ chế giải mã.

Người gửi mã hóa thông tin bằng khóa mã (Encryption Key) K và gửi cho người nhận:



Hình 4-3: Mã hóa bản tin sử dụng khóa

Người nhận sử dụng khóa giải (Decryption Key) để giải mã thông tin.

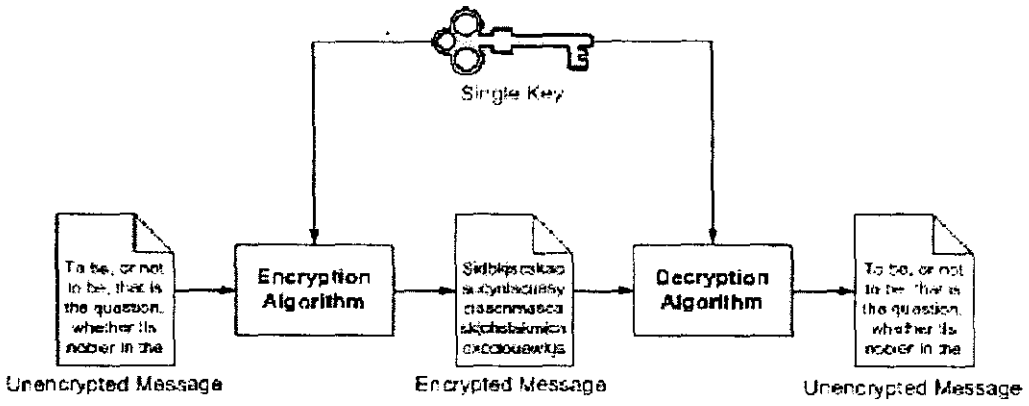


Hình 4-4: Giải mã thông điệp sử dụng khóa giải

Sau đây, ta sẽ xem xét hai phương pháp mã hóa thông dụng là mã đối xứng và mã công khai.

4.2.1.1. Mã hóa đối xứng (Symmetric encryption)

Mã đối xứng sử dụng cùng một khóa cho quá trình mã hóa và giải mã. Trong đó, hàm giải mã là hàm ngược của hàm mã hóa.



Hình 4-5: Mã đối xứng

Mặc dù các phương pháp này thông thường có tốc độ cao và dễ thực thi nhưng đồng thời lại có nhiều yếu điểm. Một nhược điểm chính đó là vì cả người gửi và người nhận đều sử dụng cùng một khóa mã do đó cần phải có sự trao đổi thông tin thống nhất khóa thông qua một kênh mật. Đây là một vấn đề lớn trong vấn đề an toàn và bảo mật.

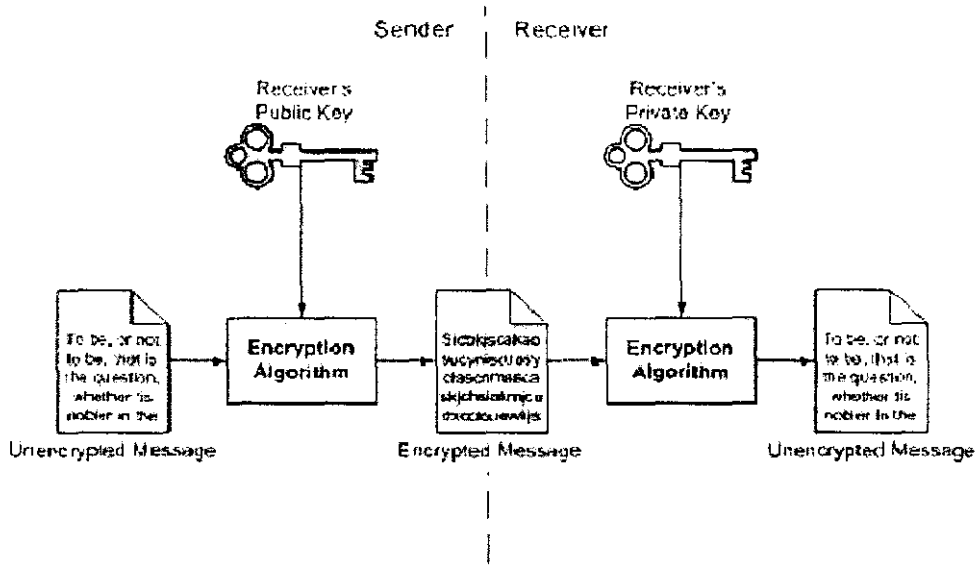
Chính vì vậy, các hệ thống bảo mật ngày nay thường sử dụng các thuật toán mã hóa không đối xứng (các khóa mã và khóa giải khác nhau). Mã công khai (Public-key) là một giải pháp được sử dụng phổ biến ngày nay.

4.2.1.2. Mã hóa công khai (Public Key - PK)

Nền tảng của mã hóa công khai là khóa mã và khóa giải là khác nhau. Các khóa này được xây dựng bằng cách chỉ ra một hàm bẫy sập một chiều (one-way). Đồng thời, cũng chỉ ra một cửa bẫy (Trap-door).

Trong hai khóa đó, một khóa được chọn làm khóa bí mật và khóa còn lại được chọn làm khóa công khai. Khóa bí mật (private key) chỉ có một người là chủ nhân của nó nắm giữ. Khóa công khai (public key) được công bố rộng rãi cho bất cứ ai muốn trao đổi thông tin mật với người sở hữu khóa.

Trong đó, khóa công khai được sử dụng để mã hóa thông tin và khóa bí mật được sử dụng để giải mã.



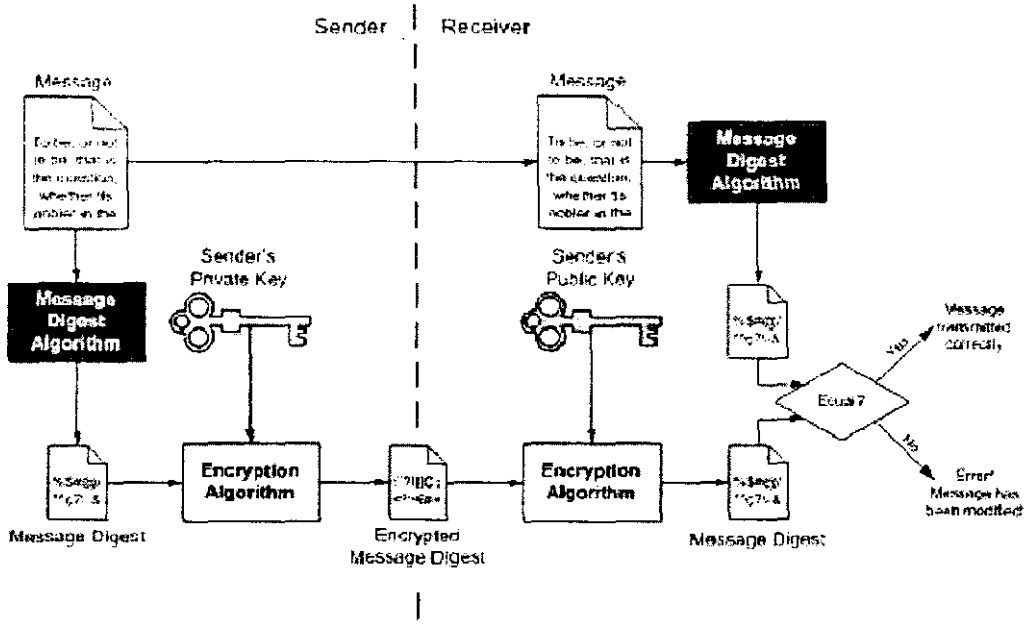
Hình 4-6: Mã công khai

So với hệ thống mã đối xứng, mã hóa công khai không sử dụng hai khóa khác nhau cho người gửi và người nhận. Đối với hệ thống mã này, độ phức tạp giải mã thường là hàm mũ trong khi độ phức tạp giải mã của hệ thống mã đối xứng thường chỉ là tuyến tính. Quá trình giao tiếp giữa hai đối tượng A và B có thể được mô tả như sau : B sinh ra một cặp khóa bí mật và công khai, khóa bí mật được cất giữ một cách an toàn và được bảo vệ bằng một mật mã còn khóa công khai được cung cấp rộng rãi. A có thể sử dụng khóa công khai (được phát hành bởi B) để mã hóa thông tin và gửi cho B. Lúc này, chỉ duy nhất B, người sở hữu khóa bí mật, có thể giải mã thông tin được giải mã bằng khóa công khai.

Ngoài ra, mã công khai còn đảm bảo được tính toàn vẹn của thông tin được mã hóa và còn được dùng trong các cơ chế xác thực. Tuy nhiên, một nhược điểm lớn duy nhất của hệ mã hóa công khai này là quá trình giải mã cũng như mã hóa mất nhiều thời gian.

4.2.1.3. Chữ ký số (Digital Signature - DS)

Thông thường, chữ ký được dùng để xác định tính hợp thức của những văn bản trong các quá trình giao dịch. Do đó, chữ ký là đặc trưng cho từng cá nhân. Chữ ký số (DS) là một hàm phụ thuộc vào thông tin mà nó ký. Trong giao dịch, chữ ký số được coi là một thông tin gắn liền với giao dịch.



Hình 4-7: Chữ ký số và mã hóa công khai

Quá trình sinh ra chữ ký số có thể được mô tả như sau:

Bản tin đầu tiên được băm sử dụng một hàm băm thích hợp để đảm bảo tính toàn vẹn của thông tin ban đầu gọi là thông điệp băm (digest message). Hàm băm có chức năng biến đổi các xâu vào có độ dài thay đổi thành những xâu có độ dài cố định và ngắn hơn các xâu vào rất nhiều.

Tiếp đó, bản tin đã được băm này được mã hóa sử dụng khóa bí mật của người gửi. Kết quả ta được một chữ ký số và chữ ký số này được gửi kèm với bản tin tới người nhận.

Người nhận sử dụng khóa công khai của người gửi giải mã chữ ký số để thu được thông điệp băm của người gửi.

Người nhận sử dụng thuật toán băm tương tự thuật toán băm của người gửi để băm bản tin nhận được thành một thông điệp băm.

So sánh thông điệp băm vừa được băm với thông điệp băm của người gửi. Nếu chúng không hoàn toàn giống nhau thì nội dung của bản tin nhận được đã bị thay đổi so với bản tin ban đầu của người gửi. Đồng thời, ta cũng có thể khẳng định đây là chữ ký số của người gửi bởi vì chữ ký số đã được mã hóa bằng khóa bí mật của người gửi và do đó, chỉ duy nhất khóa công khai của người gửi mới có thể giải mã được chữ ký số.

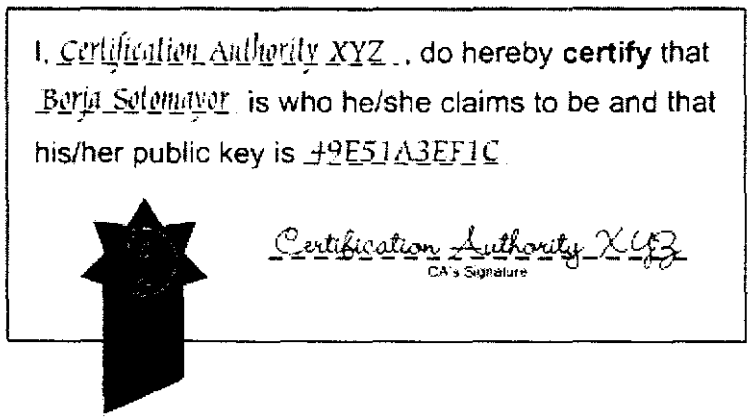
Việc kết hợp mã hóa công khai với chữ ký số cho phép ta xác định được sự toàn vẹn của bản tin. Quá trình mô tả như trên chỉ tập trung vào xác định tính toàn vẹn của bản tin bởi vì bản tin khi gửi đi không được mã hóa. Điều này phù hợp với thực tế có những thông tin chúng ta không cần che giấu nội dung mà chỉ cần bảo vệ sự toàn vẹn của nội dung (đảm

bảo nội dung không bị thay đổi). Khi cần bảo mật nội dung bản tin, ta chỉ việc mã hóa nội dung của bản tin.

Tuy nhiên, với quá trình như trên không có gì để đảm bảo rằng khóa công khai có đích thực là của người gửi hay không, có thể có một đối tượng khác giả danh, đánh tráo khoá công khai của người gửi. Chính vì vậy, cần có một cơ chế để đảm bảo xác nhận được khóa công khai mà ta đang sử dụng đích thực là của người mà ta cần giao tiếp. Giấy chứng nhận (Digital Certificates) cho phép ta giải quyết vấn đề này.

4.2.1.4. Giấy chứng nhận và Nhà chứng nhận thẩm quyền

Một giấy chứng nhận (Certificate) là một văn bản điện tử xác nhận một khoá công khai được sở hữu bởi một người cụ thể. Một giấy chứng nhận được cấp bởi một cơ quan có thẩm quyền gọi là nhà chứng nhận thẩm quyền (Certificate Authority - CA). Những người tham gia giao dịch cần có một kênh liên lạc bí mật với Nhà chứng nhận thẩm quyền.



Hình 4-8: Giấy chứng nhận

Một ví dụ về giao dịch điện tử giữa hai chủ thể A và B, thông qua một nhà thẩm quyền S như sau:

Bước 1: Xin các giấy chứng nhận

A trước khi giao dịch gửi khoá công khai cho S và S sẽ cấp một giấy chứng nhận CA. Thông tin từ A có thể gồm M = [ZA, Tên, Chu kỳ sử dụng, ...] trong đó có ZA là khoá công khai của A. Giấy chứng nhận CA được S mã hoá sử dụng khoá bí mật của mình EZs(M)). Tương tự B cũng sẽ được S cấp cho một giấy chứng nhận CB.

Bước 2: Giao dịch điện tử

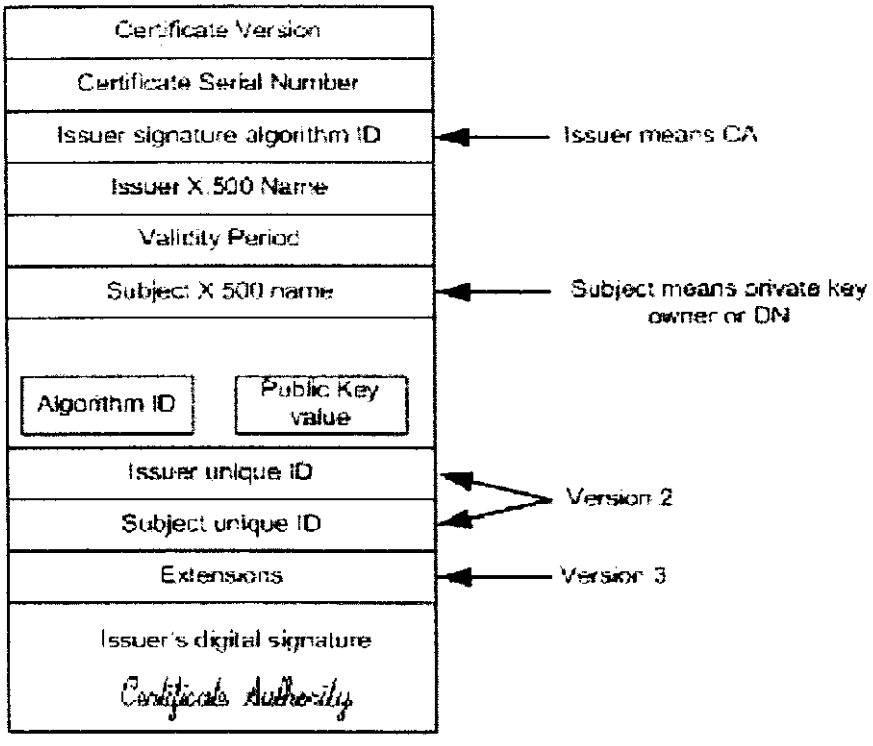
A tìm giấy chứng nhận của B CB, kiểm định chữ ký số của S đã kí lên CB. Nếu giấy chứng nhận CB đúng do S phát hành, A tách khoá công khai của B để mã hoá thông tin của mình và gửi cho B.

Với việc sử dụng giấy chứng nhận, quá trình chứng thực có sự liên quan của ba bên A, B, S. Quá trình kiểm chứng là rất nghiêm ngặt, S phải chứng minh là người phát hành các giấy

chứng nhận, các thông tin mà A gửi cho B cũng bảo đảm bí mật, bởi vì nó đã được mã hóa bằng khóa công khai của B, chỉ có B mới có khóa bí mật để giải mã thông tin từ A.

- Chuẩn X.509

Sau đây là khuyến nghị về định dạng của giấy chứng nhận theo chuẩn X.509. Một giấy chứng nhận theo định dạng X.509 là một văn bản chứa các thông tin theo các cú pháp đã định.



Hình 4-9: Giấy chứng nhận theo cơ chế chứng thực X.509

Subject: đây là tên của đối tượng xin cấp. Nó được mã hoá theo định dạng tên định danh (Distinguished Name).

Subject's public key: bao gồm các thông tin về khoá và thuật toán sử dụng để sinh ra khoá công khai đó.

Issuer's Subject: tên định danh của CA.

Digital signature: chữ ký điện tử sinh ra bởi khoá bí mật của CA. Chữ ký này có thể được kiểm định bằng khoá công khai của CA.

Tên định danh (Distinguished Name - DN): là tên gồm các cặp giá trị cách nhau bằng dấu phẩy. Ví dụ: O="University of Technology, OU=Faculty of Informatic, CN=HPCC". Một tên định danh có nhiều trường, trong đó, có một số trường thông dụng là:

O: tên tổ chức (Organization).

OU: tên của đơn vị trong tổ chức đó (Organization Unit).

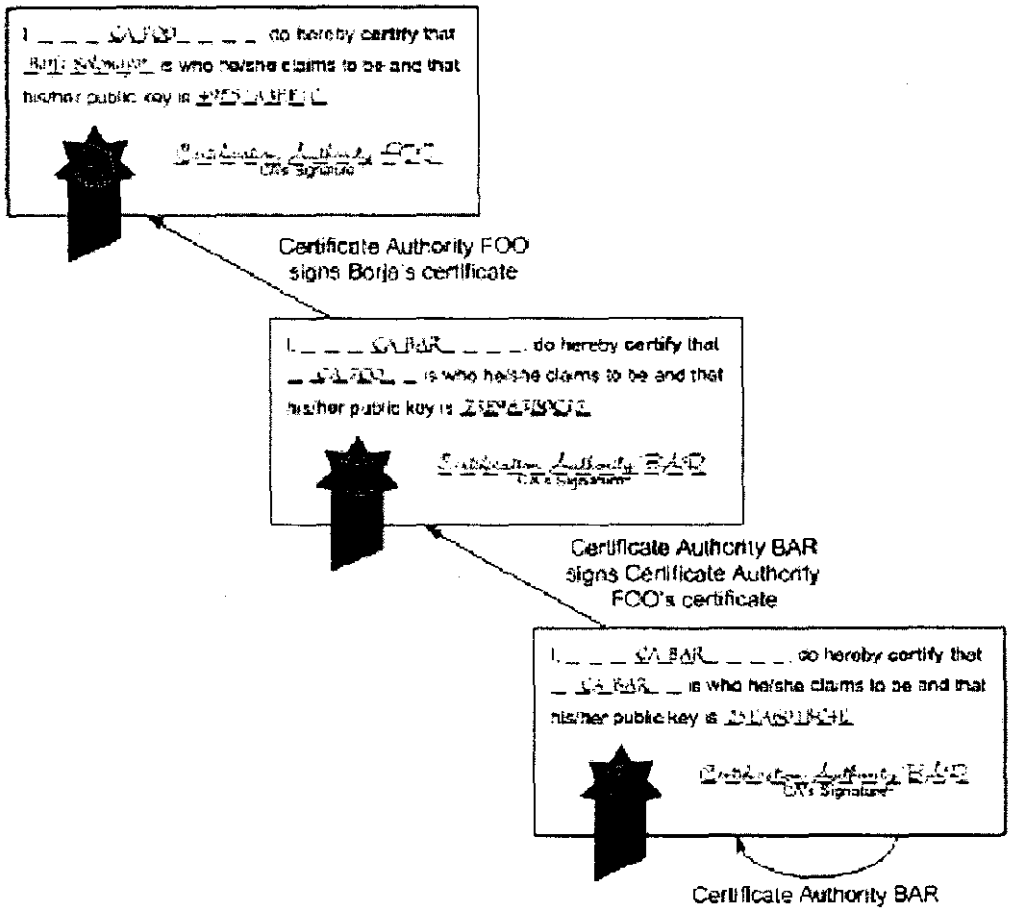
CN: tên đối tượng, thông thường là tên của người dùng.

C: đất nước (Country).

Tên định danh cho phép ta xác định được định danh duy nhất của một đối tượng trong tổ chức.

- Cấu trúc phân cấp Nhà chứng nhận thẩm quyền (CA)

Các đối tượng tham gia giao tiếp thì mỗi đối tượng có thể tin tưởng các Nhà chứng nhận thẩm quyền khác nhau. Trong trường hợp này, cấu trúc phân cấp Nhà chứng nhận thẩm quyền sẽ cho phép các bên tham gia tin tưởng các Nhà chứng nhận thẩm quyền khác nhau vẫn có thể thiết lập các mối quan hệ tin cậy trong giao tiếp.



Hình 4-10: Cấu trúc phân cấp CA

4.2.2. Cơ sở hạ tầng bảo mật lưới GSI

An toàn bảo mật là một trong những nền tảng quan trọng nhất trong hệ thống lưới. Hạ tầng bảo mật GSI được đưa ra để giải quyết những vấn đề bảo mật còn tồn tại trong tính toán lưới, mà nền tảng chính là những kiến thức cơ sở về mã mật và bảo mật mà nêu ra ở phần trên. Trong phần này đưa ra một số đặc điểm của GSI, các cài đặt ứng dụng của nó.

4.2.2.1. Cơ sở hạ tầng khóa công khai (Public Key Infracstructure - PKI)

GSI được xây dựng dựa trên cơ sở hạ tầng khóa công khai PKI. PKI là tập các thực thể (người dùng và tài nguyên), được phân biệt bởi tên duy nhất gọi là tên định danh (Distinguished Name – DN). Chứng thực trong GSI nghĩa là cho mỗi thực thể người dùng hoặc tài nguyên một tên định danh duy nhất.

Để mỗi một thực thể có một định danh duy nhất, GSI đưa ra khái niệm giấy ủy nhiệm lưới (Grid credentials), là một cặp gồm giấy chứng nhận (certificate) và một khóa mã hóa còn gọi là khóa bí mật (private key). Giấy chứng nhận, đơn giản chỉ là kết nối giữa tên định danh của thực thể đối với khóa bí mật của họ. Sự kết nối này được thực hiện bởi chữ kí điện tử (digital signature) bởi một bên thứ ba đáng tin cậy gọi là nhà thẩm quyền (Certificate Authorithy - CA). Điều này cho phép các thực thể được xác thực (sử dụng giấy ủy nhiệm) bằng cách đưa ra giấy chứng nhận của họ, đồng thời cũng chứng minh sự sở hữu của họ đối với khóa bí mật.

Một điều quan trọng trong môi trường PKI mỗi thực thể phải trao quyền sở hữu khóa bí mật của mình để bảo đảm sự toàn vẹn của hệ thống. Để bảo đảm khóa bí mật không bị đánh cắp, có thể sử dụng một số phương pháp:

- Lưu trữ khóa trong một file có quyền truy nhập hạn chế
- Lưu trữ khóa trong một file đã mã hóa mà khóa mã chỉ được biết bởi người sử hữu nó
- Lưu trữ khóa bí mật bằng các thiết bị phần cứng có mật khẩu (smart card). Giải pháp phần cứng cho ta tính bảo mật cao nhất, nhưng nó lại ít được sử dụng bởi thiếu sự phát triển của phần cứng.
- Sử dụng giấy ủy nhiệm trong một khoảng thời gian sống nhất định (life time) thì nó sẽ không còn hợp lệ nữa. Điều này đòi hỏi thường xuyên phải có tập mới các giấy ủy nhiệm, bảo vệ khóa bí mật bằng cách hạn chế sự lộ diện của nó. Thông thường thời gian sống là một năm, độ dài chính xác thường được xác định bởi các nhà thẩm quyền phát hành giấy chứng nhận này.

4.2.2.2. Giao thức bảo mật tầng Socket SSL

GSI sử dụng giao thức bảo mật tầng Socket (SSL) để thực thi các chứng thực (nhận dạng các thực thể truyền thông khác), bảo đảm sự toàn vẹn thông điệp (bảo đảm không có ai chỉnh sửa thông điệp giữa hai thực thể), bảo vệ các thông điệp (bảo đảm không có ai đọc được thông điệp giữa hai thực thể). SSL là một công cụ phần mềm chuẩn trong các trình duyệt Web, máy chủ Web và các phần mềm khác. Nó sử dụng giấy ủy nhiệm PKI cho chứng thực và được sử dụng trong GSI không có thay đổi nào.

4.2.2.3. Giấy ủy nhiệm (Proxy Credential)

Trong môi trường lưới, người sử dụng cần được chứng thực nhiều lần trong khoảng thời gian tương đối ngắn, ví dụ với nhiều tài nguyên lưới cùng cộng tác với nhau. Đòi hỏi người

sử dụng phải gõ mật khẩu nhiều lần trong chứng thực đa phương (multiple autholization) là một điều không thuận tiện, đồng thời nó cũng bất lợi cho vấn đề bảo mật, bởi mỗi lần đó, khóa bí mật sẽ được giải mã, rất dễ bị kẻ địch đánh cắp. Một cách khác là sử dụng phần mềm, không chỉ nhắc người sử dụng gõ mật khẩu một lần, mà còn lưu giữ mật khẩu hay khóa bí mật chưa giải mã cho nhiều mục đích sử dụng khác. Tuy nó rất thuận tiện cho người dùng, nó lại bất lợi trên quan điểm về bảo mật, khi mà nó để lộ khóa bí mật trong một khoảng thời gian dài.

GSI giải quyết vấn đề này với khái niệm giấy ủy nhiệm (proxy predential). Mỗi giấy ủy quyền sẽ hoạt động thay mặt người dùng trong một khoảng thời gian ủy quyền ngắn hạn. Nói cách khác, việc sử dụng giấy ủy nhiệm ngắn hạn thay thế cho các giấy chứng nhận dài hạn khi chứng thực người dùng.

Giấy ủy nhiệm có giấy chứng nhận và khóa bí mật riêng của nó, được tạo ra bằng cách kí lên giấy chứng nhận dài hạn của người dùng. Giấy ủy nhiệm, theo một cách khác, là sự liên kết ngắn hạn giữa tên định danh của người dùng với một khóa bí mật khác. Giấy chứng nhận thường được lưu trữ không mã hóa trong hệ thống file địa phương, thường được bảo vệ bởi quyền truy cập file trong hệ thống, có thể được sử dụng nhiều lần mà không có sự bất tiện nào. Còn giấy ủy nhiệm dễ bị tổn thương, nó có thời gian sống ngắn hạn hơn nhiều so với các giấy chứng nhận dài hạn của người dùng, thông thường là vài giờ cho tới một ngày.

4.2.2.4. Sự ủy quyền

Điều này là rất quan trọng trong các ứng dụng phân tán, các ứng dụng của người dùng có thể thay mặt họ trong môi trường lưới. Một ví dụ, trong một tính toán phức tạp và kéo dài liên quan tới nhiều tiến trình khác nhau, người dùng không phải lúc nào cũng có mặt để chứng thực cho mỗi tiến trình. GSI giải quyết vấn đề này bằng cách cho phép người dùng ủy quyền giấy ủy nhiệm của mình cho giao dịch các máy từ xa.

Sự ủy quyền cũng tương tự như việc tạo ra các giấy ủy nhiệm đã mô tả ở 2.2.3, khi một tập các giấy chứng nhận dài hạn sẽ được dùng để tạo ra tập các giấy ủy nhiệm mới, có thời gian sống ngắn hơn. Sự khác nhau là việc tạo ra các giấy ủy nhiệm xảy ra trong các phiên kết nối đòi hỏi chứng thực GSI, khi các tiến trình từ xa đòi hỏi giấy ủy nhiệm của người dùng cho chứng thực. Một điều đáng chú ý nữa là sự ủy quyền có thể là một chuỗi, một người có thể ủy quyền cho một host A, sau đó tiến trình sử dụng trên host A có thể ủy quyền cho host B và cứ tiếp tục như vậy.

4.2.2.5. Chứng thực

GSI hỗ trợ cơ chế cho phép chuyển các tên định danh GSI của người dùng (ví dụ tên định danh từ giấy chứng nhận của người dùng) vào trong các định danh địa phương (tài khoản của một người dùng Unix cục bộ).

Việc chứng thực các định danh GSI sẽ chuyển về chứng thực các định danh địa phương, cùng với đó, các chính sách đưa ra cũng nằm trong phạm vi cục bộ như: quyền truy nhập file, dung lượng đĩa, tốc độ CPU, ...

4.2.2.6. Ứng dụng của GSI

GSI cho phép người dùng và các ứng dụng lưới truy nhập vào các tài nguyên một cách an toàn. Một số khả năng bảo mật được GSI tập trung hỗ trợ: cơ chế uỷ nhiệm và đăng nhập một lần, thẩm quyền và chứng thực đa phương, các giấy ủy nhiệm thay mặc người dùng trong thời gian ngắn hạn, ... GSI cũng là thành phần thiết yếu cho một số công cụ như grid-proxy-init để tạo ra giấy ủy quyền từ các giấy ủy nhiệm ngắn hạn, các dịch vụ truyền file GridFTP và máy chủ thông tin LDAP, đệ trình các ứng dụng từ xa Globus Toolkit Gram hay Secure Shell (SSH) để kết nối tới các máy từ xa, . Ngoài ra, tập các thư viện bảo mật GSI trong Java CogKit tạo điều kiện thuận lợi cho phát triển các ứng dụng lưới có hỗ trợ bảo mật. Hiện tại, một GSI được triển khai trong một một số mạng lưới: lưới thông tin năng lượng của NASA - Nasa Information Power Grid, NSF PACI Grid, ..., và đặc biệt GSI là nền tảng bảo mật thiết yếu của bộ công cụ phát triển lưới Globus Toolkit, sẽ trình bày trong phần tiếp theo.

4.3. Cơ sở hạ tầng bảo mật trong GT3

Trong các phần trên đã giới thiệu tổng quan về tính toán lưới, các yêu cầu bảo mật trong đó và cơ sở hạ tầng bảo mật lưới GSI. Globus Toolkit version 3 (GT3) đã cài đặt cơ sở hạ tầng GSI trong mô hình bảo mật của mình, tích hợp GSI với OGSA, cho phép tận dụng được các kỹ thuật và cơ chế bảo mật sẵn có của các dịch vụ Web (Web Service). GT3 sử dụng các cơ chế bảo mật sau của OGSA và Web services.

+ Sắp đặt các chức năng bảo mật tương tự như các dịch vụ khác của OGSA. Do đó, chúng có thể được phân bố và sử dụng bất cứ khi nào ứng dụng yêu cầu.
+ Thiết lập các môi trường trình chủ (hosting environments) an toàn.
+ Cung cấp các chính sách bảo mật để sao cho các client có thể dễ dàng tham gia Grid.
+ Xác định các tiêu chuẩn trao đổi các mã bảo mật để có thể cung cấp khả năng liên tác.

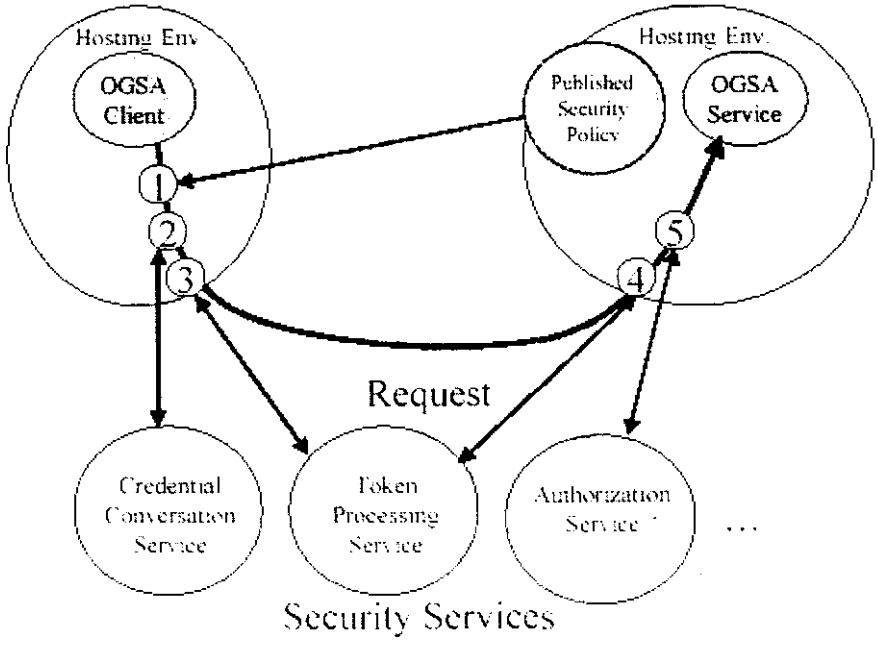
4.3.1. Một số dịch vụ bảo mật trong GT3

Cơ chế bảo mật trong tính toán lưới yêu cầu các ứng dụng và các dịch vụ phải có khả năng hỗ trợ rất nhiều chức năng bảo mật khác nhau như chứng thực, chứng nhận, ủy quyền, ủy nhiệm,... Các dịch vụ ứng dụng trong tính toán lưới cũng phải kèm theo các cơ chế bảo mật linh hoạt để dễ dàng thích nghi với các thay đổi về môi trường. Một số dịch vụ về an toàn bảo mật được đề nghị trong Global Grid Forum bao gồm:

- Credential processing service: là dịch vụ đảm nhận việc xử lý và kiểm tra các yêu cầu về chứng thực.
- Authorization service: dịch vụ này xác định quyền của các hoạt động dựa trên việc áp dụng các quyền bảo mật đối với từng đối tượng yêu cầu hành động đó và cụ thể yêu cầu đó.
- Credential Conversion service: là dịch vụ trung gian chuyển đổi giữa các cơ chế ủy nhiệm local và các VOs.
- Identity Mapping service: là dịch vụ mà ánh xạ tài khoản user trong một domain tới một tài khoản đối với một domain khác. (ví dụ, một tài khoản người dùng được chứng nhận theo chuẩn X.509 ánh xạ tới một tài khoản khác được chứng nhận theo chuẩn Kerberos).
- Audit: dịch vụ log lại các sự kiện.
- Môi trường trình chủ

Các Grid services cũng tương tự như các dịch vụ Web khác, được xây dựng trên các môi trường trình chủ như J2EE hay .NET. Những trình chủ này cung cấp mức độ cao của bảo mật và hầu hết các chức năng bảo mật đều được đặt trên trình chủ do đó sẽ làm cho việc phát triển đơn giản hơn và cho phép các chức năng bảo mật được nâng cấp hoàn toàn độc lập với các ứng dụng.

Hình vẽ sau đưa ra một ví dụ đã được đơn giản hóa của cách thức hoạt động của mô hình GT3 OGSA. Một yêu cầu xuất phát từ một OGSA client tới một OGSA service. Cả client và service đều được chứa trong các trình chủ (để đơn giản một số các chi tiết về quá trình bảo mật như auditing (kiểm định), quá trình chứng nhận client...được bỏ qua. Các chức năng này cũng sẽ được thực hiện một cách tương tự.



Hình 4-11: Ví dụ về kiến trúc bảo mật trong GT3

Client đầu tiên sẽ tạo một yêu cầu tới một dịch vụ OGSA và truyền yêu cầu đó tới trình chủ xử lý. Các bước sau dùng để xử lý yêu cầu từ client.

1. Trình chủ chứa client kiểm tra các chính sách bảo mật của dịch vụ được yêu cầu để xác định những cơ chế và ủy nhiệm nào cần phải tuân theo để có thể đệ trình yêu cầu.

2. Nếu trình chứa của client xác định được rằng các cơ chế ủy nhiệm cần thiết chưa có thì yêu cầu một dịch vụ Credential Conversion Service để chuyển các giấy ủy nhiệm đã có thành dạng cần thiết. (ví dụ CAS để chuyển các giấy ủy nhiệm cá nhân thành giấy ủy nhiệm của VO, và KCA để chuyển giữa chuẩn Kerberos và PKI – Public Key Infrastructure).

3. Trình chứa client sử dụng Token Processing và Validate Service (dịch vụ xử lý và kiểm tra) để xử lý các yêu cầu chứng thực khi giao tiếp với dịch vụ yêu cầu. Chính dịch vụ này đã làm cho ứng dụng và trình chứa không phải quan tâm nhiều đến các chi tiết cụ thể.

Về phía server, trình chứa cũng sử dụng một dịch vụ xử lý việc chứng thực của client. Sau quá trình chứng thực client, trình chứa của dịch vụ được yêu cầu chuyển các chi tiết về yêu cầu và thông tin về client tới một dịch vụ chứng thực (authorization service) ví dụ PERMIS hay Akenti để quyết định các quyền hạn thực thi.

Nếu tất cả các bước trên đều thành công thì trình chứa của dịch vụ được yêu cầu sẽ chuyển các yêu cầu đã được cho phép tới dịch vụ được yêu cầu để xử lý. Ứng dụng biết rằng trình chứa đã đảm nhận toàn bộ việc bảo mật do đó chỉ tập trung vào việc xử lý yêu cầu.

4.3.2. Các cải tiến về bảo mật trong GT3

Về cơ bản, các chức năng bảo mật trình bày trong chương 2 đã được cài đăth trong GT2. Dưới đây xin trình bày một số bổ sung về bảo mật trong GT3 so với GT2.

+ Giấy ủy nhiệm dịch vụ và người dùng: GT3 chứng thực dịch vụ và người dùng thông qua giấy ủy nhiệm theo chuẩn định dạng X509, một cặp khóa bí mật và tên định danh sẽ được sử dụng khi chứng thực. Đáng chú ý là chuẩn định dạng X509 của GT3 đã thay đổi so với GT2, tuy nhiên các giấy ủy nhiệm của GT2 vẫn hợp lệ trong môi trường lưới của GT3. Ngoài ra, tài nguyên cũng được chứng thực thông qua một giấy chứng nhận gọi là GRIM proxy, tuy nhiên không phải là chuẩn X509 (non-standard X509).

+ Chứng thực tài nguyên - Gridmap-file: GT3 sử dụng một file tên là grid-màpile để ánh xạ định danh người dùng lưới (tên định danh từ giấy chứng nhận X509 của người dùng) với một định danh địa phương có trong tài khoản Linux.

+ Giao diện ứng dụng lập trình: GT3 tiếp tục sử dụng giao diện lập trình bảo mật chung Generic Security Service API (GSSAPI) đã có trong GT2 , với một số mở rộng để cải tiến cơ chế bảo mật.

+ Giao thức Web Service: GT3 phát triển dựa trên công nghệ Web Service và OGSA (Open Grid Services Architecture), với chuẩn mở truyền thông điệp SOAP. Vì vậy, cơ chế bảo mật

được cài đặt thông qua mức bảo vệ thông điệp, bao gồm các chuẩn WS-Security, XML-Encryption và XML-Signature.

+ Cải tiến mô hình chứng thực tài nguyên: Trong các dịch vụ mạng, khi tiếp nhận và xử lý các yêu cầu, thường hay bị tấn công bằng các phương thức như lỗi logic, tràn bộ đệm, ... GT3 loại bỏ tất cả các đặc quyền từ những dịch vụ trên nhằm giảm thiểu các ảnh hưởng khi mất quyền điều khiển. Các dịch vụ lưới trong GT3 cho phép kết nối qua mạng không qua bất cứ đặc quyền địa phương đặc biệt nào, thay vào đó, chúng sử dụng hai chương trình setuid đặc biệt để thực hiện các hoạt động yêu cầu đặc quyền. Điều này khiến cho kẻ tấn công không có bất cứ đặc quyền nào đối với hệ thống khi dùng dịch vụ lưới, đồng thời nó cũng giảm bớt phần mã nguồn về thiết lập cơ chế đặc quyền.

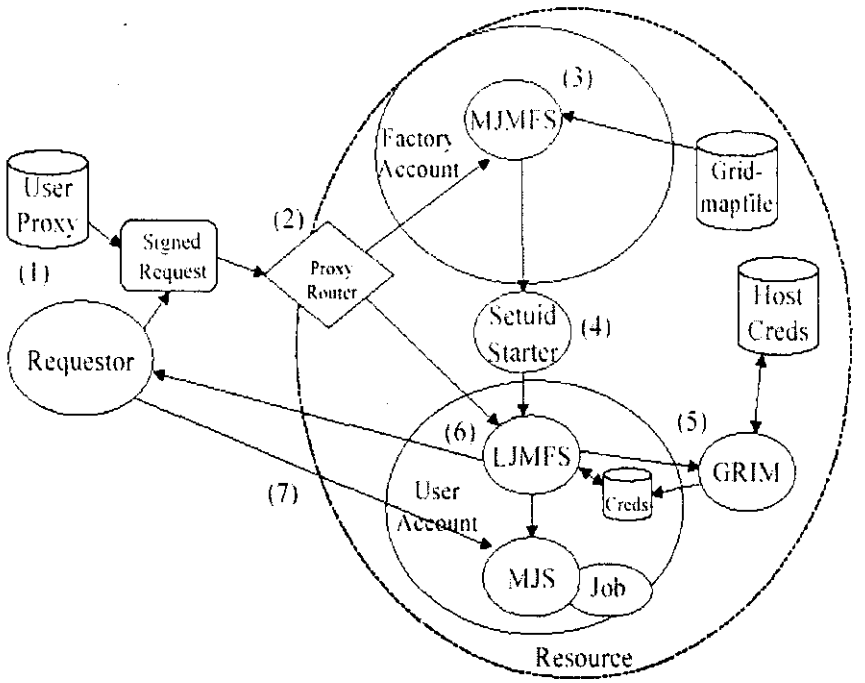
+ Loại bỏ các dịch vụ mạng từ mô hình chứng thực: Một số dịch vụ mạng rất dễ bị tổn thương khi có kết nối mạng. GT3 cung cấp cơ chế ủy nhiệm, cho phép các dịch vụ có thể triệu gọi trực tiếp nhau thông qua giấy ủy nhiệm của người dùng, hay nói cách khác, người dùng trực tiếp chứng thực tới dịch vụ.

4.3.3. Ví dụ minh họa - Cài đặt bảo mật trong GT3 GRAM

GSI3 thực thi cơ chế bảo mật với GT3 GRAM. GRAM là dịch vụ của GT cho phép client có thể khởi tạo, quản lý, theo dõi một cách an toàn các nhiệm vụ tính toán trên các máy ở xa. Việc tiến hành bảo mật đối với GRAM là một trong những công việc phức tạp nhất bởi vì nó liên quan nhiều đến cả cơ chế bảo mật địa phương và các client từ xa.

Để có thể thực hiện công việc nhờ GRAM, một client phải mô tả công việc, chỉ rõ những chi tiết về thư mục thực thi, nơi lưu trữ các đầu ra và đầu vào...Những mô tả này được gửi tới tài nguyên và kết quả là một thực thể của dịch vụ quản lý công việc (Managed Job Service - MJS). Một MJS là một dịch vụ của Grid cho phép khởi tạo công việc, điều khiển, theo dõi công việc.

MJS được tạo bởi một MJS factory service. Về lý thuyết thì ta sẽ có tương ứng với mỗi user account một MJS factory service, tuy nhiên trong thực tế để tránh lãng phí, GT3 thực thi một service là Master Managed Job Factory Service (MMJFS). Mỗi MMJFS chạy trên mỗi resource và triệu gọi Local Managed Job Factory Services (LMJFS) cho mỗi user account khi cần thiết. Một service là Proxy Router dẫn những yêu cầu từ user tới LMJFS nếu có hoặc MMJFS nếu LMJFS của user đó không tồn tại. Với MJS factory thì có thể có một hay nhiều MJS instance chạy cùng một lúc trên trình chứa.



Hình 4-12: Cơ chế thực hiện của GRAM 3

1) Requestor tạo một bản mô tả công việc và ký lên nó với một giấy ủy nhiệm thích hợp. Yêu cầu này sau đó được gửi tới target resource.

2) Proxy Router service nhận yêu cầu và gửi tới LMJFS nếu có (nhảy tới bước 6) hoặc gửi tới MMJFS (thực hiện tiếp bước 3).

3) MMJFS xác nhận chữ ký trên yêu cầu sau đó xác định local account mà job sẽ chạy dựa trên account đó sử dụng grid-mapfile, một file nằm trên máy cục bộ chứa đựng những thông tin ánh xạ từ GSI vào tài khoản cục bộ.

4) MMJFS triệu gọi quá trình Setuid Starter để khởi tạo LMJFS. Setuid Starter là một chương trình có đặc quyền (ví dụ như setuid-root) mà chức năng duy nhất của nó là khởi tạo một LMJFS đã được cấu hình sẵn cho một user.

5) Khi LMJFS đã được tạo, nó cần nhận được giấy ủy nhiệm và phải tự đăng ký với Proxy Router để nó thể nhận được các yêu cầu khác trong tương lai. LMJFS triệu gọi Grid Resource Identity Mapper (GRIM) để nhận được giấy ủy nhiệm. GRIM là một chương trình có đặc quyền truy cập và sinh ra các giấy ủy quyền cho LMJFS. Giấy ủy quyền này chứa trong nó user's Grid identity, local account name ... để giúp cho requestor có thể đảm bảo đây đúng là LMJFS cần thiết.

6) LMJFS nhận được yêu cầu công việc, LMJFS kiểm định chữ ký trên yêu cầu để đảm bảo là nó không bị giả mạo và kiểm tra xem requestor có được quyền truy cập local user account mà LMJFS đang chạy. LMJFS khởi tạo một MJS và trả lại tham chiếu dịch vụ đến user.

7) Requestor kết nối tới MJS để bắt đầu công việc. Requestor và MJS kiểm chứng lẫn nhau trước khi tiến hành quá trình giao dịch. MJS kiểm định xem requestor có đủ thẩm quyền để thực thi trong local account hay không còn requestor kiểm định xem MJS có giấy ủy nhiệm (GRIM credential) hợp lý từ máy chủ hay không, điều này cho phép client không chỉ xác định được là MJS chạy trên đúng máy chủ mà còn chạy trên đúng account.

4.4. Lập trình bảo mật trong môi trường lưới

4.4.1. Bảo mật cho các dịch vụ lưới

Các dịch vụ lưới được bảo mật chủ yếu dựa trên cơ chế truyền thông điệp của GT3. Bảo mật mức truyền thông điệp trong Globus Toolkit V3 (GT3) dựa trên các chuẩn bảo mật Web Service và XML (WS-Security, XML Encryption và XML Signature). GT3 cung cấp hai cơ chế chứng thực mức truyền thông điệp khác nhau:: an toàn các hội thoại trong môi trường lưới (GSI Secure Conversation) và an toàn các thông điệp trong môi trường lưới (GSI Secure Message).

+ Với phương thức an toàn các hội thoại, một ngữ cảnh bảo mật được thiết lập trước tiên giữa client và service. Ngữ cảnh này sau đó được dùng để kí, xác nhận, mã hóa và giải mã lên các thông điệp. Một ví dụ, dịch vụ tính toán bảo mật MathService thiết lập mức bảo vệ bí mật (privacy) cho phương thức cộng Add(). Khi đó, client muốn sử dụng khả năng cộng Add() của dịch vụ, thì phải thiết lập mức bảo vệ là mã hóa (encryption). Các thiết lập của cả client và service đều được ghi vào một ngữ cảnh bảo mật, và ngữ cảnh này sẽ trực tiếp mã hóa thông điệp giữa client và service khi có có yêu cầu tính toán từ phía client.

+ Với phương thức bảo mật thông điệp, một thông điệp được kí và mã hóa theo chuẩn giấy ủy nhiệm X509. Mã hóa trong trường hợp này được thực hiện theo hai bước.

Một khóa không đối xứng có kích cỡ là 128 bits, được sinh ra dùng kỹ thuật mã hóa không đối xứng AES, để mã hóa các thông điệp.

Chính khóa mã không đối xứng này lại được mã hóa bởi các giải thuật RSA/OAEP, sử dụng khóa công khai đã xác định của người nhận.

Trong một cuộc truyền thông bảo mật trên môi trường lưới, cần thiết lập cơ chế bảo mật phía server, client và cơ chế quản lý GRIM (Ánh xạ định danh tài nguyên lưới - Grid Resource Identity Mapper) một cách phù hợp, khi đó dịch vụ bảo mật mới có thể hoạt động được. Sau đây là công thức tổng quát nhất cho bảo mật mức truyền thông điệp:

Grim policy handling + credential delegation + authorization = message security.

4.4.1.1. Các bước để viết một dịch vụ lưới thông thường

Để viết và triển khai một dịch vụ lưới bình thường, chưa hỗ trợ bảo mật, chỉ cần thực hiện theo 5 bước đơn giản sau:

1. Định nghĩa giao diện dịch vụ lưới. Việc này được thực hiện với ngôn ngữ đặc tả giao diện lưới GWSDL.

2. Cài đặt các thực thi cho dịch vụ. Việc này được thực hiện với ngôn ngữ Java.

3. Định nghĩa các tham số triển khai dịch vụ. Việc này được thực hiện với ngôn ngữ mô tả triển khai WSDD.

4. Biên dịch các mô tả và cài đặt trên, sinh stub để cung cấp giao diện cho client và đóng gói gar (chuẩn đóng gói của dịch vụ lưới giống như đóng gói war của các ứng dụng web). Việc này được thức hiện với Ant.

5. Triển khai dịch vụ lên trình chứa Globus Toolkit. Việc này được triển khai vơi Ant.

Nhờ có công cụ phát triển nguồn mở Eclipse, một thành phần mở rộng (plugin) hỗ trợ cho việc phát triển dịch vụ lưới trên Globus Toolkit 3, việc lập trình cho các dịch vụ lưới trở nên đơn giản hơn rất nhiều. Các bước 1, 4 đã được tự động hóa hoàn toàn, chỉ có các bước 2, 3, 5 là dành cho người phát triển. Vậy khi hỗ trợ bảo mật cho các dịch vụ này, mọi việc có đơn giản như vậy không?

4.4.1.2. Hỗ trợ bảo mật cho dịch vụ lưới

Giả sử rằng ta đã thiết lập cấu hình GSI cho lưới, có một nhà thẩm quyền CA để ký và chứng giận cho các giấy chứng nhận, giấy ủy nhiệm theo chuẩn X509. Ngoài ra giả thiết ta đã có các giấy chứng nhận cần thiết cho host (máy mà trình chứa dịch vụ lưới Globus Toolkit đang chạy), người dùng globus (chủ của trình chứa Globus Toolkit), một người dùng gc_user trong lưới. Ngoài ra, cũng cần phải thân thiện bước các kỹ năng cơ bản về lập trình của dịch vụ lưới như phần giới thiệu về dịch vụ lưới ở trên. Phần khó khăn nhất sẽ là làm cho các giấy chứng nhận này hoạt động trên các dịch vụ lưới.

Để hỗ trợ bảo mật cho các dịch vụ lưới, các bước cũng được thực hiện tương tự như các dịch vụ lưới thông thường, chỉ cần một vài thay đổi nhỏ khi chèn thêm các đoạn mã bảo mật vào dịch vụ. Đồng thời việc thiết lập bảo mật phải được thực hiện ở cả hai phía client và dịch vụ.

- Bảo mật phía dịch vụ

1. Định nghĩa giao diện dịch vụ lưới. Việc này được thực hiện với ngôn ngữ đặc tả giao diện lưới GWSDL, không cần thay đổi gì khi hỗ trợ bảo mật và có thể thực hiện tự động bởi Eclipse.

2. Cài đặt các thực thi cho dịch vụ. Việc này được thực hiện với ngôn ngữ Java, có thể thêm vào các đoạn mã bảo mật khi cần thiết, dựa trên các thư viện bảo mật của OGSA, Java, IETF.

3. Định nghĩa các tham số triển khai dịch vụ. Việc này được thực hiện với ngôn ngữ mô tả triển khai WSDD, cần có một số thay đổi khi hỗ trợ bảo mật: các thẻ định nghĩa file cấu hình bảo mật, xác định phương thức chứng thực, hay các thẻ định nghĩa về giấy ủy nhiệm, giấy chứng nhận, khóa bí mật, ...

4. Viết file cấu hình bảo mật (security-config.xml), file định nghĩa danh sách điều khiển truy nhập tới dịch vụ (grid-mapfile). Bước này là tùy chọn, có thể sử dụng các file mặc định của hệ thống, tuy nhiên đối với các dịch vụ bảo mật phức tạp thì đây lại chính là bước đặc trưng để phân biệt dịch vụ lưới bảo mật vơi các dịch vụ lưới thông thường, với việc định nghĩa các mức bảo vệ như mã hóa (encryption), bí mật (privacy), ký (signature) trong file cấu hình bảo mật, hay định nghĩa các mức chứng thực như không có chứng thực (NoAuthorization), tự chứng thực (SelfAuthorization), và chứng thực trên host (HostAuthorization).

5. Biên dịch các mô tả và cài đặt trên, sinh stub để cung cấp giao diện cho client và đóng gói gar (chuẩn đóng gói của dịch vụ lưới giống như đóng gói war của các ứng dụng web). Việc này được thức hiện với Ant, được tự động hóa bởi Eclipse

6. Triển khai dịch vụ lên trình chứa Globus Tollkit. Việc này được triển khai vơi Ant.

Sau khi đã thực hiện đầy đủ các bước trên, để hoàn thiện một dịch vụ bảo mật, cần cài thêm các client có hỗ trợ bảo mật trên dịch vụ đó. Do mô hình dịch vụ lưới cho phép client và server phát triển độc lập, việc thức hiện bảo mật trên client cũng hoàn toàn độc lập với dịch vụ. Thông thường các hỗ trợ bảo mật được thực hiện dựa trên các thiết lập trên stub client.

- Bảo mật phía client

Thiết lập bảo mật phía client thường được thực hiện bằng việc thiết lập các tùy chọn cho stub phía client. Nó bao gồm các bước sau đây:

1. Lựa chọn mức độ bảo mật cho client. Có hai mức độ bảo mật là bảo mật mức thông điệp (GSI Secure Message) và bảo mật mức hội thoại (GSI Secure Conversation). Mức bảo mật thông điệp được dựa trên cơ chế chứng thực khóa công khai, các thông điệp được ký và mã hóa theo chuẩn X509. Với mức bảo vệ hội thoại, một ngữ cảnh bảo mật được thiết lập giữa client và server, các thông điệp sẽ được ký, mã hóa hay bảo vệ tính toàn vẹn dựa trên ngữ cảnh bảo mật này.

2. Lựa chọn mức chứng thực. Có các mức chứng thực sau được client thiết lập để tin tưởng các định danh phía dịch vụ: không chứng thực (NoAuthorization), tự chứng thực (SelfAuthorization) và chứng thực host (HostAuthorization). Đối với không chứng thực, client không đòi hỏi dịch vụ phải có giấy ủy nhiệm. Tự chứng thực, đòi hỏi dịch vụ phải có định danh trùng với định danh mà client đang thực hiện. Chứng thực host, đòi hỏi dịch vụ phải có giấy chứng nhận hợp lệ của host (máy mà dịch vụ đang chạy trên đó).

3. Lựa chọn cơ chế quản lý GRIM. Trước hết, cần phải giới thiệu một chút về GRIM. Giấy ủy quyền GRIM (Ánh xạ thực thể tài nguyên lưới - Grid Resource Identity Mapper) là giấy chứa các chính sách ủy quyền để liệt kê các thực thể (tên định danh/tên đối tượng) lưu giữ một phần khóa bí mật của GRIM. Kiến trúc GRIM cho phép cơ chế ủy quyền truy cập tài nguyên (e.g. host). Điều này được thực hiện thông qua một chương trình setuid, có quyền truy nhập giấy ủy nhiệm dài hạn của tài nguyên, và chương trình tạo ra một giấy ủy quyền GRIM chứa danh sách của các thực thể. Danh sách các thực thể đưa ra bao gồm ánh xạ <định danh>

<người dùng> được chứa trong grid-mapfile, để xem người dùng có hợp lệ trên tài nguyên hay không.

Khi GRIM chứa các chính sách ủy quyền không theo tiêu chuẩn, nó cần một bộ quản lý đặc biệt trong khi chứng thực. Bộ quản lý đặc biệt này sẽ cài đặt một thực thi để giải quyết vấn đề ủy quyền GRIM. Hiện tại, có hai quản lý chính sách GRIM, một loại bảo đảm một trong các thực thể trong danh sách các thực thể là khớp với thực thể được đưa ra, và loại kia lờ đi nội dung các chính sách của GRIM.

4.4.1.3. Ví dụ về một dịch vụ bảo mật lưới đơn giản

Sau khi đã đưa ra tổng quan các bước để viết một dịch vụ lưới hoàn chỉnh, phần này sẽ giới thiệu cụ thể các bước trên được thực hiện như thế nào. Dịch vụ lưới được đưa ra là một dịch vụ cung cấp tính toán thông thường, với các phương thức cộng, trừ và lấy giá trị hiện tại thức add, subtract, và getValue. Tuy nhiên, khác với dịch vụ lưới thông thường, dịch vụ này có hỗ trợ bảo mật, mỗi khi có client bất kì sử dụng dịch vụ, thông tin về định danh client, định danh dịch vụ và định danh của hệ thống sẽ được thông báo trở lại dịch vụ qua cơ chế loging.

- Viết bảo mật phía server

Các công việc để cho phép bảo mật phía server cũng hoàn toàn tương tự như bạn viết một dịch lưới thông thường, chỉ có điều là ta thêm vài kỹ thuật nhỏ để hỗ trợ bảo mật ở phía server

+ Đặc tả giao diện cho dịch vụ

File đặc tả giao diện của dịch vụ đặc tả các phương thức add, subtract, và getValue theo định dạng GWSDL, và được sinh tự động nên không có gì đáng chú ý.

+ Cài đặt dịch vụ

Một nhà cung cấp dịch vụ (OperationProvider) được sử dụng để cài đặt cho dịch vụ. Nó sẽ bao gồm cài đặt cho các phương thức add, subtract, và getValue được đặc tả ở giao diện, các phương thức gọi ngược (callback methods) được thừa kế từ lớp OperationProvider (ở đây chỉ cài đặt một phương thức postCreate), và một phương thức private logSecurityInfo được dùng để ghi ra các thông tin bảo mật khi dịch vụ hoạt động. Chúng ta sẽ thảo luận kĩ hơn về các thành phần liên quan trong mã nguồn.

Cài đặt của các phương thức chính, thể hiện chức năng của dịch vụ là rất đơn giản. Chỉ khác là ta triệu gọi phương thức logSecurityInfo trong tất cả các phương thức này, để ghi lại các thông tin về bảo mật khi thực hiện chúng.

```
public void add(int a) throws RemoteException
{
  logSecurityInfo("add");
  value = value + a;
```

```
}
<- ... ->
```

Phương thức gọi ngược postCreate đơn giản chỉ gọi logSecurityInfo:

```
public void postCreate(GridContext context) throws
GridServiceException
{
  logSecurityInfo("postCreate");
}
```

Cuối cùng, phương thức logSecurityInfo ghi một số thông tin bảo mật tới log của trình chứa. Không có gì đặc biệt trong đoạn mã này, ngoại trừ nó ghi ra định danh của người gọi, như phần in đậm trong đoạn mã dưới đây:

```
private void logSecurityInfo(String methodName)
{
  Subject subject;
  logger.info("SECURITY INFO FOR METHOD '" + methodName + "'");

  // Print out the caller
  String identity = SecurityManager.getManager().getCaller();
  logger.info("The caller is:" + identity);

  // Print out the caller's subject
  subject = JaasSubject.getCurrentSubject();
  logger.info("INVOCATION SUBJECT");
  logger.info(subject==null?"NULL":subject.toString());
<-- ... -->
```

Như vậy, có thể thấy cài đặt bảo mật cho ứng dụng lưới không hề ảnh hưởng gì mã nguồn phía sever, ít nhất là trong ví dụ này. Tuy nhiên, đối với các kịch bản bảo mật phức tạp, đòi hỏi phải thêm một số thay đổi mã nguồn phía server.

+ Mô tả triển khai dịch vụ

Để thông báo cho trình chứa biết là dịch vụ của chúng ta sắp hỗ trợ bảo mật, cần thêm các tham số sau vào mô tả:

File cấu hình bảo mật (tham số securityConfig)

Phương thức chứng thực được sử dụng (tham số authorization)

Tham số securityConfig

Tham số này cho biết nơi có thể tìm thấy file cấu hình bảo mật (security configuration file). Đây là một file XML, bao gồm chi tiết các cấu hình bảo mật, như là mức bảo mật đòi hỏi khi triệu gọi phương thức. Ở ví dụ này hết sức đơn giản là sử dụng file cấu hình bảo mật mặc định của GT3. File cấu hình mặc định này sẽ đưa ra các cấu hình bảo mật cần thiết cho dịch vụ. Cần thêm những dòng sau vào file mô tả triển khai server-deploy.wsdd:

```
<parameter name="securityConfig"
value="org/globus/ogsa/impl/security/descriptor/gsi-security-
config.xml"/>
```

Tham số authorization

Tham số này cho phép ta xác định kiểu chứng thực GSI phía server được dùng trong dịch vụ này. Hiện tại, sử dụng kiểu chứng thực đơn giản nhất: no authorization.

```
<parameter name="authorization" value="none"/>
```

- Viết bảo mật phía client

Chú ý là trước khi sử dụng, phải dùng grid-proxy-init để khởi tạo một giấy ủy nhiệm cho client. Để cấu hình cho client triệu gọi được các phương thức bảo mật cơ bản này, cần thêm vào các dòng sau:

```
<-- ... -->
((Stub)math)._setProperty(Constants.GSI_SEC_CONV,Constants.ENCRYPTIO
N);
((Stub)math)._setProperty(Constants.AUTHORIZATION,NoAuthorization.ge
tInstance());
((Stub)math)._setProperty(Constants.GRIM_POLICY_HANDLER,new
org.globus.gsi.proxy.IgnoreProxyPolicyHandler());
```

Các dòng mã này sẽ cấu hình lớp stub(môt đối tượng MathPortType) để sử dụng bảo mật. Cụ thể chúng thực hiện như sau:

Dòng đầu cho biết stub sử dụng mức bảo mật là mã hóa (full-blown encryption). GSI cho phép chúng ta lựa chọn các mức bảo mật khác nhau, mã hóa chỉ là một trong số chúng. Ví dụ đã lựa chọn chữ kí điện tử để bảo vệ tính toàn vẹn của thông điệp.

Dòng thứ hai cho biết stub sử dụng cơ chế chứng thực: no authorization ở phía client. Chú ý là các chứng thực ở phía client và server là khác nhau.

Dòng thứ ba là thiết lập GRIM_POLICY_HANDLER cho stub. Ở đây, sử dụng là "ignore handle", tức là lờ đi cơ chế quản lý của GRIM.

Chú ý: Do giấy chứng nhận GRIM không phải chuẩn X509, cho nên nếu không có dòng 3, sẽ gặp lỗi sau:

```
org.globus.gsi.proxy.ProxyPathValidatorException: Unknown policy:
1.3.6.1.4.1.3536.1.1.1.7
```

- Triển khai và thực thi dịch vụ

Triển khai dịch vụ ở tài khoản globus:

```
ant deploy \
-Dgar.name=$GLOBUS_LOCATION/gars/security_first.gar
```

Kích hoạt logging

Khi sử dụng lớp logging phía dịch vụ, chúng ta cần kích hoạt logging cho dịch vụ, bằng cách hiệu chỉnh file $GLOBUS_LOCATION/ogsilogging.properties:

```
org.globus.progtutorial.services.security.first.impl.MathProvider=co
nsole,info
```

Khởi tạo trình chứa

Trước khi khởi tạo trình chứa, bảo đảm rằng bạn đã có một giấy ủy quyền cho tài khoản globus. Sau khi đã có giấy ủy quyền, khởi tạo trình chứa trên tài khoản người dùng globus:

```
globus-start-container
```

Thực thi phía client

Trước khi thực hiện bất cứ một ứng dụng client nào, chú ý là đã có một giấy ủy quyền cho tài khoản người dùng gc_user.

```
java \
-classpath ./build/classes/:$CLASSPATH \
org/globus/progtutorial/clients/MathService/ClientGSIConvEncrypt \
http://127.0.0.1:8080/ogsa/services/progtutorial/security/first/Math
Service \
5
```

Nếu mọi chuyện tốt đẹp, phía client:

```
Added 5
Subtracted 1
Current value: 4
```

Những thông tin sau ở phía server

```
INFO: SECURITY INFO FOR METHOD 'add'
```

```
INFO: The caller is:/O=Globus/OU=GT3 Tutorial/CN=Borja Sotomayor
<-- ... ->
```

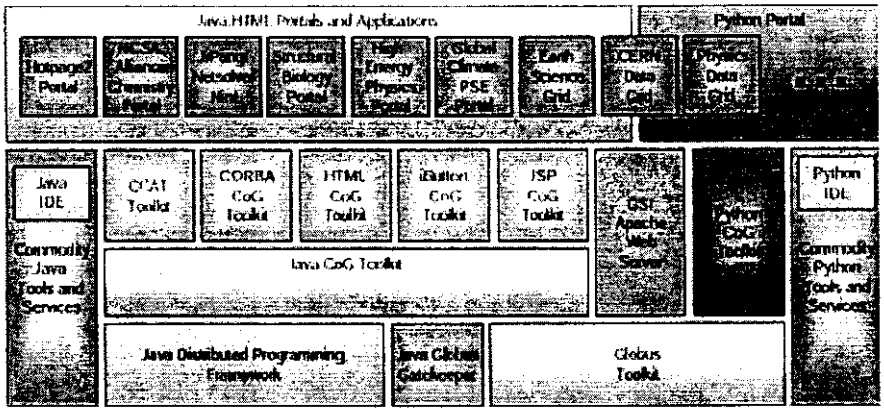
Chú ý: định danh của người gọi trong giấy chứng nhận là gc_user, và định danh của hệ thống phụ thuộc vào định danh trong giấy chứng nhận của người dùng globus.

4.4.2. Bảo mật trên Java Cog Kit

4.4.2.1. Giới thiệu

Sự phát triển của Internet và tính toán phân tán dẫn tới sự phát triển nhanh chóng của tính toán phân tán trong một vài lĩnh vực. Trong thế giới của sản phẩm tính toán thương mại, một phạm vi lớn các công nghệ tính toán phân tán (các giao thức Web, Java, Jini, Corba, DCOM) đã nổi lên với những ảnh hưởng đáng kể với cách thức mà chúng ta truy nhập và xử lý thông tin. Cùng lúc đó, cộng đồng tính toán hiệu năng cao có bước phát triển tiếp theo với việc tạo ra công nghệ Grid, với cơ sở hạ tầng cao cấp được thiết kế để tận dụng các nguồn tài nguyên phân tán cho việc giải quyết các vấn đề khoa học.

Dự án Commodity Grid Toolkit (CoG Kit) được ra đời để liên kết giữa các công nghệ tính toán thông dụng với công nghệ lưới. Dự án này đưa ra định nghĩa và cài đặt cụ thể một tập các thành phần chung, ánh xạ các chức năng của Grid tới các công cụ lập trình thông dụng như Web/CGI, Java, Corba, DCOM, ...



Hình 4-13: Cog Kit và các ứng dụng đang phát triển

Mục đích chính của dự án là dễ dàng phổ biến và sử dụng rộng rãi hơn công nghệ lưới, phát triển các dịch vụ lưới cao cấp, phát triển dễ dàng và nhanh chóng hơn các ứng dụng lưới, tái sử dụng mã nguồn các kho lưu trữ thành phần, sử dụng dịch vụ Web như một phần của lưới, và đặc biệt là phát triển ứng dụng qua các công cụ lập trình thông dụng là không hạn chế ở phía client.

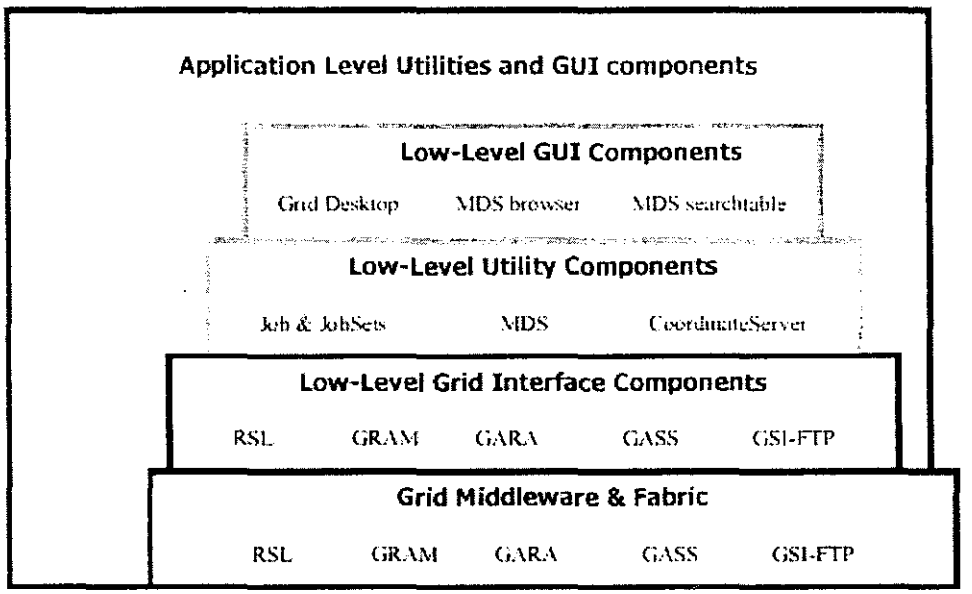
4.4.2.2. Bộ công cụ Java Cog Kit

Java Cog Kit là bộ cung cụ phát triển lưới, ánh xạ giữa công nghệ Java và Globus Toolkit, cho phép các nhà phát triển xây dựng các ứng dụng lưới bằng ngôn ngữ Java.

Java Cog Kit được lựa chọn để phát triển lưới bởi một số lý do sau đây:

- Tối đa hóa sự năng động, khả năng mở rộng và tái sử dụng phần mềm.
- Ngôn ngữ Java là độc lập về hệ điều hành và các nền tảng phát triển, do vậy nó tương thích với môi trường không đồng nhất như lưới. Ngoài ra nó cung cấp nhiều thư viện cho phép phát triển các ứng dụng mạng, các kỹ thuật liên tác JAAS, JINI, CORBA, IIOP, ...
- Sử dụng cơ sở hạ tầng bảo mật lưới GSI từ Java.

Dưới đây là kiến trúc của Java Cog Kit



Hình 4-14: Kiến trúc của Java Cog Kit

1. Grid Middleware & Fabric: Đây là tầng thấp nhất trong kiến trúc, chứa tất cả các gói để giao tiếp với các dịch vụ và các tài nguyên. Hiện tại Java Cog Kit cung cấp một số dịch vụ cơ bản sau đây:

An toàn và bảo mật (GSI).

Đệ trình và theo dõi các công việc thực thi từ xa (GRAM).

Đánh giá chất lượng dịch vụ (GARA).

Dịch vụ truyền file bảo mật (GSIFTP)

Dịch vụ tra cứu thông tin (MDS)

Dịch vụ lưu trữ giấy ủy nhiệm (MyProxy)

2. Các thành phần giao diện lưới mức thấp (Low-Level Grid Interface Components): ánh xạ tới các dịch vụ lưới hay sử dụng, được xây dựng bằng công nghệ Java Bean và Enterprise Java Bean.

3. Các thành phần tiện ích mức thấp (Low-Level Utility Components): cung cấp các tiện ích mức thấp cho phép tăng hiệu quả trong việc liên kết các thành phần Globus Toolkit được cài đặt bằng ngôn ngữ C.

4. Các thành phần giao diện mức thấp (Low-Level GUI Components): cung cấp tập các thành phần giao diện mức thấp có khả năng tái sử dụng như: bộ soạn thảo LDAP, RSL, trình duyệt LDAP, ...

5. Các thành phần giao diện và tiện ích mức ứng dụng (Application Level Utility & GUI components): là cầu nối giữa Java Cog với các ứng dụng thực tế.

Với kiến trúc trên, Java Cog Kit cho phép

- Các nhà phát triển tạo ra các thành phần trung gian (middleware), liên kết giữa nhà cung cấp dịch vụ và các nhà phát triển ứng dụng phía trên.
- Các nhà phát triển ứng dụng GridPortal dễ dàng truy cập tài nguyên lưới và phát triển dịch vụ lưới như một dịch vụ của Portal.

4.4.2.3. Cơ chế bảo mật trên Java Cog Kit

Cơ chế bảo mật của Java Cog Kit được thực hiện qua tầng socket, với hai chế độ bảo mật là SSL và GSI. Tất cả các mức bảo mật trong cơ sở hạ tầng lưới GSI như giới thiệu ở chương II (chứng thực, ủy nhiệm, tin tưởng, mã hóa, bí mật, ...) đều được thực hiện qua mã nguồn Java, và thiết lập vào một ngữ cảnh bảo mật mà cả client và server đều thống nhất, quản lý bởi ExtendedGSSContext. Phía server sẽ khởi tạo tại một cổng xác định, như một thực thể của lớp ServerSocket. Phía client sẽ kết nối tới server thông qua tên host và địa chỉ cổng, khởi tạo như một thực thể của lớp Socket. Có thể điểm qua một vài chế độ được thiết lập ở ngữ cảnh bảo mật như sau:

+ Chứng thực (authorization): có các mức chứng thực: NoAuthorization, SelfAuthorization, HostAuthorization và IdentityAuthorization. Tùy theo mức chứng thực được thiết lập, client hay server mong đợi bên kia đang thực hiện dưới định danh nào. Ví dụ, với HostAuthorization được thiết lập ở phía client, bên server phải thực hiện chương trình của mình dưới định danh host, nếu không quá trình chứng thực đa phương sẽ không thành công.

+ Ủy nhiệm (delegation): có các mức ủy nhiệm hạn chế (limited), đầy đủ (full) và không ủy nhiệm (none), được thiết lập để client hay server đồng ý hoặc không đồng ý ủy nhiệm giấy ủy nhiệm của mình cho bên kia.

+ Tin tưởng (confident): nếu cơ chế này được thiết lập, socket sẽ được mã hóa trước khi truyền sang client hay server.

+ Vô danh (anonymous): cơ chế này được thực hiện khác nhau ở client và server. Ở phía client, nếu client, nếu anonymous được thiết lập, client sẽ không gửi giấy ủy nhiệm của mình sang server trong truyền thông bảo mật. Còn phía server, anonymous được thiết lập thì sẽ không đòi hỏi client phải chứng thực, hay mức chứng thực được thiết lập ở server là NoAuthorization.

+ Chế độ (mode): cho phép client và server lựa chọn hai chế độ bảo mật SSL hay GSI.

Việc thực hiện chứng thực và thẩm quyền trong Java Cog Kit được thực hiện với các giấy ủy nhiệm theo chuẩn X509, với các thư hiện bảo mật GSS API hỗ trợ để lấy các giấy ủy nhiệm này. Đoạn mã dưới đây cho phép lấy giấy ủy nhiệm mặc định, lấy chứng thực với định danh host.

```
//Default credential
GSSCredential cred =
ExtendedGSSManager.getInstance().createCredential(GSSCredential.INIT
IATE_AND_ACCEPT);
//Host authorization
Authorization auth = HostAuthorization.getInstance();
```

Đây là đoạn mã ví dụ về kết nối với máy chủ GridFTP trong Java Cog Kit, với yêu cầu chứng thực host ở phía máy chủ và sử dụng giấy ủy nhiệm mặc định để thẩm quyền và đăng nhập vào máy chủ GridFTP.

```
//Connect to GridFTP server
public GridFTPClient connect() throws Exception {
//Default credential
GSSCredential cred =
ExtendedGSSManager.getInstance().createCredential(GSSCredential.INIT
IATE_AND_ACCEPT);
//Host authorization
Authorization auth = HostAuthorization.getInstance();
client.setAuthorization(auth);
client.authenticate(cred);
System.out.println("Connect successfully!");
return client;
}
```

Hiện tại, với các API mà Java Cog Kit cung cấp, ta có thể thực hiện đầy đủ các cơ chế chứng thực trong GSI, kết nối với các dịch vụ lưới có sẵn có hỗ trợ GSI để cung cấp các ứng dụng cho mức người dùng. Hàng loạt các ứng dụng lưới về truyền file bảo mật, đề trình công việc từ xa, dịch vụ thông tin lưới bảo mật, và đặc biệt bảo mật cho các ứng dụng GridPortal đều được thực hiện dễ dàng với các giao diện lập trình ứng dụng của Java Cog Kit.

4.4.3. Bảo mật trên Portal

Cổng dịch vụ Portal là cổng kết nối dịch vụ giữa người dùng và nhà cung cấp dịch vụ, được phát triển như một phần mềm trên mạng Intenet để cung cấp các chức năng cần thiết theo hướng người dùng. Việc sử dụng công nghệ Portal cho phép tạo môi trường làm việc riêng biệt cho từng người dùng, ở đây người dùng có thể tùy chọn một số thay đổi môi trường làm việc của mình, đồng thời tách biệt các chức năng dịch vụ riêng biệt từ phía máy chủ và tái sử dụng các thành phần chức năng của Web. Một số chuẩn nổi tiếng đang được sử dụng rộng rãi như: IBM Websphere Portlet của IBM, JSR 168 Portlet của Sun.

4.4.3.1. Bảo mật trên Grid Portal

- Các yêu cầu bảo mật

Hiện tại, khi phát triển các Grid Portal, một số yêu cầu sau liên quan tới vấn đề bảo mật như quản lý giấy ủy nhiệm lưới, giao tiếp với cơ sở hạ tầng bảo mật GSI thông qua Portal:

+ Người sử dụng phải có một trình duyệt Web chuẩn để tiếp cận các Grid Portal.

+ Người sử dụng vẫn có thể truy nhập từ những nơi mà những nơi mà giấy chứng nhận Grid là không có sẵn đối với họ.

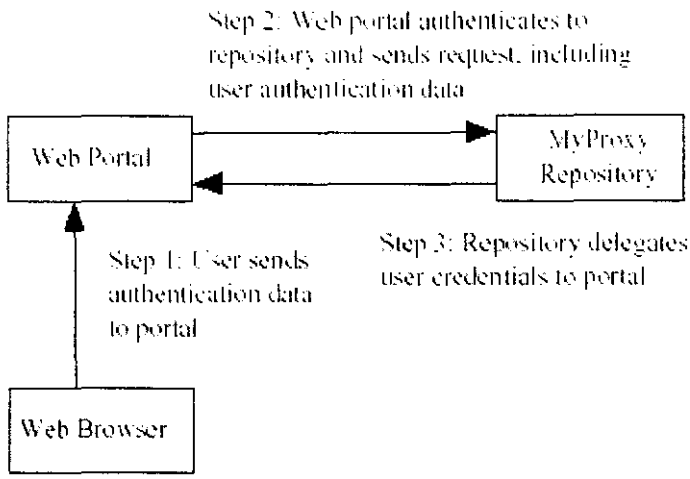
+ Người sử dụng có thể làm bất cứ điều gì thông qua Grid Portal, mà giấy ủy quyền có thể cho phép họ làm.

Tuy nhiên, hiện tại không phải tất cả các ứng dụng cho Grid đều hỗ trợ GSI, và như vậy thiếu khả năng ủy nhiệm giấy ủy quyền. Một ví dụ đơn giản là sử dụng trình duyệt Web như là máy khách đối với các portal dựa trên Web. Điều này nghĩa là người dùng có thể chứng thực với portal, nhưng lại không thể ủy nhiệm cho portal thay mặt họ được.

Do những yêu cầu trên, Grid Portal buộc phải đưa ra những đặc quyền đặc biệt mãi mãi trên portal để thay mặt người dùng trên lưới. Điều này thường được thực hiện bởi việc sở hữu giấy ủy nhiệm dài hạn của portal đối với người dùng. Tuy nhiên, mỗi portal lại muốn có một giấy ủy nhiệm của người dùng để có sự thuận tiện cao nhất. Vì thế mà sự tiết lộ các giấy ủy quyền và nguy cơ các giấy ủy quyền này bị đánh cắp. Ngoài ra, đòi hỏi phải lưu trữ giấy ủy quyền dài hạn của người dùng một cách bí mật cũng tăng thêm những yêu cầu bảo mật của chính portal.

- Sử dụng kho lưu trữ MyProxy với Grid Portal

Một giải pháp được đưa ra là sử dụng kho lưu trữ giấy ủy nhiệm MyProxy với Grid Portal, thể hiện qua 3 bước như hình vẽ:



Hình 4-15: Portal tương tác với MyProxy

1. Người dùng có thể kết nối với Grid Portal (sử dụng trình duyệt Web) và cung cấp các thông tin chứng thực đã đưa ra trước đây tới kho lưu trữ thông qua các form trên web hay các giao diện đơn giản. Người sử dụng có thể xác định một kho lưu trữ MyProxy cho Portal để sử dụng, nếu có nhiều hơn một kho.
2. Portal có thể kết nối với kho lưu trữ sử dụng chương trình myproxy-get-delegation, tự chứng thực sử dụng giấy chứng nhận Grid của nó, đưa ra các thông tin chứng thực (định danh và mật khẩu) như là đưa ra bởi người dùng, và yêu cầu một proxy cho người dùng.
3. Kho lưu trữ kiểm tra tất cả các thông tin được trình bày, có thể ủy nhiệm một giấy ủy quyền của người dùng ngược trở lại với portal. Portal lúc đó có thể tiếp cận an toàn tới Grid, sử dụng các phần mềm chuẩn của Grid như một người sử dụng bình thường. Lúc này, người dùng có thể kết nối trực tiếp thông qua trình duyệt Web tới Portal.

Hành động logout ra khỏi portal sẽ xóa giấy ủy quyền của người dùng trên portal. Khi người dùng quên logout, giấy chứng nhận sẽ mãn hạn tại một thời điểm nhất định khi yêu cầu từ dịch vụ MyProxy. Thời gian sống thường là khoảng vài giờ. Quá trình này có thể lặp lại nhiều lần, khi người dùng đòi hỏi, cho tới khi giấy ủy quyền giữ bởi MyProxy mãn hạn. Tại thời điểm này, người sử dụng cần chạy chương trình myproxy-init từ vị trí nơi các giấy chứng nhận có sẵn và ủy nhiệm tập mới các giấy ủy quyền tới kho lưu trữ. Thời gian lớn nhất của giấy ủy nhiệm được kho lưu trữ xác định bởi máy chủ lưu trữ, nhưng mặc định là một tuần/lần.

Giải pháp trên đã được cài đặt trong hệ thống Gridsphere Portal, sử dụng các thư viện lập trình của Java Cog Kit, sẽ được trình bày cụ thể hơn ở phần tiếp theo.

4.5. Dịch vụ bảo mật triển khai trong hệ thống BKGrid 2005

Dịch vụ bảo mật trong BKGrid 2005 được xây dựng ở tầng trên cùng của BKGrid 2005, không đi sâu vào cơ sở hạ tầng bảo mật phức tạp của các dịch vụ lưới ở phía dưới mà tập trung cung cấp hạ tầng bảo mật ở tầng trên, là thành phần trung gian để trung chuyển giữa bảo mật lưới và bảo mật portal. Nhiệm vụ chính của dịch vụ là cung cấp và trao chuyển các giấy ủy nhiệm lưới theo chuẩn X509 của người dùng, thiết lập một môi trường bảo mật an toàn, thuận tiện cho người dùng qua cổng điện tử portal, đồng thời bảo đảm đầy tính chứng thực và thẩm quyền, cũng như các khả năng bảo mật khác của các tài nguyên lưới phía dưới. Dịch vụ này bảo đảm cung cấp các chức năng bảo mật cần thiết cho người dùng, đồng thời sẽ hỗ trợ bảo mật cho các dịch vụ lưới khác khi có yêu cầu.

4.5.1. Các đòi hỏi bảo mật trong BKGrid 2005

Giải pháp cổng điện tử cho hệ thống BKGrid 2005 cung cấp một giao diện truy nhập dịch vụ thống nhất, tạo môi trường làm việc (workspace) riêng biệt cho mỗi người dùng khi sử dụng dịch vụ, và đặc biệt là có tính mở và tính khả chuyển rất cao.

Tuy nhiên, một điều đáng tiếc, các giao thức bảo mật chuẩn, được sử dụng giữa máy chủ Web và các trình duyệt không đáp ứng được các yêu cầu của BKGrid 2005. Hệ thống yêu cầu người dùng phải xác thực được danh tính của mình trong lưới, hoặc là ủy quyền trên máy chủ, máy chủ có thể thay mặt người dùng để khởi tạo và quản lý các giao dịch cho người dùng đó trong một tài nguyên lưới. Các tài nguyên lưới thường được bảo vệ bởi cơ sở hạ tầng lưới GSI (Grid Security Infracstructure), một thành phần của bộ công cụ Globus Toolkit, và hiện nay đang là chuẩn cho bảo mật trên môi trường lưới. Trong khi GSI hỗ trợ ủy quyền, các giao thức chuẩn của Web lại không có tính năng này. Điều này dẫn tới sự không tương thích giữa cơ chế bảo mật trên môi trường lưới và môi trường Web, ảnh hưởng tới hoạt động của hệ thống. Dịch vụ bảo mật cho BKGrid 2005 sẽ phải có nhiệm vụ kết hợp giữa bảo mật lưới và bảo mật Web, khiến cho hệ thống có thể sử dụng các tài nguyên được bảo vệ bởi GSI một cách an toàn. Các yêu cầu cụ thể đối với dịch vụ như sau:

- Kết nối với tài nguyên lưới: Do hệ thống phát triển dựa trên nền portal Gridsphere của IBM, mà portal này không hỗ trợ cơ sở hạ tầng bảo mật lưới GSI, cho nên nếu người dùng sử dụng tài nguyên lưới hay các dịch vụ lưới có hỗ trợ bảo mật GSI thì nhất thiết họ phải có giấy ủy nhiệm lưới để xác thực, thẩm quyền cũng như thực hiện các cơ chế ủy nhiệm khác. Đồng thời tài nguyên cũng phải chứng minh tính đúng đắn của mình trong môi trường lưới, bảo đảm là một thành viên, một tổ chức ảo trong lưới. sẵn sàng cung cấp các dịch vụ theo yêu cầu người dùng. Như vậy là hệ thống cần có một dịch vụ để lưu trữ, quản lý và cung cấp giấy chứng nhận, chứng thực và thẩm quyền cho người dùng và các tài nguyên trong lưới.

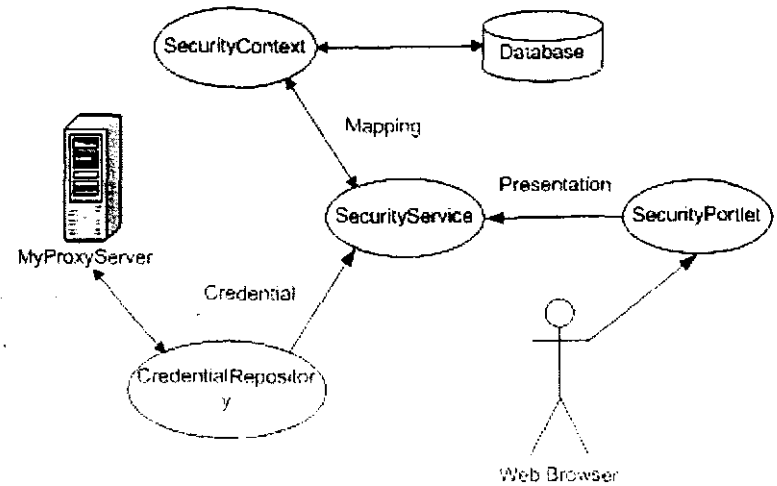
- Ủy nhiệm: Người dùng sử dụng trình duyệt Web để đăng nhập vào hệ thống, các trình duyệt Web như là máy khách đối với Portal (BKGrid2005), người dùng có thể chứng thực với Portal, nhưng lại không thể ủy nhiệm cho Portal thay mặt cho họ được. Một yêu cầu đặt ra là Portal phải đưa ra những đặc quyền đặc biệt mãi mãi để thay mặt người dùng trên lưới, hay nói cách khác là phải sở hữu giấy ủy nhiệm dài hạn của người dùng. Khi đó, việc thực thi các tiến trình trong lưới có thể được thực hiện qua portal, sau đó thông báo lại kết quả cho người dùng. Hay có thể áp dụng cơ chế đăng nhập một lần qua portal, người dùng không phải khai báo nhiều lần khi thực hiện nhiều công việc, nhiều tiến trình, mọi việc chỉ cần ủy nhiệm cho portal.
- Thuận tiện: Hệ thống phải cung cấp một giao diện đồ họa thân thiện, người sử dụng ở bất cứ đâu cũng có thể dễ dàng truy nhập vào hệ thống, cho dù là ở đó giấy chứng nhận lưới là không có sẵn đối với họ. Ngoài ra, họ chỉ việc đăng nhập một lần, và có thể làm bất cứ điều gì trong hệ thống BKGrid 2005 mà giấy chứng nhận Grid cho phép họ làm.

Bên cạnh những yêu cầu bảo mật cơ bản trên, việc lưu trữ và cung cấp các đặc quyền trên giấy ủy nhiệm dài hạn của người dùng để có sự thuận tiện cao nhất, nguy cơ dễ bị tiết lộ các giấy ủy nhiệm và các giấy ủy nhiệm này bị đánh cắp cũng là một trong những vấn đề bảo mật đáng quan tâm.

4.5.2. Kiến trúc dịch vụ bảo mật

Dịch vụ bảo mật được xây dựng sẽ đáp ứng các yêu cầu về bảo mật đối với hệ thống, cho phép người dùng tương tác với các tài nguyên lưới, sử dụng các dịch vụ được cung cấp với tính thuận tiện cao nhất.

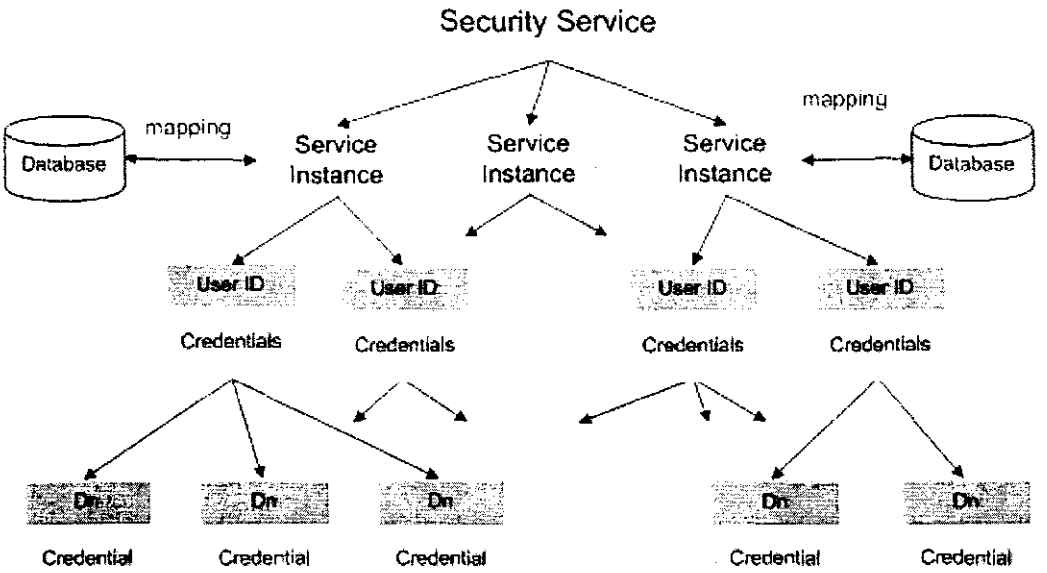
Dịch vụ được xây dựng theo công nghệ dịch vụ portlet (Portlet Service), hướng người dùng, chức năng chính là cung cấp một ngữ cảnh bảo mật ổn định trên Portal, ánh xạ giữa các các ngữ cảnh bảo mật này với các giấy ủy nhiệm trong môi trường lưới, khả năng truy xuất giấy ủy nhiệm thông qua máy chủ MyProxy. Kiến trúc dịch vụ được mô tả như sau:



Hình 4-16: Kiến trúc dịch vụ bảo mật

- Dịch vụ bảo mật (SecurityService): là thành phần trung tâm trong kiến trúc, truy xuất giấy ủy nhiệm thông qua kho lưu trữ giấy ủy nhiệm (CredentialRepository), thiết lập các thông tin liên quan tới giấy ủy nhiệm, người dùng portal, người dùng lưới vào trong các ngữ cảnh bảo mật (SecurityContext), sau đó lưu vào trong một cơ sở dữ liệu ổn định. Dịch vụ bảo mật cung cấp một số giao diện API trên portal cho các portlet bảo mật (SecurityPorlet), cho phép có thể quản lý các thông tin về giấy ủy nhiệm (thêm, sửa xóa, xem thông tin), kích hoạt hay hủy kích hoạt các giấy ủy nhiệm có sẵn, người dùng lựa chọn các giấy ủy nhiệm mặc định khi chứng thực trong lưới, hay thay đổi địa chỉ các kho lưu trữ.
- Kho lưu trữ giấy ủy nhiệm (CredentialRepository): kho lưu trữ được xây dựng thông qua các API MyProxy của JavaCog Kit, được xem như một máy khách MyProxy, kết nối tới máy chủ và thực hiện hầu hết các giao tiếp với máy chủ MyProxy như: truy xuất giấy ủy nhiệm, lưu giấy ủy nhiệm vào kho chứa, thay đổi mật khẩu, tên giấy, hủy giấy ủy nhiệm khi không dùng nữa,... Chức năng chính của kho lưu trữ là cung cấp các khả năng về trung chuyển giấy ủy nhiệm ở trên cho dịch vụ bảo mật.
- Máy chủ MyProxy: lưu trữ các giấy ủy nhiệm trong môi trường lưới tại một thư mục nào đó trong máy cục bộ, cung cấp các giao diện API cho phép truy xuất, thay đổi các thông tin về giấy ủy nhiệm như tên, mật khẩu,... Giấy ủy nhiệm khi lưu trữ phải được mã hóa, phân quyền, và áp dụng những cơ chế bảo mật nghiêm ngặt. Các thông tin lưu trữ trong máy chủ bảo đảm cho chúng ta biết được tên của một người dùng cục bộ, giấy chứng nhận lưới của họ đang sở hữu.
- Ngữ cảnh bảo mật (Security Context): chứa các thông tin do dịch vụ bảo mật ánh xạ vào, là một bản ghi dữ liệu bao gồm định danh người dùng lưới trong portal, các thông tin về giấy ủy nhiệm như thời gian sống, tên giấy ủy nhiệm, tên người dùng, tên định danh, ... Các thông tin này do dịch vụ bảo mật ánh xạ được lưu trữ trong cơ sở dữ liệu HSQL, được kết nối với hệ thống BKGrid 2005 thông qua cầu nối cơ sở dữ liệu Hybernate.
- Portlet bảo mật (SecurityPortlet): giao diện chính của dịch vụ, trực tiếp tương tác với người dùng thông qua cổng Portal. Người dùng thông qua đó, có thể sử dụng hầu hết các tính năng mà các dịch vụ bảo mật trên cung cấp.

Thiết kế kỹ thuật của dịch vụ được thể hiện như hình vẽ:

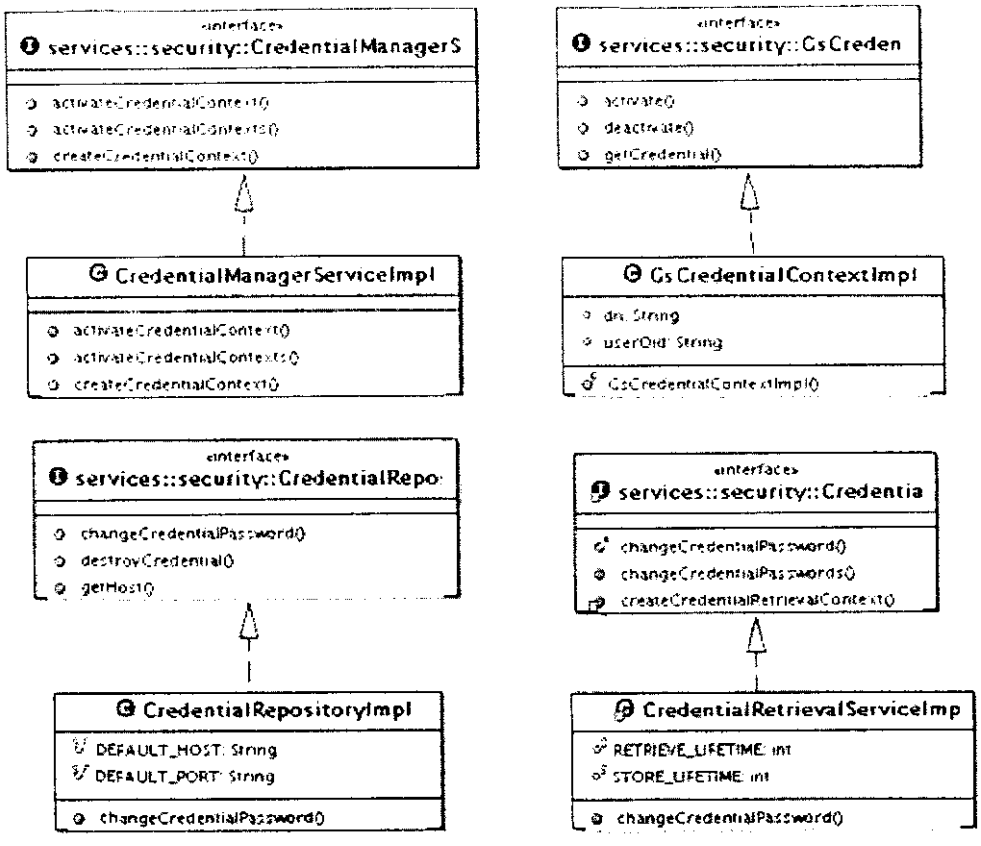


Hình 4-17: Thiết kế kỹ thuật dịch vụ bảo mật

Dịch vụ bảo mật được thực hiện theo kiến trúc của dịch vụ Portlet, mỗi khi có yêu cầu của người dùng, dịch vụ sẽ tạo ra một thực thể phục vụ riêng cho người dùng đó. Nhiệm vụ chính của dịch vụ là lưu các thông tin của người dùng, cả trên portal và trong môi trường lưới trong thời gian thực, sau đó ánh xạ các thông tin đó vào trong cơ sở dữ liệu. Như trên hình vẽ, mỗi thực thể dịch vụ sẽ lưu một bản ghi, được thiết kế dưới dạng bảng băm (HashTable), bao gồm định danh của người dùng (User ID) và tập các giấy ủy nhiệm của người dùng đó (Credentials). Tập các giấy ủy nhiệm cũng được thiết kế theo cơ chế bảng băm, bao gồm trường tên định danh của giấy và giấy ủy nhiệm thực sự theo chuẩn X509. Do cơ chế của bảng băm, mỗi người dùng có id là duy nhất, nên tập các giấy ủy nhiệm mà người dùng đó sở hữu cũng là duy nhất, các giấy ủy nhiệm có tên định danh là duy nhất, nên với mỗi tên định danh chỉ lấy ra được đúng một giấy ủy nhiệm trong tập các giấy ủy nhiệm của người dùng đó. Sau đó, các thông tin cơ bản về id người dùng, tên định danh giấy ủy nhiệm, tên người dùng trong lưới sẽ được ánh xạ vào trong cơ sở dữ liệu dưới dạng các ngữ cảnh bảo mật, khi có yêu cầu chỉ cần tìm trong các bảng băm như vừa mô tả ở trên để lấy ra các giấy ủy nhiệm còn thời hạn. Giải pháp kỹ thuật này sẽ giải quyết khó khăn về quản lý giấy ủy nhiệm của người dùng trên portal, tích hợp giữa bảo mật portal và bảo mật lưới. Tiếp theo sẽ là các mô-đun chính của dịch vụ.

- Các mô-đun chính

Dựa vào kiến trúc và thiết kế trên, các mô-đun chính bao gồm: (sơ đồ lớp)



Hình 4-18: Sơ đồ lớp của các dịch vụ bảo mật

+ Kho lưu trữ giấy ủy nhiệm (CredentialRepository)

Môdun này tương tác trực tiếp với máy chủ MyProxy, cung cấp các khả năng truy xuất, lưu trữ, hủy, thay đổi thông tin về giấy ủy nhiệm. Ngoài ra, nó còn có chức năng truy xuất giấy ủy nhiệm của portal, để thực hiện các cơ chế ủy nhiệm trên portal.

+ Các dịch vụ bảo mật (SecurityServices)

Các dịch vụ này thực hiện các chức năng về quản lý giấy ủy nhiệm, truy xuất giấy ủy nhiệm thông qua mô-đun CredentialRepository, đồng thời ánh xạ các thông tin về giấy ủy nhiệm này vào trong các ngữ cảnh bảo mật. Cụ thể, bao gồm hai dịch vụ chính sau đây:

CredentialManageService: đây là dịch vụ portlet, có chức năng chính là quản lý các ngữ cảnh bảo mật như: thêm, sửa, xóa các ngữ cảnh bảo mật. Ngoài ra, nó còn cung cấp các giao diện API cho phép kích hoạt hay không kích hoạt giấy ủy nhiệm, lấy giấy ủy nhiệm mặc định, lấy ngữ cảnh bảo mật mặc định, lấy các giấy ủy nhiệm còn đang hoạt động, ...

CredentialRetrivalService: dịch vụ này có chức năng chính là truy xuất giấy ủy nhiệm từ kho lưu trữ giấy ủy nhiệm, quản lý giấy ủy nhiệm như thay đổi thông tin, hủy giấy ủy nhiệm. Các thông tin về giấy ủy nhiệm sau đó cũng được ánh xạ vào trong các ngữ cảnh bảo mật.

+ Các ngữ cảnh bảo mật (SecurityContexts)

Như đã giới thiệu, các ngữ cảnh này được thiết lập khi người dùng truy nhập vào trong môi trường lưới sử dụng giấy ủy nhiệm của mình. Thực chất, ngữ cảnh này chính là bản tóm tắt các thông tin về người dùng lưới trên portal và giấy ủy nhiệm của họ, được lưu trữ vào cơ sở dữ liệu để sử dụng cho các mục đích chứng thực, thẩm quyền hay khả năng ủy nhiệm, đăng nhập một lần.

GsCredentialContext: ngữ cảnh này mô tả các thông tin của một giấy ủy nhiệm thuộc về một người sử dụng portal, được tạo ra và duy trì bởi dịch vụ quản lý giấy ủy nhiệm CredentialManagerService. Cụ thể, nó cung cấp các API cho phép xem các thông tin về người dùng, thông tin về giấy ủy nhiệm: ngày giờ tạo lập, thời gian sống, tên định danh, còn được kích hoạt hay đã hết thời hạn sử dụng,... và các khả năng để kích hoạt hay không kích hoạt ngữ cảnh này. Mỗi người sử dụng có thể có nhiều giấy ủy nhiệm, hay nói cách khác là họ có nhiều ngữ cảnh của giấy ủy nhiệm, nhưng không có hai ngữ cảnh nào có cùng một tên định danh (DN- distinguished name). Mỗi ngữ cảnh chỉ mô tả một giấy ủy nhiệm tương ứng với một tên định danh duy nhất.

CredentialRetrievalContext: ngữ cảnh này chứa các thông tin về truy xuất giấy ủy nhiệm bởi một người dùng portal, được tạo ra và duy trì bởi dịch vụ truy xuất giấy ủy nhiệm CredentialRetrievalService. Nó chứa các thông tin về tên người dùng, tên giấy chứng nhận, ngày giờ truy xuất, ngày giờ cập nhật, giấy ủy nhiệm này có đăng ký đăng nhập một lần hay không,... Mỗi người dùng có thể có nhiều ngữ cảnh truy xuất giấy ủy nhiệm, hay có thể truy xuất nhiều giấy ủy nhiệm từ kho lưu trữ giấy ủy nhiệm, nhưng không có hai ngữ cảnh truy xuất nào có cùng tên định danh (DN- distinguished name), hay cùng thuộc về một giấy ủy nhiệm.

+ Các portlet bảo mật (SecurityPortlet)

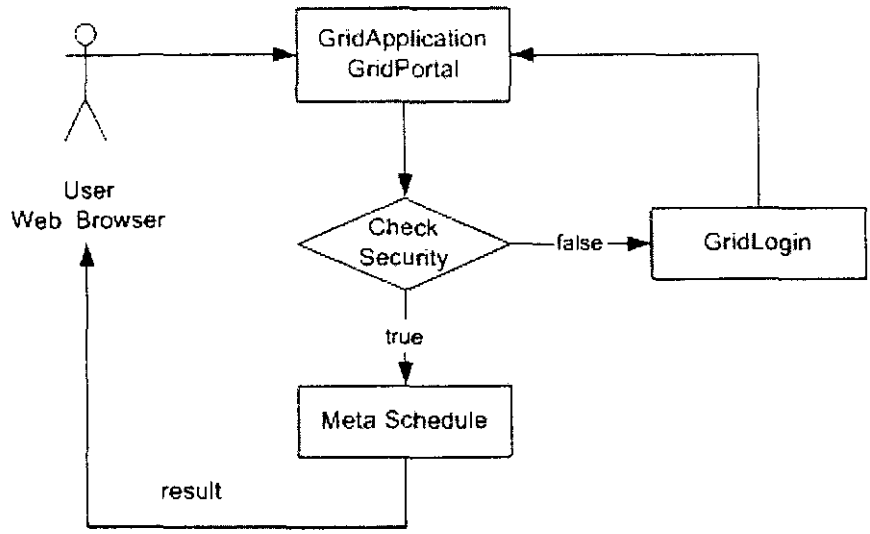
Các portlet này trực tiếp tương tác với người dùng thông qua giao diện Portal. Hiện tại, chúng được phát triển trên chuẩn JSR168 Portlet của Sun, tích hợp với các thư viện thẻ giao diện người dùng của GridSphere (UI tab). Người dùng thông qua đó, có thể sử dụng hầu hết các tính năng mà các dịch vụ bảo mật trên cung cấp.

CredentialManagerPortlet: Portlet này cho phép người dùng có thể xem thông tin về các giấy ủy nhiệm của mình, sửa xóa các thông tin. Ngoài ra, còn có một số tính năng như kích hoạt giấy ủy nhiệm, thay đổi giấy ủy nhiệm mặc định, thay đổi kho lưu trữ ủy nhiệm, các giao diện được thiết kế tiếng Việt sao cho thân thiện và tiện lợi nhất với người dùng.

GridLoginPortlet: Portlet này giúp người dùng đăng nhập vào hệ thống BKGrid 2005, kiểm tra tính hợp lệ trước khi sử dụng các dịch vụ lưới như: người dùng đã có giấy ủy nhiệm lưới hay chưa, giấy ủy nhiệm của người dùng còn hợp lệ hay không. Nó còn cho phép người dùng tùy chọn tính năng đăng nhập một lần, ủy quyền cho Portal.

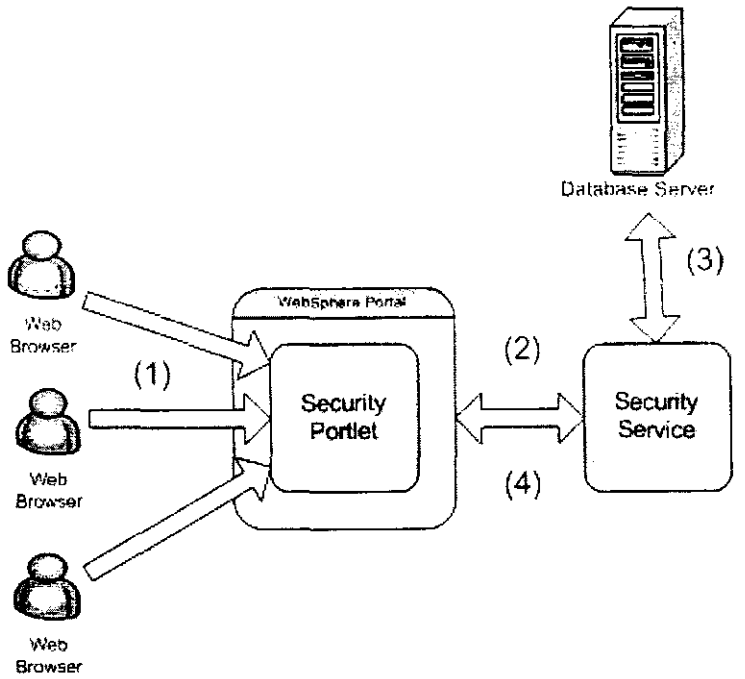
4.5.3. Tích hợp dịch vụ bảo mật trong BKGrid2005

Do dịch vụ bảo mật được thực hiện ở phía trên cùng của hệ thống, các dịch vụ trong hệ thống, nếu hỗ trợ bảo mật thì phải thông qua dịch vụ bảo mật trước tiên. Một ví dụ về tương tác dịch vụ bảo mật với bộ lập lịch trong hệ thống được biểu diễn trong lưu đồ sau:



Hình 4-19: Bộ lập lịch với hỗ trợ bảo mật

Như trong hình vẽ, người dùng qua giao diện Web, có thể truy nhập sử dụng các ứng dụng lưới hoặc GridPortal để đệ trình các công việc. Trước tiên, dịch vụ bảo mật kiểm tra tính hợp lệ của người dùng, nếu hợp lệ thì sẽ chuyển đến cho bộ lập lịch để lựa chọn tài nguyên lưới phù hợp, thực hiện yêu cầu người dùng, sau đó trả lại kết quả. Nếu việc kiểm tra không hợp lệ, người dùng sẽ phải quay lại một màn hình đăng nhập (GridLogin), nhập các thông tin cần thiết để chứng minh là thành viên của lưới, sau đó tiếp tục thực hiện các yêu cầu của mình. Chi tiết về quá trình đăng nhập của người dùng vào hệ thống, dịch vụ bảo mật hỗ trợ người dùng khi ủy quyền lên portal được thể hiện ở hình vẽ dưới:



Hình 4-20: Người dùng uỷ quyền giấy uỷ nhiệm cho portal

1) Người dùng đăng nhập vào portal, sau đó qua giao diện web để gửi tên và mật khẩu của mình.

2) Tên và mật khẩu của người dùng sẽ được portal gửi đến dịch vụ bảo mật.

3) Dịch vụ bảo mật kiểm chứng tên và mật khẩu của người dùng từ máy chủ cơ sở dữ liệu. Nếu hợp lệ, nó sẽ tải giấy ủy nhiệm người dùng cùng các chính sách bảo mật tương ứng về.

4) Bước cuối cùng, dịch vụ bảo mật gửi giấy ủy nhiệm này lên portal để thay mặt cho người dùng, thực hiện các công việc trong lưới.

Với kịch bản này, yêu cầu về cầu kết nối giữa môi trường lưới và môi trường web đã được giải quyết, người dùng thông qua portal vẫn có đầy đủ quyền hạn của mình đối với môi trường lưới, cho dù họ ở bất cứ nơi nào.

Dịch vụ bảo mật là thành phần đầu tiên và cơ bản của hệ thống, thông qua nó người dùng mới có thể triệu gọi các dịch vụ khác và truy nhập vào tài nguyên lưới, giống như một nhà cung cấp các giấy visa để cho phép thông hành qua các nước khác nhau. Do kiến trúc mở và khả năng kế thừa các nghiệp vụ, các dịch vụ khác có thể thông qua dịch vụ bảo mật để kiểm tra tính hợp lệ của người dùng trong hệ thống lưới, biết được họ là ai và có quyền hạn gì, cũng như ghi lại các thông tin người dùng để sử dụng cho mục đích thanh toán cũng như bảo mật sau này.

Chương 5: Dịch vụ thông tin

Dịch vụ thông tin hệ thống là một thành phần quan trọng trong hệ thống giúp cung cấp đầy đủ thông tin về các tài nguyên và dịch vụ trong hệ thống để giúp các bộ lập lịch và phân phối tài nguyên có thể hoạt động hiệu quả. Dịch vụ thông tin hệ thống đóng vai trò như một giao diện cho phép người dùng hay các dịch vụ khác khai thác, theo dõi và sử dụng hiệu quả các nguồn tài nguyên. Trong bất kì môi trường lưới nào, thông tin về tài nguyên luôn dao động, phụ thuộc vào khả năng xử lí và chia sẻ dữ liệu. Khi dữ liệu được giải phóng, chúng có thể cập nhật trạng thái của chúng trong dịch vụ thông tin. Các phần mềm phía máy khách, bộ môi giới tài nguyên và các bộ quản lí tài nguyên có thể sử dụng các thông tin này để phục vụ các mục đích riêng của chúng.

Dịch vụ thông tin là một trong các thành phần cốt lõi của tính toán lưới. Một mặt, dịch vụ thông tin trả lời trực tiếp các câu truy vấn từ bất kì ứng dụng nào, cung cấp các thông tin cho bộ định vị tài nguyên mặt khác là nguồn thông tin không thể thiếu được trong các bộ môi giới tài nguyên. Trong các hệ thống lưới thực tế, việc theo dõi trạng thái lưới là công việc thường xuyên của những người quản trị lưới điều này hoàn toàn có thể được trợ giúp bởi các dịch vụ thông tin từ đó người quản trị lưới sẽ có những chính sách nhằm điều chỉnh các tài nguyên trong lưới.

5.1. Tổng quan về dịch vụ thông tin trong tính toán lưới

5.1.1. Khái niệm chung về dịch vụ thông tin

Dịch vụ thông tin đóng vai trò như một giao diện cho phép người dùng hay các dịch vụ khác khai thác và sử dụng hiệu quả các nguồn tài nguyên phân tán. Trong bất kì môi trường lưới nào, thông tin về tài nguyên luôn dao động. Khi tài nguyên được giải phóng, trạng thái của chúng có thể được cập nhật trong dịch vụ thông tin. Client, bộ môi giới tài nguyên và các bộ quản lí tài nguyên có thể sử dụng các thông tin này để phục vụ các mục đích riêng của chúng.

Tóm lại, dịch vụ thông tin là dịch vụ cung cấp các thông tin về tài nguyên cho người dùng và cho một vài ứng dụng trong lưới.

5.1.2. Vai trò và nhiệm vụ của dịch vụ thông tin

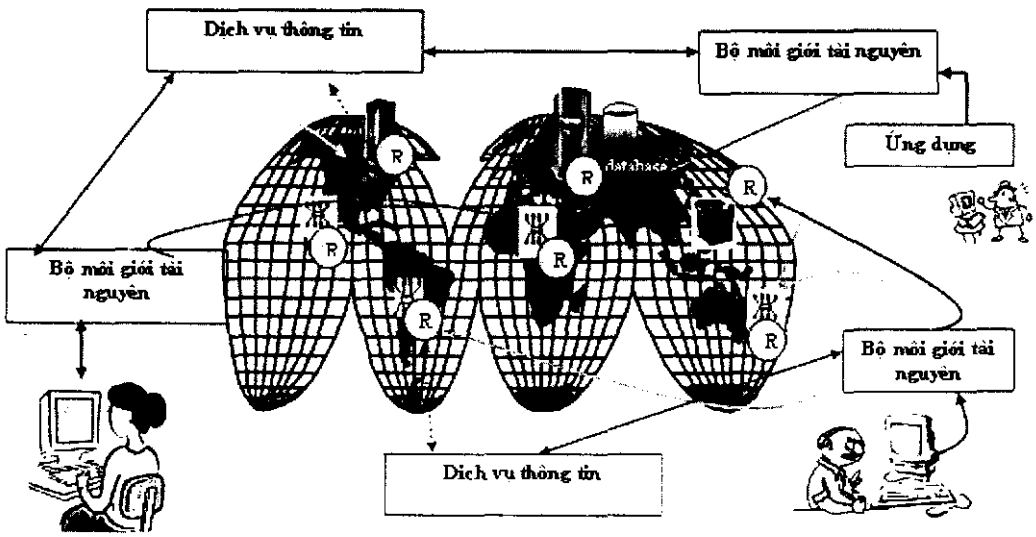
5.1.2.1. Vai trò

Trong lưới, có nhiều dịch vụ, mỗi dịch vụ cung cấp các chức năng khác nhau. Như đã nói ở trên, dịch vụ thông tin là dịch vụ lưới dùng để khai thác và theo dõi thông tin về các tài nguyên trên lưới, từ đó cung cấp thông tin cho người dùng hoặc cho các dịch vụ khác, ví dụ

như dịch vụ lập lịch (để dịch vụ lập lịch lựa chọn giải thuật cho phù hợp với công việc), cung cấp thông tin cho bộ môi giới tài nguyên.

Các công nghệ lưới cho phép chia sẻ các tài nguyên trên phạm vi lớn. Trong điều kiện này, việc khai thác và theo dõi các tài nguyên, các dịch vụ và các nút tính toán có thể là một thử thách thú vị. Do đó, có thể coi dịch vụ thông tin là thành phần cốt lõi của lưới. Nó tập trung vào việc cung cấp các thông tin giá trị tương ứng với các tài nguyên trong lưới. Dịch vụ này phụ thuộc hoàn toàn vào các bộ cung cấp thông tin như sự sẵn có của các tài nguyên, dung lượng của tài nguyên,...Dịch vụ thông tin cho phép bộ môi giới lựa chọn tài nguyên một cách hiệu quả cho các công việc đặc biệt liên quan đến tính toán lưới.

Dịch vụ thông tin là một trong các thành phần không thể thiếu của tính toán lưới. Một mặt, nó trả lời trực tiếp các câu truy vấn từ bất kì ứng dụng nào, cung cấp các thông tin cho bộ định vị tài nguyên. Mặt khác, dịch vụ này là nguồn thông tin quan trọng trong các bộ môi giới tài nguyên. Xét trong môi trường lưới, các ứng dụng lưới được đệ trình tới các nút trong lưới thông qua bộ môi giới tài nguyên, bộ môi giới thực hiện các giải thuật lựa chọn tài nguyên của mình theo các tiêu chí khác nhau và dựa trên thông tin về tài nguyên của hệ thống sẵn có. Như vậy, có thể thấy, đầu vào của bộ môi giới tài nguyên là các yêu cầu tương ứng với các ứng dụng lưới cùng với các thông tin tài nguyên được cập nhật liên tục, bộ phận chịu trách nhiệm cung cấp đầy đủ và chính xác các thông tin tài nguyên hệ thống cho các bộ môi giới tài nguyên đó chính là dịch vụ thông tin. Ngoài ra, dịch vụ thông tin còn là công cụ hữu ích cho người quản trị lưới theo dõi, đánh giá hiệu năng của hệ thống. Trong các hệ thống lưới thực tế, việc theo dõi trạng thái lưới là công việc thường xuyên của những người quản trị lưới. Điều này hoàn toàn có thể được trợ giúp bởi dịch vụ thông tin, từ đó người quản trị lưới sẽ có những chính sách nhằm điều chỉnh các tài nguyên trong lưới. Trong một số hệ thống, người ta còn tích hợp việc theo dõi thông tin về tài nguyên hệ thống với việc theo dõi trạng thái các tiến trình đang được thực hiện.



Hình 5-1: Vị trí của dịch vụ thông tin trong tính toán lưới

Công nghệ tính toán lưới cho phép chia sẻ và đồng sử dụng các tài nguyên mạng trên diện rộng. Trong nhiều trường hợp, người dùng có ít hiểu biết về các tài nguyên được xây dựng bởi các thành viên trong các tổ chức ảo, đó sẽ là một trở ngại đối với họ. Vì lí do này, dịch vụ thông tin được thiết kế để hỗ trợ công việc khai thác và theo dõi các dịch vụ, các tài nguyên và các thực thể khác trong hệ thống lưới. Các ví dụ sau chứng minh các ứng dụng tin tưởng vào dịch vụ thông tin, rất nhiều nguồn thông tin và các phương thức quản lí và truy cập thông tin liên quan đến các ứng dụng này.

Dịch vụ khai thác - service discovery ghi lại các thuộc tính của các dịch vụ là thành viên của lưới. Ở đây, các nguồn thông tin là tương đối tĩnh và bản thân các thông tin có mối liên hệ với độ sẵn dùng (dung lượng còn trống).

Bộ siêu lập lịch - superscheduler tìm các yêu cầu tính toán phù hợp với máy tính tốt nhất sẵn có trong lưới chứa nhiều máy tính đầu cuối. Từ "tốt nhất " ở đây có thể được đánh giá theo các tiêu chí sau: kiến trúc, các phần mềm được cài đặt, hiệu năng,...Ở đây, nguồn thông tin là các máy tính, và thông tin có thể bao gồm cả thông tin tương đối tĩnh (như cấu hình hệ thống: kiến trúc, phiên bản HĐH, chính sách truy cập) và các thông tin động như tải, dung lượng còn trống.

Dịch vụ lựa chọn bản sao – replica selection trả về bản sao tốt nhất cho các yêu cầu tìm kiếm bản sao của một file nào đó trên các hệ thống lưu trữ. Ở đây, các hệ thống lưu trữ không chỉ là các máy tính. Và nguồn thông tin gồm nhiều thành phần: cả ứng dụng và môi trờng mà ứng dụng đó đang chạy.

Công cụ chẩn đoán hiệu năng - performance diagnosis tool, được triệu gọi bởi một người dùng khi có một hoạt động không có quy tắc bị phát hiện, khai thác nguồn thông tin nào có liên quan tới một ứng dụng và các tài nguyên của ứng dụng đó (ví dụ: các nguồn thông tin có tính lịch sử) và truy cập vào các nguồn thông tin này để chẩn đoán hiệu năng thấp.

Các ví dụ khác nhau ở trên chứng minh ứng dụng rộng rãi của dịch vụ thông tin trong môi trường lưới.

5.1.2.2. Nhiệm vụ

Với vai trò quan trọng như đã nói ở trên, dịch vụ thông tin có các nhiệm vụ sau:

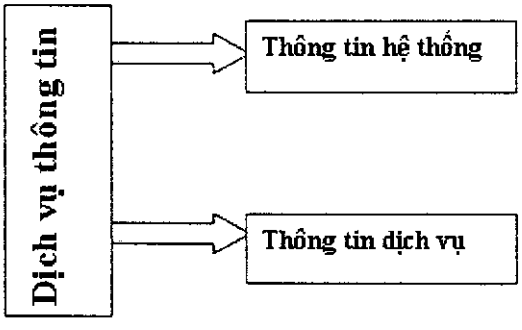
- Truy cập các thông tin một cách đồng nhất và linh hoạt;
- Truy cập hiệu quả các thông tin động;
- Truy cập nhiều nguồn thông tin;
- Tạo và quản lí thông tin một cách động thông qua các bộ cung cấp thông tin;
- Tập hợp các dữ liệu dịch vụ từ các instance khác nhau;
- Đăng kí các instance dịch vụ lưới;

5.1.3. Yêu cầu của dịch vụ thông tin

Môi trường lưới có một số các thuộc tính cần được tính đến khi thiết kế các hệ thống thông tin. Đó là các nguồn thông tin được phân tán theo nhu cầu, có thể có lỗi, và được truy cập bởi các mạng không tin cậy. Số lượng các bộ cung cấp thông tin có thể rất lớn, kiểu thông tin cũng như cách sử dụng thông tin có thể biến đổi. Như vậy, chúng ta khó có thể cung cấp cho người dùng các thông tin chính xác về một tài nguyên nào đó (vì bất kì thông tin động nào cũng có thể là "cũ" với người dùng). Tóm lại, sự phân tán của các tài nguyên tin mang lại hạn chế trong việc thu thập các loại thông tin. Vì vậy, dịch vụ thông tin lưới nên tập trung vào hiệu quả thu thập các thông tin từ các tài nguyên phân tán.

Dịch vụ thông tin cần cung cấp:

- Các thông tin về tài nguyên hệ thống
- Các thông tin về các dịch vụ mà tài nguyên đó cung cấp



Hình 5-2: Các loại thông tin đầu ra của dịch vụ thông tin

Khai thác tài nguyên:

Thông tin về các tài nguyên trong lưới là vô cùng cần thiết cho cả người dùng và các ứng dụng trong lưới. Đối với các tài nguyên tính toán, thông tin có thể là các chi tiết phần cứng và phần mềm.

Các bộ cung cấp thông tin là các chương trình cung cấp các thông tin về trạng thái tài nguyên. Ví dụ các thông tin sau:

Thông tin tĩnh về host:địa chỉ IP, tên và phiên bản hệ điều hành, nhà cung cấp, phiên bản, tốc độ bộ vi xử lí, số lượng bộ vi xử lí, tổng dung lượng bộ nhớ vật lí, tổng dung lượng bộ nhớ ảo,...

Thông tin động về host: Thông tin về tải CPU,...

Thông tin về hệ thống lưu trữ: tổng dung lượng đĩa, dung lượng đĩa còn trống,...

Thông tin về tài nguyên tính toán:nhà cung cấp CPU, loại CPU, tốc độ,....loại bộ nhớ, dung lượng, tốc độ,...

Thông tin về tài nguyên mạng: băng thông mạng, độ trễ Latency, topology...

Khai thác các dịch vụ lưới

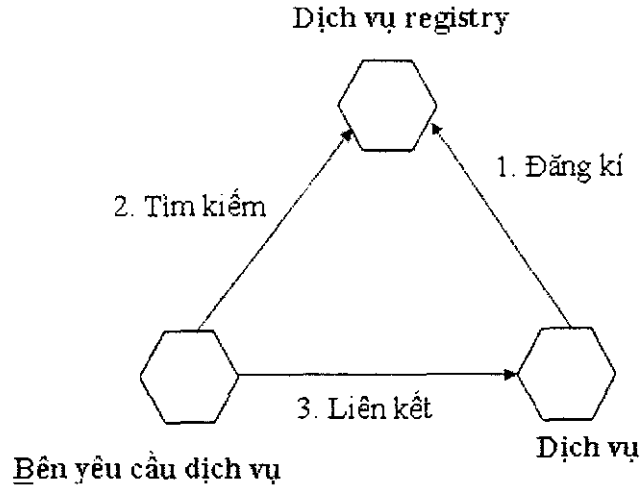
Trong kiến trúc hướng dịch vụ, các dịch vụ được tìm thấy trong registry. Registry là một dịch vụ nhóm trong mô hình OGSI.

Dịch vụ nhóm là dịch vụ biểu diễn một tập các dịch vụ khác.Và dịch vụ thông tin sẽ tìm kiếm các dịch vụ trong Registry để lấy các SDE của các dịch vụ đó và cung cấp cho người dùng hoặc cho các ứng dụng khác trên lưới.

Ví dụ về dịch vụ nhóm trong GT3

Container registry – danh sách các dịch vụ địa phương

MDS3 Index service – tập hợp các dịch vụ đã được đánh chỉ mục



Hình 5-3: Cơ chế khai thác các dịch vụ lưới

Các dịch vụ sẽ đăng kí với dịch vụ registry, khi bên yêu cầu muốn biết thông tin về dịch vụ nào đó thì nó phải tìm kiếm dịch vụ đó trong registry.

5.1.4. Kiến trúc chung của dịch vụ thông tin

Sau đây là kiến trúc một dịch vụ thông tin đáp ứng các yêu cầu của môi trường lưới. Kiến trúc này bao gồm hai yếu tố cơ bản:

Các bộ cung cấp thông tin (tập hợp các thông tin phân tán) cung cấp khả năng truy cập vào thông tin của từng thực thể riêng lẻ. Thông tin là một kiểu dữ liệu có cấu trúc, một thực thể được mô tả bởi một tập các đối tượng với nhiều cặp thuộc tính – giá trị. Các bộ cung cấp thông tin cung cấp khả năng truy cập vào các thông tin động.

Các dịch vụ tầng cao hơn làm nhiệm vụ thu thập, quản lí, đánh chỉ mục và trả về thông tin đã được cung cấp bởi các bộ cung cấp thông tin. Các dịch vụ ở tầng cao có thể sử dụng các thông tin này để phục vụ cho việc môi giới theo dõi thông tin. Việc tương tác giữa các dịch vụ tầng cao (hay người dùng) và các bộ cung cấp được định nghĩa qua 2 giao thức: giao thức đăng kí và giao thức truy vấn.

Giao thức đăng kí (registration protocol): để xác định các thực thể tham gia vào dịch vụ thông tin

Giao thức truy vấn (enquiry protocol): để thu thập thông tin về các thực thể thông qua các câu truy vấn.

Các giao thức này cũng có thể được kết hợp với các giao thức lưới khác để xây dựng nên các dịch vụ bổ sung ở tầng cao ví dụ như: môi giới, theo dõi, phát hiện lỗi, và xử lí lỗi.

Tóm lại, một bộ cung cấp sử dụng giao thức đăng kí để biết được sự tồn tại của các dịch vụ ở tầng trên

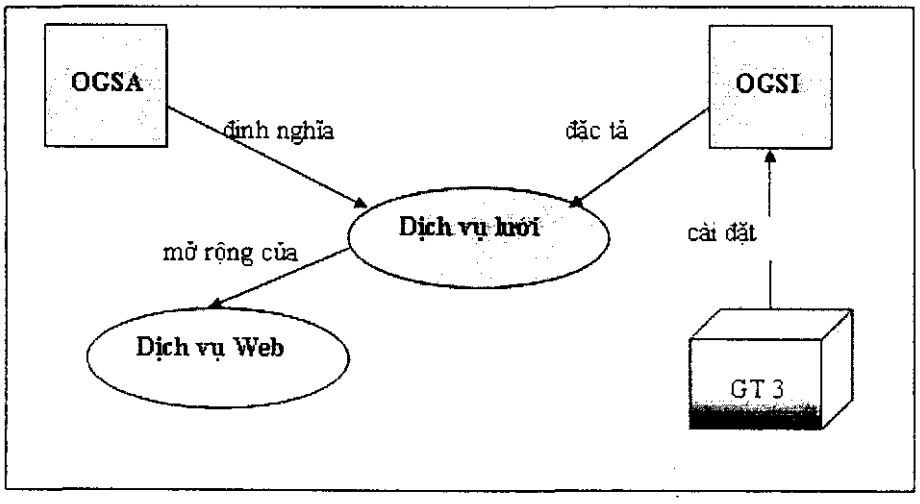
Một dịch vụ ở tầng trên sử dụng giao thức truy vấn để lây các thông tin từ các bộ cung cấp.

5.2. Dịch vụ thông tin trên nền GT3

5.2.1. Dịch vụ và dữ liệu dịch vụ

5.2.1.1. Dịch vụ

Xem xét kiến trúc hướng dịch vụ sau:



Hình 5-4: Dịch vụ lưới trong kiến trúc hướng dịch vụ

- OGSA (Open Grid Services Architecture): nhằm mục đích định nghĩa ra một chuẩn kiến trúc mới cho các ứng dụng chạy trên lưới. OGSA định nghĩa dịch vụ lưới là gì, chúng có khả năng gì, và dựa trên nền công nghệ nào. Nhưng OGSA không đưa ra một đặc tả chi tiết và nặng tính kỹ thuật cần để triển khai một dịch vụ lưới.
- OGSI (Open Grid Services Infrastructure): là đặc tả nặng tính kỹ thuật cho các khái niệm được đưa ra trong OGSA, trong đó có các dịch vụ lưới.
- Globus Toolkit (GT) là phần triển khai của OGSI, nó rất hữu dụng trong việc triển khai những gì được đặc tả bởi OGSI (và do đó, triển khai những gì được định nghĩa bởi OGSA).
- Dịch vụ lưới dựa trên nền tảng của dịch vụ Web.

OGSI là một kiến trúc hướng dịch vụ. Trong kiến trúc này, mọi tài nguyên đều được coi như dịch vụ. Ví dụ: các tài nguyên tính toán, tài nguyên lưu trữ, mạng, các chương trình, cơ sở dữ liệu, v.v đều được coi như những dịch vụ. Mục đích của kiến trúc OGSI là tạo ra một nền

của tính toán lưới hỗ trợ việc tạo, duy trì, quản lí các dịch vụ trong môi trường lưới tính toán lưới dựa trên nền công nghệ dịch vụ web.

Các dịch vụ có thể được thực thi ở một nút lưới nào đó trong hệ thống, cũng có thể trên nhiều nút lưới hay chỉ đơn giản là một chương trình tính toán đơn thuần trên chính máy cục bộ, tuy nhiên với người sử dụng hay cả với người lập trình ứng dụng ở mức cao nó đều chỉ là một giao diện chuẩn cho dịch vụ được OGSA định nghĩa và xây dựng như một chuẩn với bất kì một dịch vụ nào.

Các dịch vụ trong kiến trúc này có thể là tập các dịch vụ bền vững hoặc các dịch vụ tạm thời. Có thể có một hay nhiều thực thể cho một dịch vụ lưới. Mỗi một thực thể của dịch vụ lưới có một tập các dữ liệu dịch vụ gắn liền với nó, và dữ liệu này được biểu diễn dưới dạng chuẩn là Service Data Element (SDE). Và các thao tác của dịch vụ chỉ mục chủ yếu là thực hiện trên SDE.Ví dụ: tải CPU của một tài nguyên tính toán được biểu diễn trong một SDE bởi một dịch vụ tài nguyên tính toán.

Dịch vụ lưới là mở rộng của dịch vụ web. Cũng như dịch vụ web, một yếu tố quan trọng của dịch vụ lưới là máy chạy các tiến trình client và máy chạy các tiến trình server có thể là hai máy khác nhau, nói cách khác, dịch vụ lưới có thể là một hệ thống phân tán.

Các tương tác giữa dịch vụ lưới và client có thể là một - một hoặc **một** - nhiều. Một - một tức là mỗi client có một thực thể dịch vụ lưới riêng, và như vậy sau khi tương tác giữa client và dịch vu hoàn thành, thực thể sẽ bị huỷ bỏ. Một - nhiều tức là một dịch vụ chỉ có một thực thể và thông tin của dịch vụ có thể được dùng cho một số client.

Các dịch vụ lưới sử dụng factory để tạo các thực thể cho dịch vụ. Mỗi dịch vụ lưới có một factory để tạo và quản lí các thực thể của dịch vụ đó.

5.2.1.2. Dữ liệu dịch vụ

- Định nghĩa:

Dữ liệu dịch vụ có lẽ là một trong những khái niệm quan trọng nhất của dịch vụ lưới. Nó là một tập hợp có cấu trúc các thông tin liên quan đến thực thể của dịch vụ lưới để hiển thị các dữ liệu dịch vụ của thực thể cho các bên yêu cầu dịch vụ, và dữ liệu dịch vụ đó được biểu diễn dưới dạng chuẩn Service Data Element (SDE).

- Dữ liệu dịch vụ có hai loại:

+ State information (thông tin trạng thái): cung cấp thông tin về trạng thái hiện tại của dịch vụ. Ví dụ: kết quả thao tác, các kết quả trung gian...

+ Service metadata (siêu dữ liệu dịch vụ): thông tin về chính dịch vụ đó. Ví dụ: dữ liệu hệ thống, các giao diện hỗ trợ, chi phí sử dụng dịch vụ.

Mọi thực thể dịch vụ lưới theo mặc định sẽ có một vài SDE chuẩn. Mỗi SDE phải có một tên duy nhất và chứa các cặp tên - giá trị. Các SDE có thể là tĩnh hoặc động. Các dữ liệu dịch vụ động được thêm vào thực thể của dịch vụ. Để sử dụng dữ liệu dịch vụ động, các client phải lấy được danh sách các SDE tại thời gian chạy.

Ví dụ về SDE:

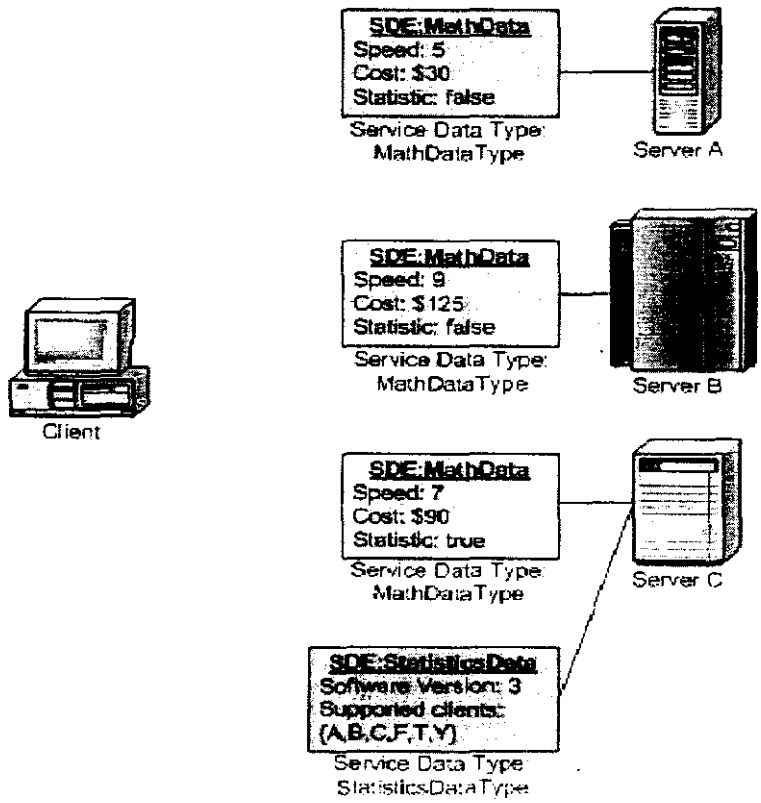
Giả sử ta có dịch vụ MathService, một dịch vụ thực hiện các phép toán cơ bản: cộng, trừ, nhân, chia. Khi client yêu cầu dịch vụ MathService, và giả sử có 3 server cung cấp dịch vụ này. Client phải chọn một trong số 3 server này sao cho thoả mãn tối đa các tiêu chí mà client đưa ra.

Trong hình 5-5, ta có 3 dịch vụ MathService.

Hai dịch vụ đầu, mỗi dịch vụ có một SDE MathData. Chú ý rằng cả hai SDE đều chứa cùng một loại thông tin (MathDataType, chứa speed, cost, and statistics).

Dịch vụ thứ 3 có hai SDE: MathData và StatisticsData. Chú ý rằng SDE StatisticsData có kiểu thông tin khác (StatisticsDataType, chứa phiên bản phần mềm, và các client hỗ trợ).

Giả sử, client của chúng ta cần một dịch vụ MathService để biểu diễn một vài phép toán, và rất cần dịch vụ đó tính càng nhanh càng tốt. Client sẽ tìm SDE MathData của mỗi dịch vụ, sau đó dựa vào giá trị của thuộc tính speed, và chọn dịch vụ có tốc độ cao nhất. Nếu client cần phân tích trạng thái thì client sẽ chọn dịch vụ ở server C (vì đây là dịch vụ duy nhất có thuộc tính statistics được thiết lập là true). Sau đó, client sẽ tìm SDE StatisticsData để đảm bảo các thông tin khác như phiên bản phần mềm và các client hỗ trợ cũng thoả mãn.



Hình 5-5: Các thành phần trong SDE

Ngoài các dữ liệu dịch vụ chung mà dịch vụ nào cũng có, ta còn có thể bổ sung các dữ liệu dịch vụ do người dùng tự định nghĩa. Các dữ liệu dịch vụ phụ thuộc vào giao diện cài đặt

của dịch vụ. Dữ liệu dịch vụ chung mô tả các đặc trưng của dịch vụ. Ví dụ như GSH mô tả URL của thực thể dịch vụ.

Các SDE hay dùng:

gridServiceHandle: SDE đa trị, chứa GSH của dịch vụ lưới

factoryLocator: SDE đơn trị, chỉ ra nơi định vị của factory đã tạo ra dịch vụ này. Nếu dịch vụ không được tạo ra bởi một factory nào thì giá trị của SDE này sẽ là null.

terminationTime: SDE đơn trị, chứa thông tin về thời gian kết thúc vủa dịch vụ

serviceDataNames: SDE đa trị, chứa tên của các SDE trong dịch vụ

interfaces: SDE đa trị, chứa tên của các giao diện được cài đặt trong dịch vụ.

Ghi chu:

SDE đơn trị: Là SDE chỉ có một giá trị duy nhất

SDE đa trị: Là SDE có thể có một hoặc nhiều giá trị

Các thuộc tính của SDE:

minOccurs: Số lượng giá trị tối thiểu mà SDE có thể có

maxOccurs: Số lượng giá trị tối đa mà SDE có thể có. Giá trị của thuộc tính này là không giới hạn, tức là một mảng không có giới hạn về kích thước

modifiable: Là true hoặc false. Được xác định khi giá trị của SDE bị thay đổi bởi một client

nillable: Là true hoặc false. Được xác định khi giá trị của SDE là NULL

mutability: Thuộc tính này có thể có các giá trị sau:

+ *static*: Giá trị của SDE được cung cấp trong file mô tả GWSDL.

+ *constant*: Giá trị của SDE được thiết lập khi dịch vụ lưới được tạo và sau đó sẽ không bị thay đổi

+ *extendable*: Các phần tử mới có thể được thêm vào SDE mà không thể xoá đi được

+ *mutable*: Các phần tử mới có thể thêm vào hoặc xoá khỏi các SDE

Trong ví dụ của chúng ta, SDE MathData sẽ chỉ có một và chỉ một giá trị, và cũng sẽ bị thay đổi trong suốt thời gian tồn tại của dịch vụ lưới, nhưng không thể là NULL và cũng không thể bị sửa đổi bởi client.

Sau đây chúng ta sẽ xem một vài SDE và các giá trị của chúng:

• interface

Là QNames của tất cả portType trong phần định nghĩa giao diện của thực thể dịch vụ.

```
<sd:serviceData name="interface"
type="xsd:QName"
minOccurs="1" maxOccurs="unbounded"
mutability="constant"
modifiable="false"
```

```
nillable="false"/>
```

• serviceDataName

Là một tập hợp QName của phần tử dữ liệu dịch vụ. Tập này phải chứa tất cả các QName của tất cả các phần tử dữ liệu dịch vụ được khai báo trong file WSDL.

```
<sd:serviceData name="serviceDataName"
type="xsd:QName"
minOccurs="0" maxOccurs="unbounded"
mutability="mutable"
modifiable="false"
nillable="false"/>
```

• factoryLocator

Là dịch vụ trỏ tới factory tạo ra thực thể của nó. Nếu thực thể không được tạo ra bởi factory, giá trị của factoryLocator sẽ là xsi:nil. Locator phải tham chiếu đến cùng một factory trong suốt thời gian sống của dịch vụ lưới, mặc dù GSH hoặc GSR của thực thể có thể thay đổi.

```
<sd:serviceData name="factoryLocator"
type="ogsi:LocatorType"
minOccurs="1" maxOccurs="1"
mutability="mutable"
modifiable="false"
nillable="true"/>
```

• gridServiceHandle

Giá trị có thể là Zero hoặc các GSH của thực thể dịch vụ lưới. Có thể có những GSH của thực thể không xuất hiện trong tập này.

```
<sd:serviceData name="gridServiceHandle"
type="ogsi:HandleType"
minOccurs="0" maxOccurs="unbounded"
mutability="extendable"
modifiable="false"
nillable="false"/>
```

• gridServiceReference

Chứa một hoặc nhiều GSH của dịch vụ. Trong đó, phần tử giá trị dữ liệu dịch vụ phải là biểu diễn WSDL của GSR. Các phần tử giá trị dữ liệu dịch vụ khác có thể biểu diễn các dạng khác của GSR.

```
<sd:serviceData name="gridServiceReference"
type="ogsi:ReferenceType"
minOccurs="1" maxOccurs="unbounded"
mutability="mutable"
modifiable="false"
nillable="false"/>
```

- findServiceDataExtensibility

```
<sd:serviceData name="findServiceDataExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>
```

- setServiceDataExtensibility

```
<sd:serviceData name="setServiceDataExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="2" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>
```

- terminationTime: Thời gian kết thúc của dịch vụ.

```
<sd:serviceData name="terminationTime"
type="ogsi:TerminationTimeType"
minOccurs="1" maxOccurs="1"
mutability="mutable"
modifiable="false"
nillable="false"/>
```

Thêm vào đó, GridService portType định nghĩa tập khởi tạo các phần tử giá trị dữ liệu dịch vụ.

```
<sd:staticServiceDataValues>
<ogsi:findServiceDataExtensibility
inputElement="ogsi:queryByServiceDataNames" />
<ogsi:setServiceDataExtensibility
inputElement="ogsi:setByServiceDataNames" />
<ogsi:setServiceDataExtensibility>
inputElement="ogsi:deleteByServiceDataNames" />
</sd:staticServiceDataValues>
```

5.2.1.3. Truy cập dịch vụ

Để truy cập và lấy thông tin về dịch vụ lưới, ngoài khái niệm SDE còn có hai thuật ngữ quan trọng khác là GSH (Grid Service Handle) và GSR (Grid Service Reference).

- (GSH) Grid Service Handle

GSH là địa chỉ xác định duy nhất một Grid Service. Nó giống như địa chỉ URL của các trang web.

Theo định nghĩa trong OGSI, "Grid service URI" được gọi là Grid Service Handle (GSH). GSH được trả về từ thao tác tạo dịch vụ của Factory. GSH được sử dụng để đặt tên cho một thực thể của dịch vụ.

Mỗi GSH phải là duy nhất và trỏ tới một thực thể của dịch vụ. Không thể có 2 thực thể có cùng một GSH. Một GSH có thể được coi như là một con trỏ trỏ tới một thực thể cụ thể. GSH không cung cấp các thông tin để cho phép client truy cập vào dịch vụ, cho nên để lấy được các thông tin về dịch vụ, GSH cần được ánh xạ sang GSR.

- GSR (Grid Service Reference)

GSR là một file có mục đích cung cấp những thông tin cần thiết để chương trình client biết cách giao tiếp với Web Service. GSR có nhiều dạng khác nhau, nhưng thường người ta sử dụng định dạng SOAP.

GSR mô tả cách một client có thể giao tiếp với một thực thể của một dịch vụ lưới. Trong khi GSH chỉ biểu diễn tên thì GSR bao gồm cả các thông tin phục vụ cho các giao thức vận tải và định dạng mã hóa dữ liệu.

Để đáp ứng nhu cầu liên lạc với dịch vụ, GSH phải được chuyển thành một GSR.

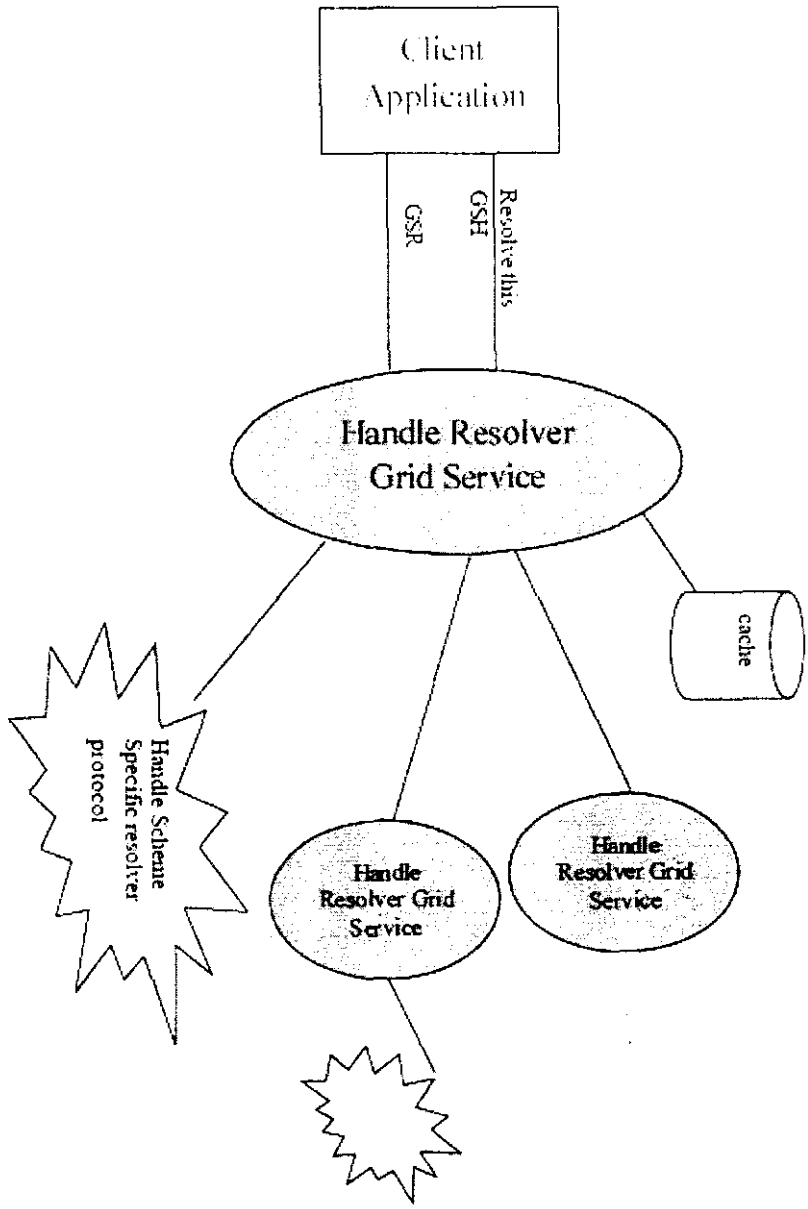
GSR chứa tất cả các thông tin mà client cần để liên lạc với dịch vụ. GSR không phải là con trỏ bền vững trỏ tới một thực thể, GSR có thể trở thành không hợp lệ vì nhiều lý do (ví dụ, thực thể của dịch vụ lưới bị chuyển tới một server khác). Khi một GSR bị phát hiện là không

hợp lệ mà thực thể vẫn tồn tại thì một client có thể lấy một GSR mới bằng cách sử dụng GSH của thực thể đó.

Tóm lại, GSH trỏ tới một dịch vụ lưới, còn GSR xác định cách liên lạc với dịch vụ lưới. GSH phải được chuyển sang GSR (Grid Service Reference) để client có thể sử dụng dịch vụ, vì thông tin chứa trong GSH là chưa đủ. [7]

- Cơ chế chuyển từ GSH sang GSR:

GT3 cung cấp một cơ chế tên là HandleResolver để hỗ trợ client trong việc chuyển đổi một GSH sang một GSR. GSR có thể có nhiều dạng, nhưng vì GT3 sử dụng một kết nối SOAP để liên lạc với một dịch vụ nên GSR ở dạng WSDL (file WSDL mô tả giao diện Web service: dịch vụ đó cung cấp những thao tác nào, thuộc tính nào...).



Hình 5-6: Cơ chế của Hand Resolver

Khi ứng dụng truy cập vào dịch vụ thông qua GSH của dịch vụ đó, Handle Resolver Grid Service sẽ sử dụng giao thức Handle Scheme Specific resolver Protocol để tìm ra GSH tương ứng với GSR đó và trả về cho ứng dụng.

5.2.2. MDS (Dịch vụ khai thác và theo dõi thông tin)

5.2.2.1. Tổng quan

Thành phần Monitoring and Discovery Service 3 (MDS3) của Globus Toolkit cung cấp một giao diện với khả năng theo dõi thông tin cho các ứng dụng lưới khác nhau khi chúng sử

dụng các giao thức đã được cung cấp trong MDS. Nhờ đó, người dùng và ngay cả các công cụ lưới có thể có những lựa chọn tài nguyên tối ưu nhất cho các ứng dụng của họ.

MDS sử dụng một framework để quản lí các thông tin tĩnh và động về trạng thái của lưới tính toán và các thành phần của lưới: mạng, các nút máy tính, các hệ thống lưu trữ, và các công cụ.

- Khả năng

MDS có khả năng:

Đăng kí động thông qua cơ chế đăng kí - OGSI ServiceGroupRegistration

Truy vấn tài nguyên thông qua giao thức Info. Protocol

Sử dụng các cơ chế truy cập dữ liệu dịch vụ OGSI

Có khả năng truy vấn trực tiếp từ các tài nguyên

Truy cập tĩnh và động vào thông tin về các thành phần hệ thống.

Truy cập thông tin một cách linh hoạt, đồng nhất

Truy cập nhiều nguồn thông tin

Ta có thể sử dụng MDS để trả lời các câu hỏi sau về trạng thái của lưới:

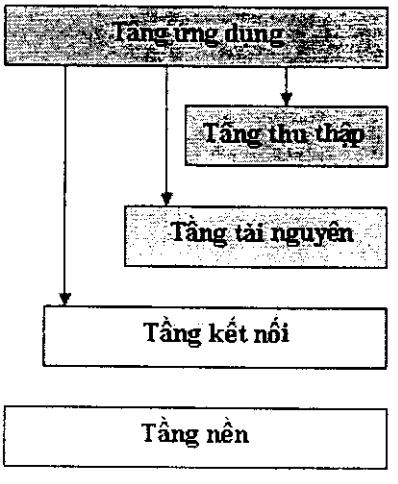
Những tài nguyên nào đang sẵn dùng?

Trạng thái của lưới tính toán là gì?

Cách mà các ứng dụng được tối ưu dựa trên cấu hình của hệ thống đơn giản?

- Đặc điểm

Trước hết ta tìm hiểu kiến trúc phân tầng của lưới:



Hình 5-7: Kiến trúc phân cấp giao thức lưới

Tầng nền (Fabric layer)

Tầng nền trong lưới cung cấp các tài nguyên được gián tiếp truy nhập chung bởi các giao thức lưới; ví dụ như: các tài nguyên tính toán, các hệ thống lưu trữ, các bảng danh mục liệt kê, các tài nguyên mạng và các cảm biến. Mỗi tài nguyên có thể là một thực thể logic như là hệ thống file, các cụm máy tính, hay là một máy tính phân tán đem góp chung. Trong những

trường hợp này, việc thực thi trên tài nguyên có thể bao gồm những giao thức trong, nhưng những điều này không được xem xét trong kiến trúc lưới.

Các thành phần của tầng nền thực hiện các thao tác trên các tài nguyên địa phương cụ thể (vật lý hoặc logic) tương ứng với các thao tác chia sẻ ở các mức cao hơn. Các chức năng thực hiện tại tầng nền một mặt phụ thuộc vào nhau một cách chặt chẽ và tinh tế, mặt khác thì hỗ trợ các thao tác chia sẻ. Các chức năng của tầng nền càng tốt thì càng cho phép tạo nên nhiều thao tác chia sẻ tinh vi; tại cùng một thời điểm, nếu chúng ta đặt ra ít yêu cầu cho các phần tử Fabric thì quá trình triển khai cơ sở hạ tầng của lưới sẽ đơn giản đi.

Tầng kết nối (Connectivity layer)

Tầng này xác định các giao thức lõi về kết nối và chứng thực cho các giao dịch trên mạng. Các giao thức kết nối cho phép trao đổi dữ liệu giữa các tài nguyên của tầng nền. Giao thức chứng thực được xây dựng trên các dịch vụ kết nối để cung cấp các cơ chế bảo mật an toàn trong việc thẩm tra người dùng và tài nguyên.

Tầng tài nguyên (Resource layer)

Tầng này dựa trên các giao thức truyền thông và bảo mật của tầng kết nối để xây dựng các giao thức dùng trong việc quản lý tài nguyên như thỏa thuận an toàn, khởi tạo, theo dõi, điều khiển các chi phí trong các hoạt động chia sẻ. Các giao thức tầng này chỉ quan tâm đến việc tương tác với từng tài nguyên cụ thể.

Việc thực thi của các giao thức này sẽ gọi các hàm của tầng nền để truy cập các nguồn tài nguyên.

Tầng thu thập (Collective layer)

Trong khi tầng tài nguyên tập trung vào sự tương tác với từng tài nguyên thì tầng này chứa đựng các giao thức và dịch vụ quản lý chung cho các nguồn tài nguyên. Các giao thức này được xây dựng dựa trên một số lượng nhỏ các giao thức nền tảng của các tầng tài nguyên và tầng kết nối. Chính vì vậy nó có thể thực thi nhiều dịch vụ khác nhau từ các giao thức nền tảng này.

Tầng ứng dụng (Application layer)

Tầng này bao gồm các ứng dụng của người dùng. Các ứng dụng có khả năng triệu gọi dịch vụ từ bất kì tầng nào. Tại mỗi tầng có thể có các APIs (thường do hãng thứ ba cung cấp) thực thi cùng các dịch vụ khác để đáp ứng nhu cầu người dùng.

MDS3 được xây dựng dựa trên OGSI (Open Grid Services Infrastructure), và các thành phần của MDS nằm ở hai tầng: tầng tài nguyên và tầng thu thập. MDS3 quản lý thông tin về trạng thái của lưới và tất cả các thành phần của lưới: mạng, các nút máy tính, hệ thống lưu trữ, và các công cụ. MDS3 cung cấp các thông tin lưới như các tài nguyên sẵn có và trạng thái của lưới tính toán. Những thông tin này có thể bao gồm cả thuộc tính của các máy tính, thuộc tính của mạng như: các bộ vi xử lí sẵn dùng, tải CPU, các giao diện mạng, thông tin hệ thống, băng thông, các thiết bị lưu trữ, và bộ nhớ.

Có hai loại dịch vụ trong MDS:

Các dịch vụ tài nguyên - Resource services

Các dịch vụ thu thập - Collective services

+ Các dịch vụ tài nguyên - Resource services

Trong kiến trúc OGSI, mọi tài nguyên đều được biểu diễn dưới dạng dịch vụ, mọi dịch vụ trong OGSI đều chứa dữ liệu dịch vụ về chính nó. Mọi tài nguyên đều được tham chiếu bởi một Grid Service Handle (một URI).

Các dịch vụ tài nguyên cung cấp các thông tin tài nguyên. Nhiều thông tin có thể là thông tin động: như tải, thông tin về bộ xử lí, thông tin lưu trữ...

+ Các dịch vụ thu thập - Collective services

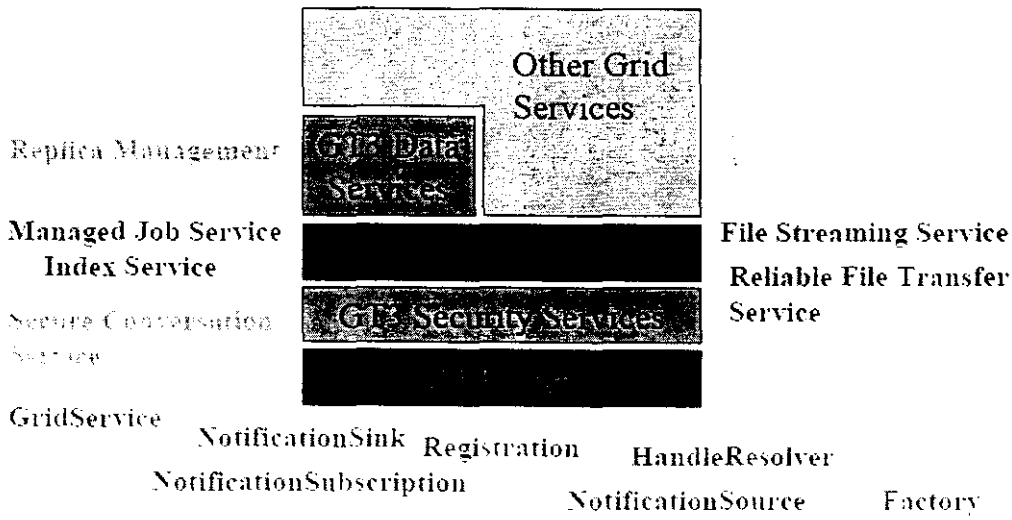
Các dịch vụ thu thập sẽ tập hợp thông tin từ nhiều dịch vụ tài nguyên khác nhau. Một trong các dịch vụ collective là dịch vụ chỉ mục - Index Service.

Trong khi, tầng tài nguyên chỉ cho phép truy cập đến một loại tài nguyên đơn thì tầng thu thập lại chứa các giao thức và dịch vụ cho phép giao tiếp giữa các tài nguyên trong tập hợp. Tầng này chứa một thành phần quan trọng của MDS3, đó là dịch vụ chỉ mục (Index Service).

5.2.2.2. Dịch vụ chỉ mục

- Vị trí của dịch vụ chỉ mục trong GT3

GT3 cung cấp các dịch vụ cơ bản (GT3 Base Services). Các dịch vụ cơ bản không có sẵn trong lõi GT3. Muốn sử dụng các dịch vụ này, ta phải cài đặt.



Hình 5-8: Các loại dịch vụ trong kiến trúc GT

Các dịch vụ cơ bản trong GT3 gồm:

Dịch vụ quản lí công việc - Job managerment

Dịch vụ chỉ mục - Index service, dùng để truy vấn thông tin

Dịch vụ truyền file tin cậy - Reliable File Transfer (RFT) services (multiRFT)

Dịch vụ chỉ mục là một trong các dịch vụ cơ bản của GT3, đây là một dịch vụ ở tầng cao dùng để truy cập vào dịch vụ và lấy các dữ liệu từ dịch vụ đó.

Dịch vụ chỉ mục có vai trò nhận và lưu trữ thông tin về các dịch vụ khác, thu thập dữ liệu dịch vụ từ nhiều tài nguyên khác nhau. Nó cho phép các client đăng kí và truy vấn dữ liệu, các dịch vụ khác có thể truy vấn về các phần tử dữ liệu dịch vụ hoặc đăng kí với dịch vụ chỉ mục để nhận được thông báo khi dữ liệu dịch vụ thay đổi...

- Dịch vụ chỉ mục là gì?

Như đã nhắc đến ở trên, trong OGSI, mọi tài nguyên đều được biểu diễn dưới dạng dịch vụ. Mỗi thực thể dịch vụ lưới có một tập các dữ liệu dịch vụ gắn với nó, tập dữ liệu dịch vụ này được biểu diễn dưới dạng SDE. Dịch vụ chỉ mục được coi như giao diện cho phép sinh, tập hợp, và truy vấn các SDE.

Dịch vụ chỉ mục trong GT3 là một công cụ hữu ích trong việc xây dựng các ứng dụng lưới. Nó được dùng để đánh chỉ mục cho các dữ liệu dịch vụ mang thông tin trạng thái từ nhiều thực thể dịch vụ lưới để dùng trong việc khai thác, lựa chọn và tối ưu tài nguyên cho các công việc trên lưới.

Dịch vụ này chịu trách nhiệm thu thập, lưu trữ thông tin của nhiều nguồn tài nguyên khác nhau. Các dịch vụ khác có thể truy vấn các phần tử dữ liệu dịch vụ hoặc đăng kí để được thông báo khi giá trị dữ liệu dịch vụ thay đổi. Dịch vụ chỉ mục đánh chỉ mục cho các dịch vụ lưới, ví dụ, nó có thể đánh chỉ mục cho dữ liệu dịch vụ từ nhiều thực thể dịch vụ lưới thông qua các giao diện: đăng kí (subscriptions) và thông báo (notifications) mà OGSI đã định nghĩa. Việc đánh chỉ mục có thể thực hiện theo cách tĩnh hoặc động:

+ Đánh chỉ mục tĩnh: sử dụng file cấu hình của dịch vụ chỉ mục

+ Đánh chỉ mục động: bằng cách gọi giao diện của dịch vụ chỉ mục

- Vai trò của dịch vụ chỉ mục

Có thể coi dịch vụ chỉ mục là một lớp các server, tập hợp thông tin từ nhiều server tài nguyên khác nhau. Trong MDS3, các server cung cấp thông tin chính là các dịch vụ lưới. Dịch vụ chỉ mục giống như các công cụ tìm kiếm hiệu quả, trả lời các câu truy vấn.

Đóng vai trò như các dịch vụ thông tin trong một hệ thống tính toán lưới dịch vụ chỉ mục (Index Service) đáp ứng đầy đủ những yêu cầu của một dịch vụ thông tin, đó là cung cấp các thông tin về hệ thống trong môi trường không đồng nhất, đảm bảo tính động của môi trường lưới, truy nhập dữ liệu hiệu quả từ nhiều nguồn dữ liệu khác nhau và được quản lí một cách phân tán. Trong kiến trúc OGSA, khi mà mọi thứ đều được coi như những dịch vụ, dịch vụ chỉ mục thực hiện các chức năng cơ bản sau:

Cập nhật và quản lí các dữ liệu dịch vụ một cách động thông qua các chương trình cung cấp thông tin (tương ứng với các bộ cung cấp thông tin – IP trong MDS 2);

Tổng hợp dữ liệu từ nhiều thực thể dịch vụ khác nhau;

Hỗ trợ đăng kí cho các thực thể dịch vụ;

Như vậy có thể thấy được những khả năng chính của dịch vụ chỉ mục trong GT3:

Cung cấp một giao diện cho phép việc tương ứng một thẻ dịch vụ GSH với một chương trình cung cấp thông tin lấy các thông tin tương ứng về dịch vụ;

Cung cấp một cơ chế cho phép việc tổng hợp các dữ liệu của nhiều thực thể dịch vụ với nhau. Việc này được thực hiện bằng cách kết hợp các chương trình cung cấp thông tin theo các cách khác nhau để có được các thông tin mong muốn. Lưu ý điều này là không thể thực hiện được với MDS2;

Đăng kí cho tất cả các dịch vụ khác. Điều này đã được đề cập qua ở phần trước tức là các dịch vụ có thể đăng kí và được quản lí bởi dịch vụ đăng kí (Registry) dịch vụ này sử dụng giao thức đăng kí lưới để cập nhật các thông tin về dịch vụ mà nó quản lí;

- Các thành phần của dịch vụ chỉ mục

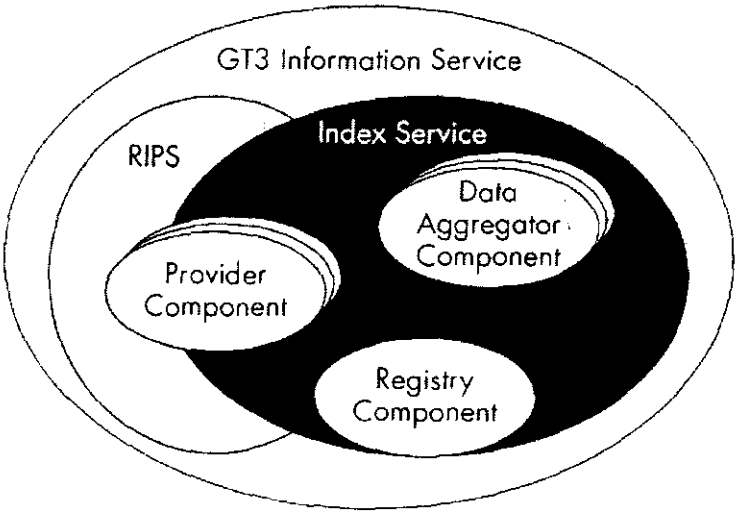
Đây là một trong những dịch vụ quan trọng nhất ở tầng trên mà dịch vụ thông tin sử dụng. Dịch vụ này chịu trách nhiệm quản lí thông tin trạng thái của các dịch vụ lưới (cả thông tin tĩnh và động).

3 thành phần quan trọng trong dịch vụ chỉ mục:

+ Thành phần Service Data Provider

+ Thành phần Service Data Aggregation

+ Thành phần Registry



Hình 5-9: Các thành phần chính trong dịch vụ chỉ mục

Các thành phần trong dịch vụ chỉ mục cung cấp các đặc tính sau:

+ Provider: cung cấp dữ liệu dịch vụ

+ Data Aggregator: tập hợp dữ liệu dịch vụ

+ Registry: cho phép các dịch vụ đăng kí với dịch vụ chỉ mục

- Các khả năng của dịch vụ chỉ mục

Mỗi một thực thể của dịch vụ lưới có một tập riêng biệt các dữ liệu dịch vụ gắn trên nó. Bản chất của dịch vụ chỉ mục là cung cấp một giao diện cho các thao tác truy cập, tập hợp, phát sinh, và truy vấn dữ liệu dịch vụ này.

Dịch vụ chỉ mục trong GT3 cung cấp:

+ Một giao diện để kết nối các thực thể với các bộ cung cấp dữ liệu dịch vụ bên ngoài: Có thể sinh dữ liệu dịch vụ một cách động sử dụng các chương trình bên ngoài (các bộ cung cấp có sẵn trong GT3 hoặc các bộ cung cấp do người dùng tự tạo)

+ Một framework để tập hợp dữ liệu dịch vụ: Có thể đánh chỉ mục cho các dịch vụ và hỗ trợ đăng kí, thông báo,...

+ Một cây đăng kí cho các dịch vụ lưới

Dịch vụ thông tin được cài đặt dựa trên dịch vụ chỉ mục. Nó có nhiệm vụ thu thập, tập hợp, và truy vấn các dữ liệu dịch vụ. Dịch vụ thông tin còn được sử dụng để đánh chỉ mục cho các dữ liệu dịch vụ chứa thông tin trạng thái từ nhiều thực thể của dịch vụ lưới nhằm phục vụ cho việc khai thác, lựa chọn và tối ưu việc sử dụng nguồn tài nguyên.

Tóm lại, các khả năng chính của dịch vụ chỉ mục có:

+ Sinh và quản lí các dữ liệu động thông qua các chương trình cung cấp dữ liệu dịch vụ (Service Data Provider)

+ Đăng kí các thực thể của dịch vụ bằng cách sử dụng giao diện ServiceGroup

+ Tập hợp dữ liệu dịch vụ từ nhiều thực thể của các dịch vụ lưới thông qua cơ chế tập hợp

+ Đánh chỉ mục

+ Truy vấn

+ Khả năng sinh

Dịch vụ chỉ mục cung cấp một cơ chế cho phép sinh các dữ liệu dịch vụ động thông qua các chương trình bên ngoài. Các bộ cung cấp bên ngoài này có thể là các bộ cung cấp lõi (có sẵn) hoặc do người dùng tự tạo.

Thành phần Service Data Provider (SDP) cung cấp một cơ chế cho phép sinh các SDE. GT3.2 chứa một tập các SDP lõi cũng như cho phép bạn tạo các SDP của riêng mình.

Các giao diện SDP được thiết kế và hỗ trợ chạy ở cả hai chế độ:

Chế độ đồng bộ ("pull")

Chế độ không đồng bộ ("push")

Một bộ cung cấp hợp lệ phải thoả mãn:

chứa một lớp Java cài đăt ít nhất một trong 3 giao diện cơ bản.

sau khi chạy, sinh ra đầu ra dạng XML

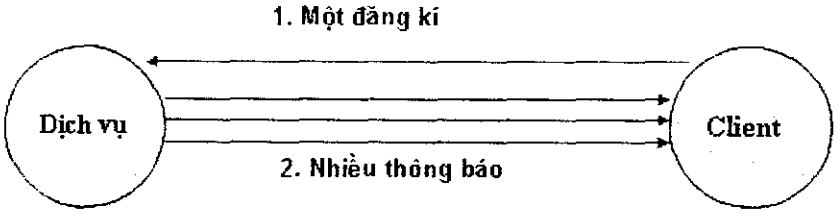
Đăng kí

Một tập các dịch vụ lưới sẵn có được lưu trữ trong bộ đăng kí. Bộ đăng kí cho phép đăng kí các dịch vụ lưới một cách mềm dẻo (tức là một tập các dịch vụ có thể được đăng kí và cập

nhật định kì theo yêu cầu). Sau đó, ta có thể truy vấn hoặc thực hiện các thao tác khác để yêu cầu các thông tin về dịch vụ trong bộ đăng kí đó.

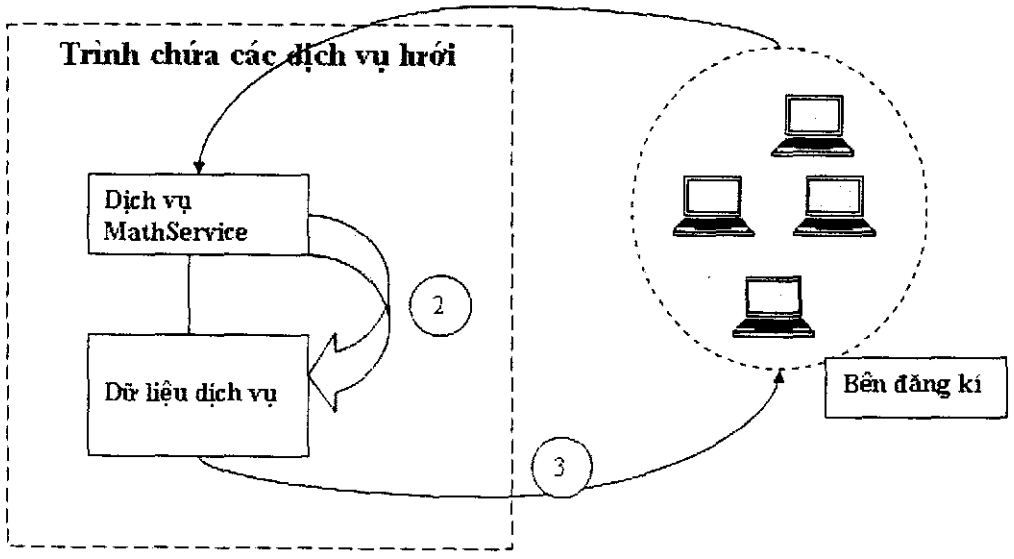
Dịch vụ chỉ mục có thể là một bộ đăng kí của các dịch vụ lưới. Bộ đăng kí này có thể được dùng để hỗ trợ các vấn đề: vòng đời, truy vấn, tập hợp dữ liệu hay các thao tác khác thực hiện trên các dịch vụ trong bộ đăng kí.

Chỉ cần đăng kí một lần, sau đó, mỗi lần dữ liệu dịch vụ thay đổi, client sẽ nhận được thông báo.



Hình 5-10: Cơ chế đăng kí

Giả sử ta có dịch vụ MathService thực hiện các phép toán cơ bản cộng, trừ, nhân, chia. Khi các bên yêu cầu có nhu cầu biết thông tin về dịch vụ MathService, chúng sẽ gửi đăng kí tới dịch vụ MathService, khi có thay đổi xảy ra, dịch vụ MathService sẽ thông báo về cho bên đã đăng kí



Hình 5-11: Đăng kí và nhận thông báo

1. Một dịch vụ nào đó đăng kí với dịch vụ MathService để được nhận thông tin về SDE của dịch vụ MathService

2. Cập nhật giá trị của SDE khi thay đổi xảy ra

3. Thông báo lại cho bên đăng kí khi có thay đổi

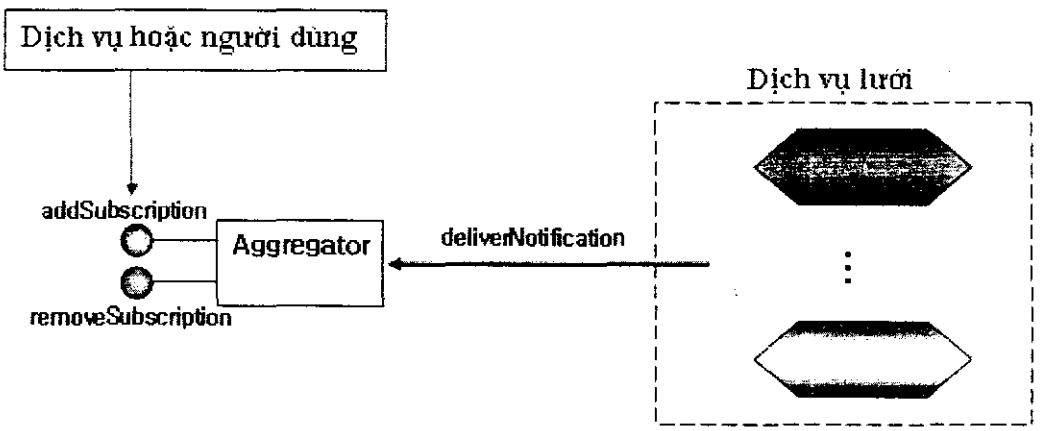
Tập hợp

Dịch vụ chỉ mục tập hợp dữ liệu dịch vụ từ nhiều thực thể dịch vụ lưới sử dụng cơ chế đăng kí-thông báo đã được định nghĩa trong OGSI. Dịch vụ chỉ mục đăng kí với các dữ liệu dịch vụ mà nó muốn đánh chỉ mục, nhờ đó mà nó sẽ thông báo và lấy giá trị mới nhất khi giá trị đó thay đổi. Dịch vụ cung cấp dữ liệu được đánh chỉ mục phải đượcc cài đặt giao diện NotificationSource.

Dịch vụ chỉ mục cung cấp một framework phục vụ việc tập hợp các dữ liệu dịch vụ từ các dịch vụ khác. Dữ liệu dịch vụ từ các chương trình SDP, các dịch vụ lưới khác có thể được tập hợp theo nhiều cách khác nhau và được đánh chỉ mục để việc truy vấn được hiệu quả. Nhiều công cụ dòng lệnh khác nhau và các giao diện GUI có thể được sử dụng như là các client để hiển thị dữ liệu. Dịch vụ chỉ mục sử dụng các cơ chế khai báo OGSA chuẩn để đăng kí, thông báo và cập nhật dữ liệu dịch vụ.

Thành phần ServiceDataAggregator: cung cấp cơ chế đăng kí, thông báo và cập nhật các bản sao dữ liệu dịch vụ được lưu trữ địa phương (các dữ liệu này được sinh ra bởi các dịch vụ khác). Với việc sử dụng lớp ServiceDataAggregator trong dịch vụ, các dữ liệu dịch vụ từ các bộ cung cấp thông tin địa phương và các thực thể dịch vụ có thể được tập hợp lại trong một dịch vụ nào đó.

Các SDE được tập hợp sẽ được tổ chức theo SDE QName và được lưu trữ trong một mảng.



Hình 5-12: Cơ chế tập hợp

Đánh chỉ mục

Dịch vụ chỉ mục là một nút đánh chỉ mục và sinh dữ liệu động. Nó đánh chỉ mục cho các thành phần đã được tập hợp bởi ServiceDataAggregator. Như vậy, dịch vụ chỉ mục tương tự như một GIIS (Grid Infomation Index Service) phân cấp trong MDS2.

Có hai cách đánh chỉ mục:

+ Đánh chỉ mục tĩnh
+ Đánh chỉ mục động

Đánh chỉ mục tĩnh cho các dịch vụ lưới

Dịch vụ chỉ mục có thể được cấu hình để đánh chỉ mục tĩnh cho dữ liệu dịch vụ của các dịch vụ lưới thông qua file cấu hình của dịch vụ chỉ mục. Đường dẫn của file cấu hình được xác định trong tham số serviceConfig trong phần mô tả triển khai trong file cấu hình server server-config.wsdd, mà đường dẫn mặc định là etc/index-service-config.xml. Do đó, khi GT3 được triển khai trên một server ứng dụng như Tomcat hay WebSphere Application Server, file index-service-config.xml nên được sao lưu tới thư mục tương ứng WEB-INF/etc.

File cấu hình của dịch vụ chỉ mục gồm 3 phần, trong đó phần aggregatedSubscriptions được dùng để xác định dữ liệu dịch vụ cần được đánh chỉ mục. Phần aggregatedSubscriptions chứa một danh sách các phần tử AggregatorSubscription, mỗi phần tử xác định một dữ liệu dịch vụ cần được đánh chỉ mục. Phần tử con serviceDataName xác định QName của dữ liệu dịch vụ. Phần tử con source xác định dịch vụ sẽ đăng kí. Phần tử con lifetime xác định thời gian sống của đăng kí.

Đánh chỉ mục động cho dịch vụ lưới

Ngoài cách đánh chỉ mục tĩnh, dịch vụ chỉ mục còn có khả năng đánh chỉ mục động thông qua giao diện đươc cung cấp bởi Index Service. Có hai thao tác tên là addSubscription và removeSubscription. Thao tác addSubscription được dùng để thêm một đăng kí vào một dữ liệu dịch vụ mới, để dữ liệu dịch vụ được đánh chỉ mục động, còn thao tác removeSubscription xoá một đăng kí đã có.

Tóm lại, đánh chỉ mục tĩnh và động là hai cách để đánh chỉ mục cho các dịch vụ lưới sử dụng dịch vụ chỉ mục trong GT3, cả hai cách đều đảm bảo độ tin cậy như nhau.

Khả năng truy vấn

Công việc truy vấn dữ liệu trong dịch vụ chỉ mục cũng tương tự như truy vấn trong bất kì dịch vụ lưới nào, các truy vấn này đều được xử lí thông qua giao diện findServiceData của dịch vụ lưới. Tức là, theo mặc định thì sẽ truy vấn theo tên, tên của dữ liệu dịch vụ để truy vấn chính là tên được xác định bởi QName, còn nếu ta đưa ra các tuỳ chọn, ta có thể truy vấn theo các tiêu chí khác.

Có hai cơ chế truy cập vào dữ liệu dịch vụ:

+ findServiceData

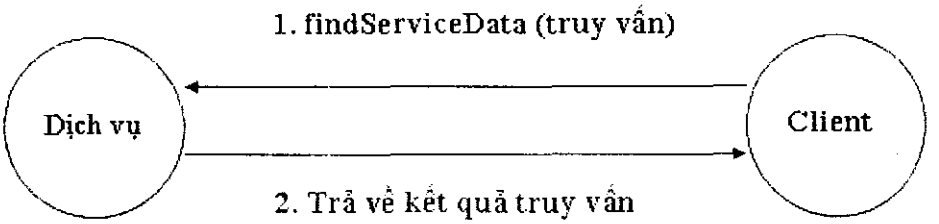
+ Subscription/notification

Mỗi cơ chế có hai loại truy vấn mở rộng:

+ Chuẩn OGSI: truy vấn theo tên (by name)

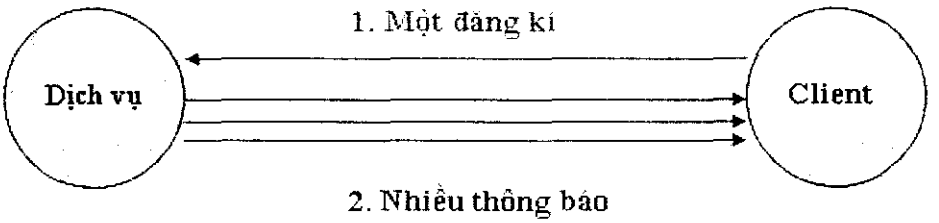
+ Đặc trưng của GT3: truy vấn theo XPath (by XPath)

Với cơ chế findServiceData: Mỗi một truy vấn có một câu trả lời



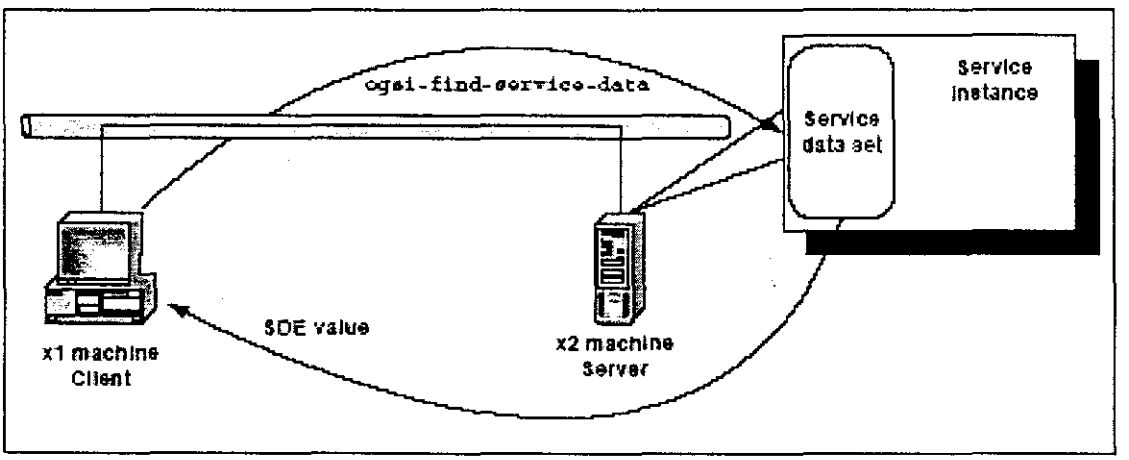
Hình 5-13: Cơ chế truy vấn

Với cơ chế đăng kí: Chỉ cần một lần đăng kí nhưng sẽ có câu trả lời mỗi lần dữ liệu dịch vụ thay đổi.



Hình 5-14: Cơ chế đăng kí

Dịch vụ chỉ mục cung cấp một lệnh phía client tên là ogsi-find-service-data. Lệnh này cho phép truy vấn một SDE từ một dịch vụ bất kì.Ví dụ minh họa trong hình sau truy vấn một SDE trong một dịch vụ cụ thể chạy trên lưới.

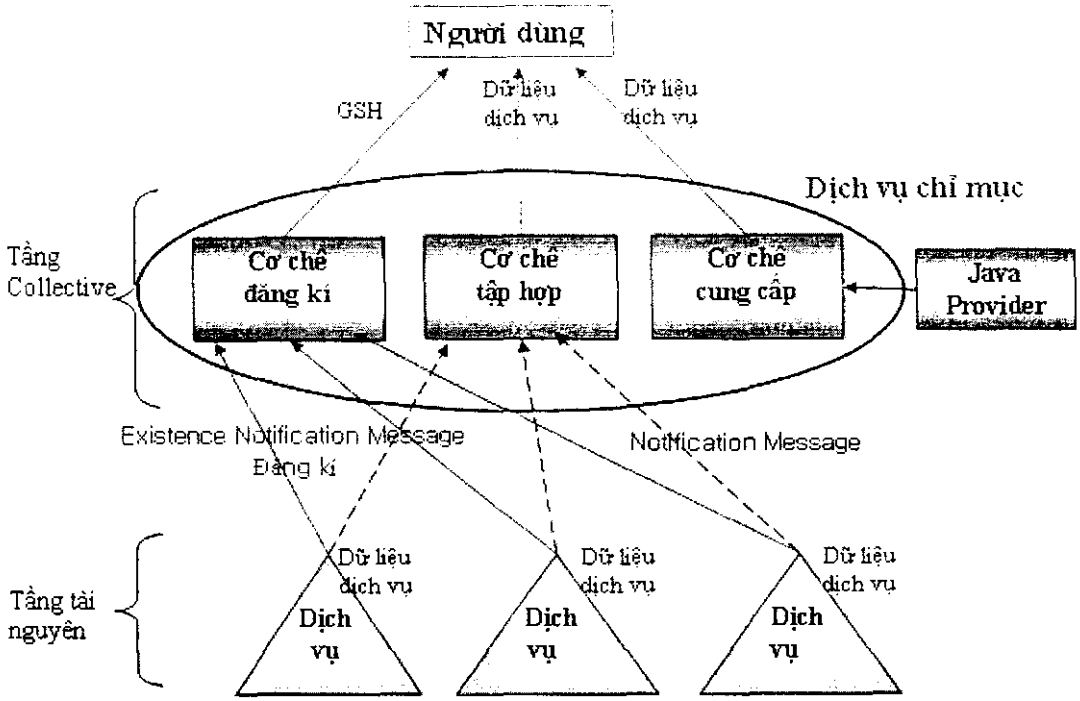


Hình 5-15: Minh hoạ cho lệnh ogsi-find-service-data

Client gửi lệnh ogsi-find-service-data tới server để truy vấn thông tin, server gửi trả về cho client giá trị SDE mà client truy vấn.

- Cơ chế làm việc

Để hiểu hơn về cơ chế hoạt động của dịch vụ chỉ mục, ta xem sơ đồ dưới đây.



Hình 5-16: Cơ chế cập nhật thông tin trong dịch vụ chỉ mục

Có thể thấy có hai cơ chế qua đó người sử dụng hay các chương trình ứng dụng có thể lấy được thông tin về dịch vụ qua dịch vụ chỉ mục, thứ nhất dịch vụ chỉ mục hỗ trợ cơ chế tập hợp, cơ chế này thực hiện nhờ giao diện notification trong OGSA tức là mỗi dịch vụ được đăng kí với dịch vụ chỉ mục bằng giao thức đăng kí lưới, sau mỗi khoảng thời gian nhất định các dịch vụ này thực hiện thao tác thông báo về trạng thái cũng như dữ liệu tương ứng tới dịch vụ chỉ mục, tại đây dữ liệu của các dịch vụ có thể được lưu trữ tạm thời nhằm đáp ứng nhu cầu tìm kiếm và truy vấn. Như vậy, có thể coi mỗi dịch vụ lưới là một chương trình cung cấp thông tin với điều kiện chúng phải được cài đặt giao diện notification.

Dịch vụ chỉ mục nhìn chung đã đáp ứng được các yêu cầu chung của dịch vụ thông tin trong tính toán lưới và phù hợp với kiến trúc OGSA cũng như công nghệ dịch vụ lưới.

- Các giao diện có liên quan đến dịch vụ chỉ mục

Trong phần này ta sẽ xem cách mà các dịch vụ khác lấy thông tin từ dịch vụ thông tin (hay các giao diện và các chức năng mà các dịch vụ sử dụng khi giao tiếp với dịch vụ chỉ mục).

Factory

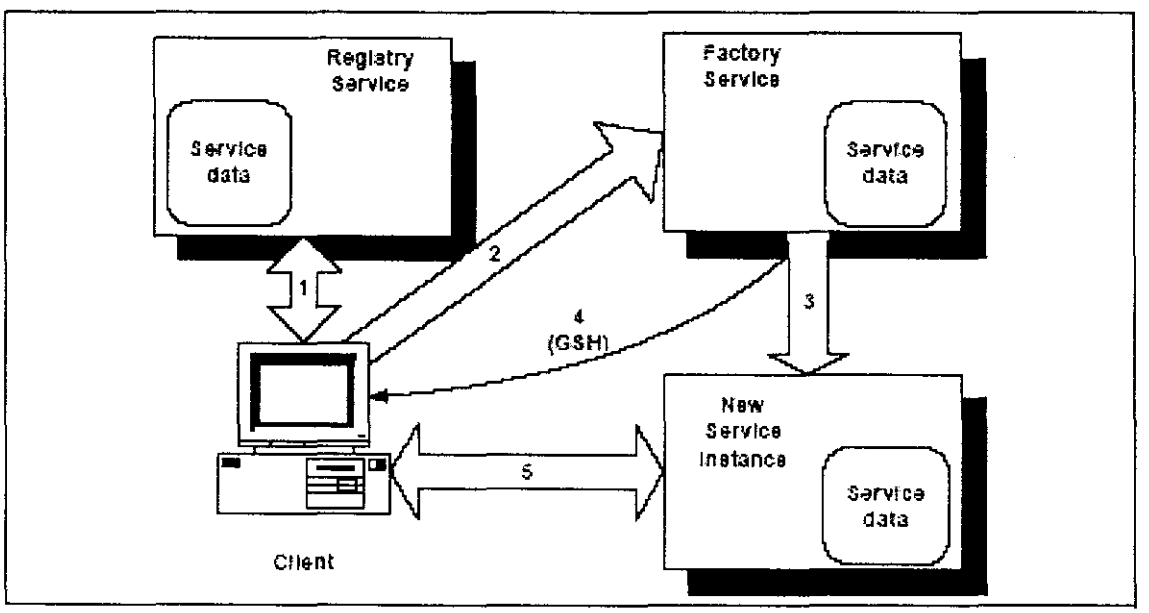
Factory được dùng để tạo ra một thực thể mới cho dịch vụ lưới thông qua thao tác CreateService. Các dịch vụ muốn sử dụng chức năng này cần phải cài đặt một factory (khái niệm factory tương tự như factory trong lập trình hướng đối tượng). Factory là một dịch vụ tạm thời tạo ra các thực thể để các client có thể tương tác với dịch vụ. Trong lập trình hướng đối tượng, factory được dùng để tạo ra các thực thể cho một dịch vụ. Trong môi trường lưới,

một factory tạo ra các thực thể và có một bộ đăng kí để theo dõi các thực thể tạo điều kiện cho việc khai thác các dịch vụ của các client.

Các client đầu tiên phải định vị factory, sau đó yêu cầu tạo thực thể của dịch vụ. Theo yêu cầu, một factory tạo một thực thể của dịch vụ và trả về một GSH và một GSR cho các client. GSH là định danh duy nhất và các client có thể sử dụng để giao tiếp với thực thể của dịch vụ. Client không giao tiếp với factory nữa mà giao tiếp trực tiếp với thực thể của dịch vụ. Thực thể của dịch vụ duy trì các dữ liệu trạng thái liên quan đến client triệu gọi nó. Mỗi client khi truy cập đến dịch vụ sẽ được truy cập đến một thực thể riêng của dịch vụ đó và cũng có thể có nhiều client cùng tương tác với cùng một thực thể. Thực thể sẽ bị huỷ khi các client không cần đến nó nữa, hoặc ta cũng có thể đặt thời gian kết thúc (việc này là tuỳ chọn, không bắt buộc). Khi thực thể bị huỷ, nó giải phóng các tài nguyên mà nó đã sử dụng trong thời gian hoạt động.

Hoạt động tương tác giữa client – factory và thực thể được mô tả như sau:

1. Client khai thác một factory bằng cách truy vấn dịch vụ registry để lấy handle của factory.
2. Client gọi phương thức factory để tạo một thực thể của dịch vụ lưới
3. Factory tạo một thực thể mới của dịch vụ lưới
4. Factory trả về GSH của thực thể cho client
5. Client và dịch vụ tương tác trực tiếp với nhau



Hình 5-17: Minh hoạ khái niệm Factory

Ví dụ: hình trên minh họa khái niệm một client yêu cầu một thực thể từ factory. Factory tạo một thực thể và trả về GSH hay URL của thực thể cho client. Client sử dụng GSH này để giao tiếp trực tiếp với thực thể của dịch vụ.

Tóm lại, factory là một thao tác để tạo một dịch vụ và trả về GSH gắn với dịch vụ được yêu cầu, tuy nhiên thông tin này chưa đủ để có thể triệu gọi một dịch vụ, muốn gọi dịch vụ, ta

phải cần đến GSR. Để chuyển từ GSH sang GSR, ta cần đến phương thức HandleResolver. Chú ý là GSH là một tham chiếu bền vững, nhưng GSR lại có giới hạn về thời gian.

Truy vấn

Phương thức **FindServiceData** có trong giao diện của dịch vụ lưới và nó được dùng để truy vấn các SDE. Client sử dụng câu lệnh **ogsi-find-service-data** làm giao diện truy vấn. Theo mặc định thì sẽ truy vấn theo tên dịch vụ, còn nếu ta thêm các tuỳ chọn thì có thể truy vấn theo GSH,...

Ví dụ, để biết tên các dữ liệu dịch vụ của một dịch vụ lưới, ta dùng lệnh sau:

```
ogsi-find-service-data                                    -service
http://128.9.72.46:9103/ogsa/services/core/registry/ContainerRegistr
yService
    -sde gsdl:serviceDataName
```

Lệnh này sẽ cho hiện ra tên của các SDE trong dịch vụ Container Registry Service.

Kết quả đầu ra như sau:

```
<ns1:result                              xsi:type="ns1:ExtensibilityType"
xmlns:ns1="http://www.gridforum.org/namespaces/2003/03/OGSI">
                                                    <ns2:serviceDataValues
xmlns:ns2="http://www.gridforum.org/namespaces/2003/03/serviceData">
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:entry</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:gridServiceHandle</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:serviceDataName</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:interface</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:setServiceDataExtensibility</ns1:serviceDat
aName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:terminationTime</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:membershipContentRule</ns1:serviceDataName>
                                                        <ns1:serviceDataName
xsi:type="xsd:QName">ns1:subscribeExtensibility</ns1:serviceDataName
>
```

```
                                                    <ns1:serviceDataName
xsi:type="xsd:QName">ns1:gridServiceReference</ns1:serviceDataName>
                                                    <ns1:serviceDataName
xsi:type="xsd:QName">ns1:notifiableServiceDataName</ns1:serviceDataN
ame>
          <ns1:serviceDataName     xsi:type="xsd:QName">ns1:ogsi-find-
service-dataExtensibility</ns1:serviceDataName>
     </ns2:serviceDataValues>
    </ns1:result>
```

Đăng kí - Registry

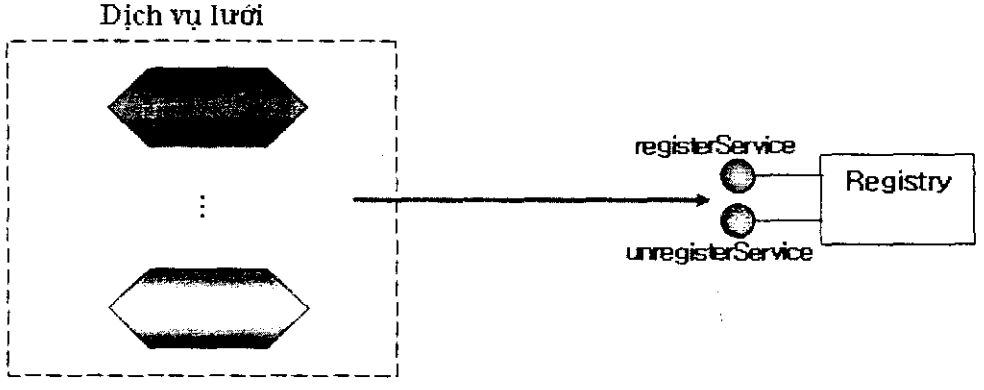
Các giao diện dịch vụ nhóm - Service group portTypes

Một dịch vụ nhóm (ServiceGroup) là một dịch vụ lưới duy trì thông tin về một nhóm các dịch vụ lưới khác

Một dịch vụ lưới có thể thuộc về một hoặc nhiều dịch vụ nhóm

Một dịch vụ cài đặt giao diện ServiceGroup sẽ trở thành dịch vụ nhóm

Giao diện Servicegroup không bắt buộc trong OGSI



Hình 5-18: Cơ chế đăng kí

Registry là một dịch vụ nhóm

Registry mô tả một tập hợp các dịch vụ khác

Nội dung của một dịch vụ nhóm chính là các entry SDE

Các entry này có một tham chiếu đến dịch vụ thành viên và các dữ liệu riêng của từng dịch vụ

Một số dịch vụ nhóm:

Registry: Tập hợp các dịch vụ đã được đăng kí

ContainerRegistry: Tập hợp các dịch vụ đang chạy trên môi trường host.

Index service: Tập hợp các dịch vụ mà dữ liệu dịch vụ của chúng đã được đánh chỉ mục.

Bất kì một dịch vụ GT3.2 nào cũng có thể đăng kí với một dịch vụ chỉ mục của GT3.2 bằng cách sử dụng RegistryPublishProvider. RegistryPublishProvider được cấu hình trong file

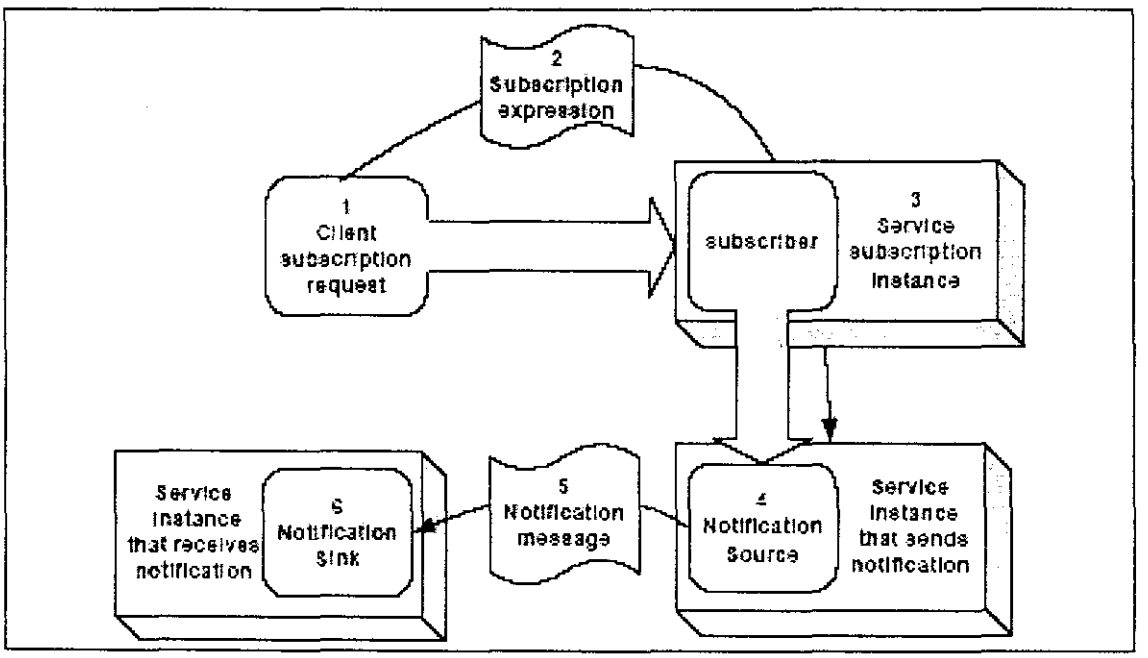
WSDD của các dịch vụ tài nguyên (ví dụ dịch vụ MMJFS), và công việc cấu hình này được thực hiện bởi admin của các dịch vụ đó.

Thông báo - Notification

Client sử dụng giao diện NotificationSource để đăng kí với một dịch vụ mà nó quan tâm. Client sẽ gửi cho dịch vụ thông tin về các vấn đề mà nó muốn đăng kí. Giao diện NotificationSink được sử dụng để truyền các bản thông báo khi có dữ liệu thay đổi.

Khi một ứng dụng hay một dịch vụ lưới quan tâm đến một SDE cụ thể của một dịch vụ lưới nào đó. Các đăng kí được gửi từ bên muốn nhận thông báo đến dịch vụ lưới chứa SDE mong muốn. Nói cách khác, bên nhận thông báo sẽ gọi phương thức đăng kí của bên gửi thông báo. Các bản thông báo sau đó sẽ được gửi tới bên nhận thông báo khi giá trị của SDE thay đổi.

Thông báo là một cơ chế rất hiệu quả dùng để theo dõi sự thay đổi của dữ liệu dịch vụ. Cơ chế này được dùng khi một thành viên trong **registry** muốn nhận được thông báo khi giá trị của SDE thay đổi. Bên nhận thông báo gọi là **notification sink**. Bên gửi thông báo là **notification source** là dịch vụ chứa các SDE mà dịch vụ khác quan tâm, và cũng là dịch vụ sinh ra các thông báo.



Hình 5-19: Cơ chế làm việc Notification

1. Yêu cầu đăng kí

Thông báo được gửi đến bên nhận đăng kí (notification sink) để yêu cầu đăng kí. Một yêu cầu đăng kí sẽ tạo nên một thực thể của dịch vụ để gọi một đăng kí.

2. Bản đăng kí

Là một văn bản dạng XML mô tả các luật và các định dạng của bản thông báo, ví dụ như nơi nhận bản thông báo, và khi nào thì bản thông báo được gửi.

3. Bộ quản lí đăng kí

Một thực thể của bộ quản lí đăng kí được tạo ra để quản lí các thông tin đã đăng kí.

4. Bên gửi thông báo

Là dịch vụ nhận đăng kí, và chịu trách nhiệm gửi thông báo khi có SDE thay đổi. Đây chính là nơi sẽ tạo ra thực thể của bộ quản lí đăng kí. Bên gửi thông báo trả về handle của thực thể của bộ quản lí đăng kí cho bên nhận thông báo.

5. Bản thông báo

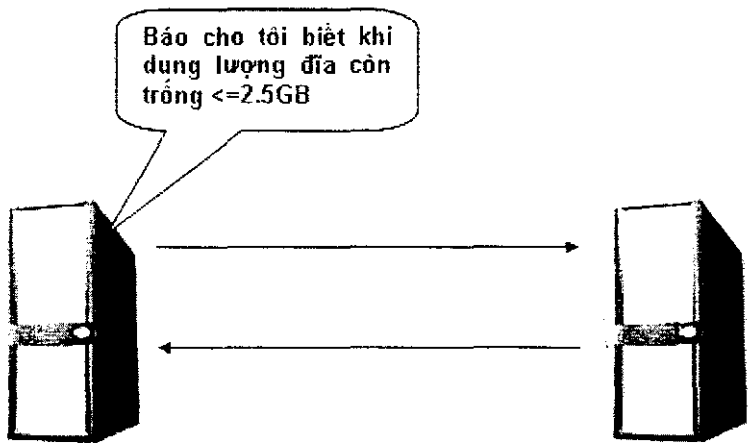
Khi một thực thể của dịch vụ muốn nhận được một thông báo liên quan đến một SDE cụ thể, dịch vụ cần được đăng kí để được thông báo khi dữ liệu bị thay đổi.

6. Bên nhận thông báo

Bên nhận thông báo là một thực thể của dịch vụ lưới chịu trách nhiệm nhận thông báo. Nó có thể sử dụng handle của bộ quản lí đăng kí để quản lí vòng đời đăng kí.

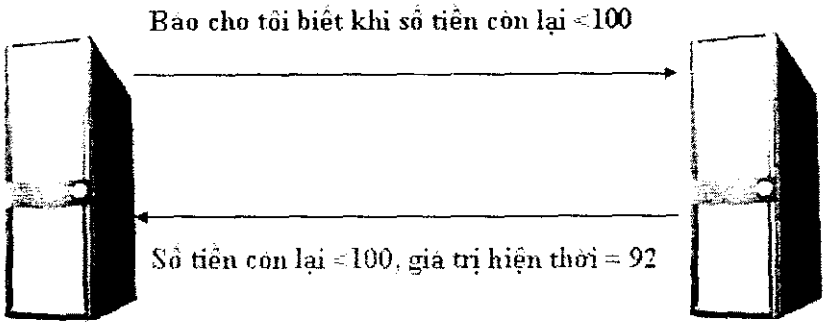
Khi điều kiện trong bản đăng kí thoả mãn, một bản thông báo được gửi tới bên đã đăng kí để nhận thông báo.

Ví dụ, một người sử dụng dịch vụ muốn nhận được thông báo khi hệ thống file trên nút mà dịch vụ đó đang chạy không còn đủ chỗ trống để submit thêm công việc.



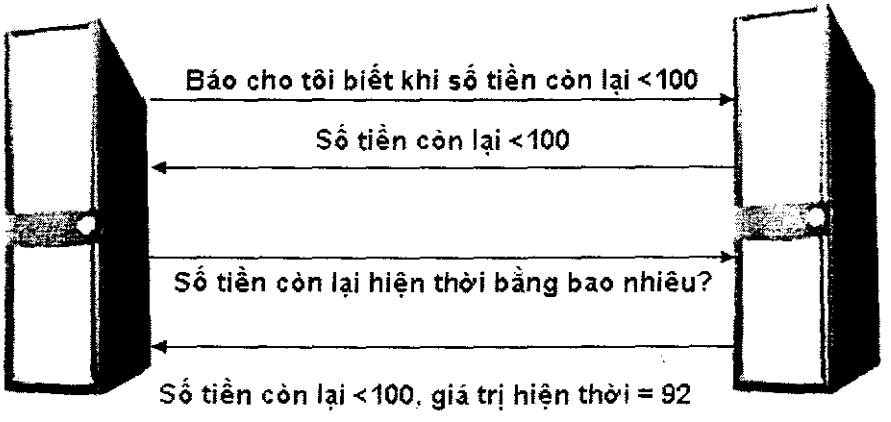
Hình 5-20: Nhận thông báo khi không còn chỗ trống để submit công việc

Có hai kiểu thông báo, kiểu thứ nhất gọi là kiểu push, tức là một client đăng kí để nhận được thông báo khi thoả mãn một điều kiện nào đó. Khi điều kiện thoả mãn, một thông báo được gửi tới bên đăng kí, thông báo đó bao gồm cả các thông tin về điều kiện đã thoả mãn. Trong nhiều trường hợp, bản thông báo được gửi kèm theo giá trị của Service Data Element đã được đăng kí.



Hình 5-21: Cơ chế thông báo Push

Không phải tất cả các client đăng kí với notification đều cần biết giá trị mới của Service Data Element. Trong một vài trường hợp, client có thể chỉ cần biết giá trị đó đã thay đổi, mà không cần biết giá trị mới là bao nhiêu. Trong trường hợp này, ta có một kiểu đăng kí khác, gọi là kiểu pull, kiểu này có vẻ hiệu quả hơn. Trong kiểu pull, bên đăng kí được thông báo sự thay đổi giá trị của SDE nhưng giá trị mới không được gửi tới dịch vụ đăng kí. Các bên đăng kí muốn biết giá trị hiện thời của SDE thì phải gửi thêm một lời gọi nữa và yêu cầu thông báo giá trị đó.



Hình 5-22: Cơ chế thông báo Pull

Có thể có nhiều client cùng đăng kí để nhận được thông báo khi giá trị SDE thay đổi. Các bên đăng kí hay các client được gọi là bên nhận thông báo. Các thông báo được xử lí bởi dịch vụ quản lí đăng kí. Dịch vụ hoặc bên thông báo sẽ thông báo cho dịch vụ quản lí đăng kí rằng giá trị SDE đã thay đổi. Bên quản lí đăng kí có trong tay danh sách các client và các dịch vụ quan tâm đến giá trị của SDE và dịch vụ quản lí đăng kí sẽ thông báo với mỗi client hoặc các bên đã đăng kí để nhận thông báo về sự thay đổi của SDE đó.

Chú ý: Khi cài đặt giao diện thông báo, dịch vụ nên là bền vững và không phải là một dịch vụ được tạo bởi client để làm một việc gì đó, sau đó bị huỷ.

5.2.2.3. Bộ cung cấp dữ liệu dịch vụ - Service Data Provider

- Định nghĩa

Service Data Provider (SDP) có vai trò thu thập dữ liệu, tập hợp một tài liệu XML từ các dữ liệu, và sau đó phát hành tài liệu dạng XML.

Đặc biệt, một SDP được định nghĩa là một lớp Java thi hành ít nhất một trong ba giao diện Java (SimpleDataProvider, DOMDataProvider, và AsyncDataProvider), và sinh ra một tài liệu dạng XML có đặc điểm là tương thích và định dạng tốt.

"Định dạng tốt" có nghĩa là tài liệu XML có thể được phân tích trong bất kì môi trường nào, tức là bất kì công cụ phân tích nào được viết bằng bất kì ngôn ngữ nào cũng có thể được sử dụng.

"Tương thích" có nghĩa là một dạng tích hợp với bộ quản lí cung cấp dữ liệu dịch vụ, một biểu diễn dạng Java output stream hoặc DOM.

- Các bộ cung cấp Core GT3

GT3 chứa các bộ cung cấp dữ liệu dịch vụ sau:

+ SimpleSystemInformationProvider: cung cấp thông tin về host. Bộ cung cấp này liệt kê các dữ liệu sau: bộ đếm CPU (CPU count), các trạng thái bộ nhớ, kiểu HĐH (OS type), dung lượng bộ nhớ logic.

+ HostScriptProvider: Bộ cung cấp này dành riêng cho các hệ hống Unix, Linux, nó cung cấp nhiều loại thông tin tài nguyên host chi tiết khác nhau. Đây là một tập gồm các shell scripts theo dõi dữ liệu host đặc tả hệ thống. Những scripts này về cơ bản giống với các thông tin được bộ cung cấp thông tin cung cấp MDS2 (thích hợp với đầu ra dạng XML).

+ AsyncDocumentProvider: Một bộ cung cấp tiện ích sử dụng giao diện AsyncDataProvider. Bộ cung cấp dữ liệu này làm việc với dữ liệu dạng XML. Đây là một phiên bản đồng bộ của một bộ cung cấp SimpleSystemInformationProvider.

+ ScriptExecutionProvider: Có khả năng cung cấp cách để chạy các scripts và sinh ra tài liệu dạng XML.

Theo mặc định, các bộ cung cấp dữ liệu dịch vụ SDP sau là sẵn có:

+ SimpleSystemInformation: Bộ cung cấp này liệt kê các dữ liệu sau: bộ đếm CPU (CPU count), các trạng thái bộ nhớ, kiểu HĐH (OS type), dung lượng bộ nhớ logic.

+ HostScriptProvider: Một bộ cung cấp dùng riêng cho các hệ thống Linux, và Unix.

Các bộ cung cấp này được chạy khi dịch vụ chỉ mục kích hoạt. Muốn kích hoạt ta có 2 cách: dựa vào giao diện của công cụ Service Data Browser hoặc dựa trên các tham số được xác định trong file index-service-config.xml trong thư mục gốc /etc của GT3.2.

- Các bộ cung cấp do người sử dụng tạo

Các bộ cung cấp dữ liệu dịch vụ có thể đơn giản hay phức tạp tuỳ theo các yêu cầu. Trường hợp cơ bản chỉ yêu cầu nhà phát triển bộ cung cấp tạo một lớp Java thực thi các hàm của giao diện – SimpleDataProvider – mà mục đích của nó là để cung cấp đầu ra dưới dạng XML.

- Các bước tạo một bộ cung cấp thông tin

Các bước sau rất cần thiết cho việc tạo một SDP mới:

1. Chọn giao diện bộ cung cấp để thực thi, dựa trên các nhu cầu hoặc ràng buộc của ứng dụng.

2. Viết code để sản xuất tập dữ liệu dưới dạng một tài liệu XML. Việc này có thể được thực hiện trong một chương trình ngoại trú.

3. Tạo một entry cho dịch vụ, trong dịch vụ đó bạn dự định chạy một bộ cung cấp trong một file cấu hình dịch vụ trợ giúp. Dịch vụ này giả sử có chức năng của SDP Manager, các thuộc tính của bộ cung cấp đã được xác định trong danh mục mô tả trong file cấu hình mặc định: server.config.wsdd, tải bộ cung cấp về, và chạy nó theo các tham số được xác định bởi một dịch vụ phía client.

Thành phần cơ bản của thông tin trong thư mục là các danh mục (entry), mỗi danh mục chứa các thông tin mô tả một đối tượng như thông tin về một máy chủ v.v..

- Các giao diện của bộ cung cấp

Các giao diện SDP được thiết kế để hỗ trợ việc chạy trong chế độ ("pull") đồng bộ hoặc chế độ ("push") không đồng bộ. Điều đó tuỳ thuộc vào các nhà phát triển, họ sẽ chọn giao diện bộ cung cấp phù hợp để thực hiện, dựa trên các nhu cầu của các ứng dụng đặc trưng.

Có 3 giao diện cho các bộ cung cấp: SimpleDataProvider, DOM DataProvider, và AsyncDataProvider, 3 loại này được mô tả dưới đây:

+ SimpleDataProvider

Đây là một bộ cung cấp đồng bộ sản xuất đầu ra dạng XML. SimpleDataProvider là giao diện cơ bản mà tất cả các bộ cung cấp dữ liệu dịch vụ phải cài đặt

+ DOMDataProvider

Đây là sự mở rộng *đồng bộ* của SimpleDataProvider cũng có thể cung cấp đầu ra dưới dạng XML

+ AsyncDataProvider

Đây là một phiên bản *không đồng bộ* của SimpleDataProvider cho phép phân phối dữ liệu ở chế độ "push".

- Đầu vào và ra của các bộ cung cấp dữ liệu dịch vụ

Đầu vào cho bộ cung cấp dữ liệu dịch vụ được xác định thông qua một tập các đối số của phương thức run. Chuỗi đối số được chuyển đến các bộ cung cấp. Ngoài ra, phương thức getDefaultArgs có thể được sử dụng để phục hồi danh sách đối số mặc định cho bộ cung cấp.

Đầu ra của một bộ cung cấp dữ liệu dịch vụ là ở dạng XML. Đầu ra này trở thành giá trị của SDE (Service Data Element) và do đó nó trở nên thông dụng như là một phần của SDE. Các SDE có thể được sử dụng trong rất nhiều hàm của dịch vụ thông tin trong GT3.2.

- Bộ quản lí cung cấp dữ liệu dịch vụ - SDP Manager

Bộ quản lí cung cấp dữ liệu gồm các entry dùng để cấu hình cho bộ cung cấp, sau khi cấu hình, các bộ cung cấp có thể chạy dưới sự quản lí của bộ quản lí, và thông báo sự tồn tại của bộ cung cấp cho các client.

Các thuộc tính cần có trong entry cấu hình được đặt trong "class", trong đó ghi đường dẫn đầy đủ của các lớp giao diện.

```
xml version="1.0" encoding ="UTF-8"

<providers>

<provider
class="org.globus.ogsa.impl.base.providers.servicedata.impl.SimpleSy
stemInformationProvider"/>

<provider
class="org.globus.ogsa.impl.base.providers.servicedata.impl.AsynDocu
mentProvider"/>

<provider
class="org.globus.ogsa.impl.base.providers.servicedata.impl.ScriptEx
eccutionProvider"/>

<provider
class="org.globus.ogsa.impl.base.providers.servicedata.impl.HostScri
ptProvider"/>
```

Việc chạy bộ cung cấp dữ liệu dịch vụ được xử lí bởi lớp SDP Manager, SDP Manager sử dụng file cấu hình dịch vụ hỗ trợ để load và link các bộ cung cấp dữ liệu dịch vụ đã cài đặt trong suốt thời gian chạy.

Giao diện chạy SDP và bộ quản lí cung cấp dữ liệu dịch vụ

Giao diện chạy SDP gồm 2 phương thức, enumProviders và executeProvider, cả 2 đều được thực hiện bởi lớp ServiceDataProviderManager.

Phương thức enumProviders sản xuất một danh sách các bộ cung cấp.

```
- <xsd:complexType name="ServiceDataProviderElement">
  - <xsd:sequence>
      <xsd:element name="ServiceDataProvider" type="tns:ServiceDataProviderType" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="ServiceDataProviderType">
  - <xsd:sequence>
      <xsd:element name="providerName" type="xsd:string" />
      <xsd:element name="providerDesc" type="xsd:string" />
      <xsd:element name="providerImpl" type="xsd:string" />
      <xsd:element name="providerArgs" type="xsd:string" />
      <xsd:element name="async" type="xsd:boolean" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:element name="enumProviders">
  - <xsd:complexType>
    - <xsd:sequence>
        <xsd:element name="rescanConfig" type="xsd:boolean" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="enumProvidersResponse" type="tns:ServiceDataProviderElement" />
```

Hình 5-23: Kết quả đầu ra của phương thức enumProviders

Phương thức executeProvider:

Các tham số ServiceDataProviderExecutionType được xác định như sau:

serviceDataProviderName (string): Là tên hiển thị của Service Data Provider để chạy.

serviceDataProviderImpl (string): Là tên lớp Java có đủ điều kiện để sử dụng trong việc cài đặt provider. Tham số này không thể ghi đè, và phải phù hợp với tham số serviceDataProviderName trong cấu trúc ServiceDataProviderExecutionType.

serviceDataProviderArgs (string): Là các tham số gửi đến cho bộ cung cấp tại thời điểm chạy, tuỳ thuộc vào việc cài đặt các bộ cung cấp.

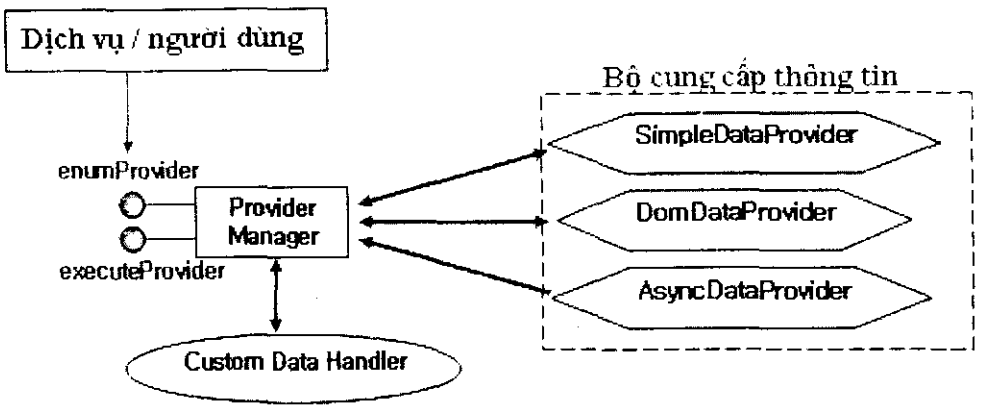
serviceDataName (QName): Là XML Qualified Name trong Service Data Elements sẽ được tạo ra từ output của provider. Chú ý: Tham số này có thể không được dùng đến nếu một custom callback handler được dùng để xử lí dữ liệu kết quả.

refreshFrequency (integer): Là tần số làm tươi, tính bằng giây, với mỗi tần số này, bộ cung cấp sẽ được thực hiện theo tần số này.

async (Boolean): Chỉ rõ rằng, provider nên chạy không đồng bộ, nếu có thể.

- Các bộ xử lý dữ liệu tuỳ biến - Custom Data Handlers

Bộ quản lí cung cấp thông tin làm nhiệm vụ lấy kết quả dạng XML từ bộ cung cấp, tạo một SDE mới, và sau đó thêm SDE vào dữ liệu của dịch vụ.



Hình 5-24: Cơ chế của các bộ cung cấp thông tin

5.3. Cài đặt dịch vụ thông tin cho hệ thống BKGrid

Cũng như mọi hệ thống lưới, dịch vụ thông tin trong hệ thống BKGrid 2005 có vai trò cung cấp các thông tin về môi trường lưới cho các thành phần trong hệ thống, cụ thể là bộ môi giới tài nguyên. Như ta đã biết, yêu cầu được gửi tới bộ môi giới bao gồm các ứng dụng lưới kèm theo là các yêu cầu về tài nguyên hệ thống. Dịch vụ thông tin cung cấp cho bộ môi giới các thông tin về hệ thống như máy nào đang sẵn sàng, máy nào ngừng hoạt động, thông tin cơ bản về các máy đang sẵn sàng như tải CPU, bộ nhớ còn trống, đĩa cứng còn trống...

Như vậy, nếu không có dịch vụ thông tin, bộ môi giới tài nguyên không thể có dữ liệu đầu vào để thực hiện việc công việc của mình, hiệu năng của hệ thống ở mức độ nào đó trở nên rất nghèo nàn.

Ngoài ra đối với người quản trị hệ thống lưới việc theo dõi trạng thái các nút lưới, các tài nguyên trên nút lưới để có những phương án hợp lí là điều khá quan trọng. Dịch vụ thông tin sẽ cung cấp một giao diện thuận tiện cho việc theo dõi tài nguyên với các thông tin động được cập nhật liên tục.

5.3.1. Cài đặt dịch vụ thông tin

5.3.1.1. Cài đặt dịch vụ thông tin cung cấp thông tin hệ thống

Dịch vụ SystemService cung cấp các thông tin hệ thống:

- Định nghĩa dịch vụ SystemService sử dụng ngôn ngữ lập trình Java

File mã nguồn của dịch vụ SystemService được viết bằng Java. Trong file này, các hàm public phải có đầu vào, đầu ra tuân theo những gì được đặc tả trong giao diện hoặc trong file GWSDL. Để là một file mã nguồn của một dịch vụ lưới, lớp thực thi phải kế thừa từ lớp GridServiceImpl hoặc phải cài đặt từ giao diện có tên OperationProvider.

Sau đây là các lớp đã được cài đặt trong dịch vụ SystemService:

+ Lớp Host

Cung cấp hàm

getHost(): ID của máy

+ Lớp OperatingSystem

Cung cấp các hàm

getName(): Tên hệ điều hành

getVersion(): Phiên bản hệ điều hành

getRelease(): Bản cập nhật hệ điều hành

+ Lớp Memory

Cung cấp các hàm

getRamAvaiable(): Dung lượng Ram còn trống

getTotalRam(): Tổng dung lượng Ram

getVirtualAvaiable(): Dung lượng bộ nhớ ảo còn trống

getVirtual(): Tổng dung lượng bộ nhớ ảo

+ Lớp NetworkInformation

Cung cấp các hàm

getIP(): Địa chỉ IP

getMTU(): Maximum Transmission Unit

getInboundIP(): Địa chỉ IP tại giao diện vào

getOutboundIP(): Địa chỉ IP tại giao diện ra

+ Lớp Proccesor

Cung cấp các hàm

getCacheL2(): Dung lượng bộ nhớ Cache L2

getClockSpeed(): Tốc độ đồng bộ

getModel(): Mô hình (Ví dụ: Pentium III)

getDescription(): Mô tả (vài dòng text)

getVendor(): Nhà cung cấp

getVersion(): Phiên bản

+ Lớp CPULoad

Cung cấp các hàm

get15Minute(): Thông tin tải CPU trước đó 15 phút

get5Minute(): Thông tin tải CPU trước đó 5 phút

get1Minute(): Thông tin tải CPU trước đó 1 phút

+ Lớp FileSystem

Cung cấp các hàm

getSpaceAvaiable(): Không gian còn trống

getReadOnly(): Xem file này có phải là ReadOnly hay không

getType(): Kiểu

getSize(): Kích thước

getRoot(): Đường dẫn từ Root

- Định nghĩa giáo diện thông qua file GWSDL

GWSDL dùng XML để diễn tả các hàm, các kiểu dữ liệu mà dịch vụ lưới tương ứng cung cấp. Khi chương trình client yêu cầu một dịch vụ lưới, nó sẽ đọc file GWSDL này để biết xem dịch vụ lưới đó cung cấp những hàm nào, với đầu vào, đầu ra ra sao để biết cách gọi các hàm đó sau này.

Cụ thể, trong file GWSDL của dịch vụ SystemService có định nghĩa các kiểu dữ liệu phức tạp sau:

```
<complexType name="OperatingSystemType">
</complexType>

<complexType name="ProcessorType">
</complexType>

<complexType name="MainMemoryType">
</complexType>

<complexType name="NetworkAdapterType">
</complexType>

<complexType name="LoadType">
</complexType>

<complexType name="FileSystemType">
</complexType>
```

- File triển khai dịch vụ WSDD

Sau khi đã đặc tả giao diện của dịch vụ lưới thông qua file GWSDL, viết mã nguồn trong file Java, người phát triển cần viết một file WSDD (Web Service Deployment Descriptor). File này viết bằng XML, trong nó chỉ ra các tham số khi triển khai một service như: đường dẫn đến dịch vụ, tên dịch vụ, tên thực thể của dịch vụ,...

- Viết chương trình client sử dụng dịch vụ lưới

Vấn đề cuối cùng cần quan tâm đối với dịch vụ lưới chính là làm thế nào để viết chương trình client truy nhập đến những dịch vụ đã có. Điều này thật đơn giản, chỉ cần biết được

GSH tương ứng với dịch vụ lưới mà ta cần sử dụng, biết các hàm và quy cách truyền tham số, nhận kết quả. Trong chương trình client cần import lớp giao diện của dịch vụ lưới tương ứng và sử dụng các câu lệnh cần thiết để tham chiếu tới giao diện này.

5.3.1.2. Cài đặt dịch vụ thông tin cung cấp thông tin dịch vụ

Tương tự như vậy, ta viết dịch vụ ServiceInfo lấy các Interface, các Service Data Name của dịch vụ. Để lấy các thông tin về dịch vụ, ta phải truy cập vào dịch vụ và lấy các SDE của dịch vụ đó.

Dịch vụ cung cấp hai hàm: getInterface() và getServiceDataName()

Interface: cung cấp thông tin về các giao diện của dịch vụ

Service Data Name: cung cấp tên các dữ liệu dịch vụ.

Chương 6: Dịch vụ môi giới tài nguyên

Công nghệ Agent được ứng dụng trong hệ thống BKGrid 2005, với chức năng chính là môi giới các tài nguyên trong lưới, thu thập các thông tin động cũng như thông tin tĩnh về dịch vụ và hệ thống qua dịch vụ thông tin, sau đó cung cấp cho bộ quản lý tài nguyên để tiến hành lập lịch. Ngoài ra một ứng dụng rất quan trọng khác của Agent là quản lý từng tài nguyên và dịch vụ đơn lẻ. Từng tài nguyên và dịch vụ sẽ có một Agent đại diện và Agent này sẽ chịu trách nhiệm thay mặt người sở hữu tài nguyên đó định ra các chính sách sử dụng, thương lượng với Agent người dùng.

6.1. Tổng quan về công nghệ Agent

6.1.1. Giới thiệu Agent

Agent có thể hiểu là một phần mềm có cài đặt các hoạt động tự động. Như vậy, lập trình Agent chính là lập trình hướng đối tượng kết hợp với trí tuệ nhân tạo. Agent có các thuộc tính sau:

- Tính tự trị: Một Agent có thể hoạt động mà không cần sự can thiệp trực tiếp của con người hoặc các thành phần khác. Điều này đòi hỏi trong các phần mềm Agent phải cài đặt các hoạt động có chu kì để tạo ra khả năng thực thi tự phát trong đó Agent phải có khả năng thực hiện các hành động có mức ưu tiên khác nhau hay độc lập và cuối cùng đem lại kết quả mong muốn cho con người.
- Khả năng cộng tác: Các Agent có thể tương tác với các Agent khác và/hoặc con người. Sự hợp tác giữa người sử dụng và Agent có thể được miêu tả như một hình thái cộng tác trong việc tạo nên hợp đồng. Người sử dụng chỉ rõ những hành động nào sẽ được thực hiện và Agent chỉ rõ nó có thể làm cái gì và cung cấp cho người dùng kết quả. Sự hợp tác giữa các Agent là một cơ chế mà thông qua đó các Agent trao đổi nhận thức của chúng, các kế hoạch cộng tác giữa chúng để giải quyết những vấn đề lớn hơn khả năng của mỗi Agent.
- Khả năng phản ứng: Các Agent có thể nhận thức được môi trường của chúng và phản ứng lại đúng lúc với sự thay đổi xảy ra trong môi trường đó. Các hành động của chúng được thực hiện như là kết quả của việc kích hoạt các luật hoặc của các hoạt động lặp đi lặp lại như: cập nhật cơ sở nội tại của Agent và gửi thông điệp tới các Agent khác trong môi trường.
- Khả năng di trú: Các Agent có thể di chuyển đến các môi trường khác, tìm kiếm thông tin cần thiết và trả lại kết quả mong muốn.
- Tính cá nhân: Mục đích của Agent là cho phép con người thực hiện các tác vụ tốt hơn. Nếu con người không làm tất cả các tác vụ giống nhau và thậm chí họ chia các tác vụ giống nhau

để thực hiện theo các cách khác nhau thì một Agent phải được "dạy" về tác vụ đó cũng như là cách thực hiện.

6.1.2. Phân loại Agent

Dựa trên các thuộc tính đã được trình bày ở trên ta có thể chia Agent thành các loại sau:

- Agent tự trị:

Các Agent này thường dùng trong môi trường phức tạp, không ổn định. Chúng cảm nhận và hành động một cách tự trị bằng việc nhận thức các mục đích và tác vụ.

- Agent giải trí:

Agent này dùng để mô phỏng thế giới thực, cung cấp các dịch vụ giải trí cho người sử dụng ví dụ như game, các sản phẩm film/video.

- Agent ủy quyền:

Các Agent này thực hiện một tập thao tác đại diện cho một người sử dụng hoặc các chương trình khác một cách độc lập.

- Agent giao diện:

Agent giao diện như một người phụ tá cùng cộng tác với người sử dụng trong cùng một môi trường. Agent quan sát và giám sát các hành động của người sử dụng trên giao diện, "học" điều đó và đưa ra cách tốt hơn để thực hiện các tác vụ đó. Agent giao diện "học" để trợ giúp tốt hơn cho người sử dụng theo 4 cách khác nhau:

Bằng cách quan sát và bắt chước người sử dụng (học từ người sử dụng).

Thông qua các phản hồi tích cực và tiêu cực nhận được từ người sử dụng (học từ người sử dụng).

Bằng việc nhận được những lời hướng dẫn rõ ràng từ phía người sử dụng (học từ người sử dụng).

Bằng việc hỏi lời khuyên từ các Agent khác.

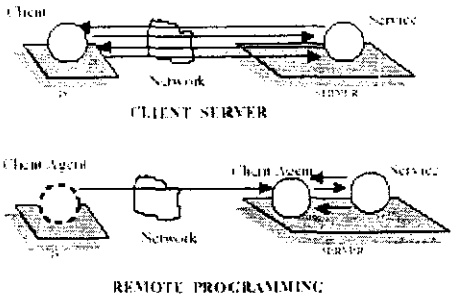
Sự hợp tác của Agent giao diện với các Agent khác chỉ giới hạn trong việc hỏi các thông tin.

- Agent cộng tác:

Các Agent này nhấn mạnh tính tự trị và sự cộng tác (với các Agent khác) để thực hiện các tác vụ. Để có được sự cộng tác với các Agent khác thì các Agent này phải đàm phán để đạt sự thoả thuận về cơ chế cộng tác.

- Agent động:

Các Agent động là các quy trình phần mềm tính toán có khả năng "di chuyển" trên các mạng diện rộng, như Internet, tương tác với các máy chủ phân tán , thu thập thông tin cho người sử dụng và trở về sau khi đã thực hiện các công việc cho người sử dụng. Thuộc tính của khả năng động đã đưa đến một khái niệm về lập trình từ xa (remote programming) trong đó các Agent tham gia tương tác có vai trò đúp: server (cung cấp dịch vụ) và client (khai thác dịch vụ).



Hình 6-1: Sự khác nhau giữa mô hình Client – Server và Remote Programming

- Agent phản ứng:

Các Agent phản ứng miêu tả một dạng đặc biệt của Agent. Chúng không xử lý mô hình bên trong đặc trưng của môi trường, thay vào đó chúng phản ứng lại với sự kích thích đối với trạng thái hiện tại của môi trường mà chúng được nhúng ở bên trong. Cấu trúc của các Agent phản ứng và sự giao tiếp giữa chúng nhìn chung rất đơn giản. Tuy nhiên, khi xem xét tổng quát hệ thống Agent ta có thể thấy được các tương tác phức tạp. Ta có thể đưa ra ba nguyên tắc cơ bản của các Agent phản ứng sau đây:

Tương tác động giữa các Agent không thông minh có thể dẫn tới sự phức tạp.

Một Agent phản ứng thường được xem như là một tập hợp các đối tượng hoạt động một cách độc lập và thực hiện các tác vụ cụ thể (ví dụ như cảm biến, mô tơ điều khiển, tính toán,..). Giao tiếp giữa các đối tượng này thường ở mức thấp.

Một Agent phản ứng có xu hướng hoạt động với việc biểu diễn gần với dữ liệu cảm biến thô, ngược lại với biểu diễn ở mức cao được sử dụng bởi các loại Agent khác.

6.1.3. Truyền thông giữa các Agent

6.1.3.1. Các giao thức

Truyền thông là cách để các Agent trao đổi thông tin, trên cơ sở đó chúng cộng tác với nhau để thực thi công việc. Giao thức truyền thông là nhân tố quan trọng quyết định chất lượng hợp tác giữa các Agent. Trong hệ đa Agent, các Agent trao đổi thông tin với nhau bằng nhiều cách: các Agent có thể trao đổi trực tiếp bằng các thông điệp, hoặc chúng có thể tự tổ chức trao đổi trong một hệ thống hợp nhất và giao tiếp thông qua các bộ môi giới Agent (Agent facilitator), hay chúng có thể quảng bá các thông điệp. Một cách khác phổ biến hơn cho các Agent giao tiếp với nhau là thông qua một kho dữ liệu dùng chung (gọi là blackboard) trong đó các thông tin được gửi vào và lấy ra.

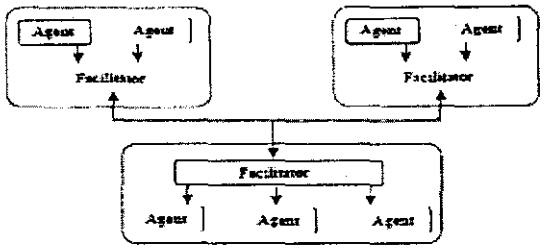
- Truyền thông trực tiếp

Truyền thông trực tiếp thông qua việc thiết lập các liên kết vật lý trực tiếp giữa các Agent khác nhau sử dụng một giao thức như TCP/IP bảo đảm an toàn cho các gói thông điệp. Các liên kết vật lý giúp cho các Agent có thể nhận thức được các Agent khác trong hệ thống. Địa chỉ của Agent được xác định hoặc thông qua thông điệp quảng bá nhận được từ các Agent khác hoặc thông qua một đối tượng kiểm soát. Đối tượng này giống như một giám đốc dịch

vụ mà mỗi Agent phải đăng kí khi tham gia vào hệ thống. Agent gửi thông điệp có thể nhận biết địa chỉ của Agent nhận bằng cách tra cứu ở đối tượng kiểm soát này. Việc đăng kí giống như một quá trình khởi tạo để giám đốc dịch vụ nhận biết về tất cả các Agent trong hệ thống. Theo tiêu chuẩn của FIPA 1997, bất cứ một hệ đa Agent nào theo đúng tiêu chuẩn kĩ thuật cũng phải có một giám đốc Agent (Agent Directory) chứa thông tin về tất cả các Agent trong một môi trường cụ thể và việc truy cập, định dạng các dịch vụ của các Agent. Cơ cấu giao thức truyền thông trực tiếp được sử dụng trong phần lớn các ngôn ngữ xây dựng Agent trong đó một Agent liên kết hội thoại với một Agent riêng biệt và biết được chính xác nó sẽ gửi thông điệp cho Agent nào.

- Hệ thống hợp nhất

Khi số lượng các Agent trong một hệ thống tăng lên thì chi phí và quá trình xử lý cho việc truyền thông trực tiếp trở nên khó khăn. Việc tập hợp các Agent trong một hệ thống hợp nhất có thể khắc phục được các vấn đề của truyền thông trực tiếp. Hình sau minh họa một hệ thống hợp nhất



Hình 6-2: Hệ thống hợp nhất

Theo hình trên, các Agent không truyền thông trực tiếp với nhau mà thông qua các Agent facilitator (môi giới Agent). Mỗi nhóm các Agent có một facilitator lưu giữ thông tin về nhu cầu cá nhân và khả năng của chúng. Các Agent có thể gửi các yêu cầu và thông tin về mức độ ứng dụng tới các facilititator và nhận về phản hồi. Facilitator sẽ sử dụng các thông tin nhận được từ các Agent để truyền những thông điệp mức ứng dụng và gửi chúng tới địa chỉ tương ứng. Như thế, các Agent trong hệ thống hợp nhất từ bỏ quyền tự trị của chúng và để cho các facilitator chịu trách nhiệm hoàn thành các nhu cầu của chúng. Tiêu chuẩn FIPA 1997 đã định nghĩa một Agent chuyên gia Domain Facilitator cho mỗi vùng và công việc của nó là duy trì một giám đốc Agent cho mỗi vùng tạo điều kiện thuận lợi cho việc truyền thông giữa các Agent của vùng đó.

- Truyền thông quảng bá

Trong trường hợp này, các thông điệp được truyền tới tất cả các Agent trong môi trường, hoặc Agent gửi không biết ai sẽ nhận được (giống như khi nó thông báo một nhiệm vụ và phải chọn từ tất cả các Agent có khả năng để thực hiện nhiệm vụ). Có 2 khả năng:

Agent có thể quảng bá một cách vật lý thông điệp tới tất cả các Agent trong hệ thống.

Nó có thể duy trì các liên kết truyền tin riêng rẽ tới tất cả các Agent trong hệ thống và gửi trực tiếp cho chúng các thông điệp (sử dụng giao thức TCP/IP).

Khi chiều dài của thông điệp lớn và có một số lượng lớn các Agent trong hệ thống thì việc sử dụng băng thông để truyền thông điệp là rất có ý nghĩa vì ở đây ta sử dụng các liên kết riêng rẽ để gửi nhiều thông điệp giống hệt nhau tới mỗi Agent nhận. Truyền thông quảng bá ngăn không quá tải mạng. Nó giúp cho hệ đa Agent trở nên độc lập, linh hoạt vì các Agent có thể tham gia hoặc ra khỏi một hệ thống mà không cần thông tin cho bất kỳ ai miễn là nó hoàn thành tất cả các tác vụ được giao hoặc các tác vụ đó có ảnh hưởng tới hoạt động của các Agent khác. Có hai cách tiếp cận chủ yếu và phổ biến trong truyền thông quảng bá là: *contract-net* và *specification-sharing*:

Trong cách tiếp cận *contract-net*, các Agent đang có nhu cầu được phục vụ sẽ gửi đi thông báo "sẵn sàng phục vụ" (quảng bá các thông điệp) tới các Agent khác. Agent nhận được thông báo này sẽ xem xét các yêu cầu đó và nếu có thể chấp nhận, nó sẽ gửi lời mời tới các Agent đang có nhu cầu phục vụ.

Trong cách tiếp cận *specification-sharing*, các Agent sẽ quảng bá các khả năng và nhu cầu của chúng và các Agent khác sẽ sử dụng cá thông tin này để đối chiếu với các nhu cầu và hoạt động của chúng xem có phù hợp hay không, và có thể đáp ứng được không.

- Hệ thống Blackboard

Blackboard là một kho mà các Agent viết các thông điệp, gửi các kết quả từng phần và thông tin thu được. Nó thường được chia thành từng mức độ tương ứng với các vấn đề xử lý, và một Agent làm việc ở mức độ cụ thể nào có thể truy nhập đến mức độ *blackboard* thích hợp. Theo cách này, dữ liệu được tổng hợp ở bất kỳ mức nào có thể giao tiếp với các mức độ cao hơn, trong khi các mục đích ở cấp độ cao hơn có thể được chọn lọc để điều khiển theo ý muốn của các Agent mức thấp.

6.1.3.2. Ngôn ngữ truyền thông giữa các Agent

Các Agent có khả năng tương tác với người sử dụng, với các dịch vụ hệ thống hoặc Agent khác. Để giao tiếp với nhau thì chúng cần ngôn ngữ. Knowledge Query and Manipulation Language (KQML) là ngôn ngữ dùng cho giao tiếp mức thông điệp giữa các Agent. Gần đây, tổ chức FIPA (Foundation for Inteligent Physical Agent) đã đưa ra chuẩn ACL của riêng nó: FIPA-ACL. FIPA-ACL được thiết kế nhằm giải quyết sự không thống nhất của rất nhiều phiên bản KQML khác nhau. FIPA-ACL trở thành một ngôn ngữ chuẩn phù hợp cho giao tiếp giữa các Agent.

- Những đặc điểm cơ bản của KQML

KQML được hiểu như là một giao thức định dạng thông điệp và xử lý thông điệp để cung cấp khả năng chia sẻ nhận thức (run-time knowledge) giữa các Agent. Các đặc điểm cơ bản của KQML là:

+ Các thông điệp KQML không trong suốt với nội dung mà chúng mang. Các thông điệp KQML không chỉ đơn thuần là những câu giao tiếp mà chúng còn trao đổi quan điểm về một nội dung (sự xác nhận, yêu cầu, truy vấn, câu trả lời cơ bản,...).

+ Cơ sở của ngôn ngữ là các *biểu hiện* (performative). Các biểu hiện định nghĩa các hành động (thao tác) cho phép các Agent truyền thông với nhau.

+ KQML hỗ trợ truyền thông điểm-điểm. Nghĩa là, tại một cấp độ Agent thì sự truyền thông như là sự chuyển thông điệp từ điểm này tới điểm khác.

+ KQML cho phép sự góp mặt của các Agent môi giới. Môi trường trong đó các Agent giao tiếp với nhau bằng KQML có thể có sự góp mặt của các Agent đặc biệt, gọi là các facilitator, đó là các Agent hỗ trợ cho các Agent khác các chức năng để giải quyết các vấn đề xảy ra trên mạng (liên hệ giữa các địa chỉ vật lý với các tên tượng trưng, việc đăng ký của các Agent, và/hoặc các dịch vụ được đưa ra và được tìm kiếm bởi các Agent, các dịch vụ giao tiếp tăng cường như forwarding, brokering, broadcasting...).

+ KQML định nghĩa một tập các mẫu thông điệp. Sau đây là một ví dụ về thông điệp KQML:

```
( ask-if
      :sender A
      :receiver B
      :language Prolog
      :ontology foo
      :reply- with id1
      :content "bar(a,b)" )
```

Trong ngôn ngữ KQML, ask-if là một biểu hiện. Một biểu hiện thiết lập các tham số. Trong ví dụ trên, một Agent có tên là A (:sender) hỏi Agent B (:receiver) bằng ngôn ngữ Prolog (:language) về trạng thái thật của "bar(a,b)" (:content). Bất kỳ câu trả lời nào cho thông điệp KQML này sẽ được nhận dạng bởi id1 (:reply-with). Ontology tên foo có thể cung cấp thêm thông tin về việc thông dịch :content.

Giả sử rằng nếu B không có khả năng thực hiện các thao tác mà A đề nghị trong thông điệp trước. Khi đó B sẽ trả lời A bằng biểu hiện tiếp theo như sau:

```
(sorry
:sender B
:receiver A
:in-reply-to id1
:reply-with id2 )
```

Agent B (:sender) sử dụng *biểu hiện* sorry để thông tin cho A(:receiver) biết rằng nó không thể thực hiện việc xác định bar(a,b). Agent A sẽ biết được chính xác thông điệp này muốn nói đến sự xác định nào bởi vì B sử dụng tham số: in-reply-to với giá trị id1.

- Cú pháp chuỗi KQML

Một *biểu hiện* (ví dụ như một thông điệp KQML) được biểu diễn bằng chuỗi ASCII sử dụng cú pháp được định nghĩa trong BNF. Các ký hiệu này có những ưu điểm:

Con người có thể đọc được.

Đơn giản cho các chương trình có thể phân tích cú pháp.

Dễ dàng di chuyển giữa các tiến trình xử lý thông điệp.

Các tham số trong *biểu hiện* được chỉ dẫn bằng các từ khoá, phải bắt đầu bằng một dấu hai chấm (:) và đặt trước một giá trị tham số thích hợp. Chúng có thứ tự độc lập. Bảng sau chỉ ra các từ khoá và ý nghĩa của chúng:

Từ khoá	ý nghĩa
:sender	Người gửi hiện tại của *biểu hiện*
:receiver	Người nhận hiện tại của *biểu hiện*
:reply-with	Nhãn mong đợi cho câu trả lời của thông điệp hiện tại
:in-reply-with	Nhãn mong đợi trả lời cho câu trả lời của thông điệp trước (giống với giá trị :reply-with trong thông điệp trước)
:language	Tên của ngôn ngữ thể hiện trong :content
:ontology	Tên của ontology được thừa nhận trong :content
:content	Thông tin về những gì mà *biểu hiện* thể hiện quan điểm
:from	Nguồn gốc *biểu hiện* trong :content khi thông điệp được sử dụng
:to	đích đến của thông điệp

- Các biểu hiện trong KQML

Có thể phân các biểu hiện thành các loại sau:

Các *biểu hiện* ngôn từ: đây là các *biểu hiện* được sử dụng trong ngữ cảnh trao đổi thông tin và nhận thức giữa hai Agent.

Các *biểu hiện* về sự can thiệp và cơ cấu của cuộc hội thoại: vai trò của biểu hiện này là can thiệp vào một cuộc hội thoại thông thường. Một cuộc hội thoại thông thường diễn ra như sau: Agent A gửi một thông điệp KQML (bắt đầu cuộc nội thoại) và Agent B trả lời ngay khi có câu trả lời. Các *biểu hiện* kiểu này hoặc là sớm gián đoạn cuộc hội thoại (error, sorry) hoặc là không quan tâm đến giao thức ngầm định này (standby, ready, next, rest, và discard).

Các *biểu hiện* về mạng và môi giới: các *biểu hiện* này cho phép các Agent tìm kiếm các Agent khác để thực hiện các yêu cầu của chúng.

Bảng sau đây là một số biểu hiện trong ngôn ngữ truyền thông điệp KQML với Agent gửi là S, Agent nhận là R, VKR là cơ sở trí thức ảo (Virtual Knowledge Base).

Biểu hiện	ý nghĩa
ask-if	S muốn biết trong trong VKB của R có :content không
Eos	S kết thúc chuỗi trả lời cho biểu hiện stream-all bằng biểu hiện này
Tell	S muốn nói cho R biết :content có trong VKB của S
Untell	S nói cho R biết :content không có trong VKB của S
Deny	S nói cho R biết :content là sai
Insert	S yêu cầu R thêm :content vào VKB của R
Uninsert	S yêu cầu R huỷ bỏ hành động của insert trước
delete-one	S muốn R xoá :content tương ứng trong VKB của R
delete-all	S muốn R xoá tất các câu có dạng :content trong VKB của R
Undelete	S muốn R huỷ bỏ hành động của biểu hiện delete trước
Advertise	S muốn R biết S có thể và sẽ xử lý những thông điệp giống như :content
Error	S coi thông điệp trước đó gửi từ R là lỗi
Sorry	S hiểu thông điệp của R nhưng nó không thể đáp ứng yêu cầu của S

6.1.4. Các ứng dụng của Agent

Agent được ứng dụng trên phạm vi rộng lớn. Các ứng dụng của Agent được phát triển trong nhiều lĩnh vực như: sản xuất, giải trí và thương mại điện tử.

- Các ứng dụng trợ giúp người sử dụng

Các hệ thống này làm việc với người sử dụng cuối (end-user) để nâng cao năng suất và giúp người sử dụng dễ dàng hơn trong việc sử dụng hệ thống tính toán phức tạp. Nhìn chung, chúng giao tiếp với người sử dụng để giúp họ quản lý nhật trình và thư điện tử, trợ giúp bộ nhớ. Chúng có thể giao tiếp với các Agent khác để thu thập thông tin. Chúng khác so với các giao diện người sử dụng chuẩn ở chỗ chúng được trao quyền để hành động ít

nhất là bán tự động và không chỉ là một công cụ để con người sử dụng và điều khiển. Có thể kể đến một vài ứng dụng trợ giúp sau:

Hệ thống học về profile của người sử dụng.

Hệ thống giao diện đa hình thái.

Các ứng dụng số cá nhân hoặc các ứng dụng giao tiếp thông minh cá nhân (ví dụ : thư ký điện thoại số).

- Các ứng dụng phục hồi thông tin

Hệ thống này liên quan đến tất cả các dịch vụ giúp đỡ người sử dụng tìm kiếm nhanh và dễ dàng các thông tin mà họ yêu cầu. Điều đó có thể thực hiện được bởi một tập hợp các Agent bao gồm:

Danh bạ các dịch vụ (các trang trắng và trang vàng).

Hướng dẫn cơ sở dữ liệu.

Sự môi giới thông tin.

Chỉ dẫn truyền thông.

- Các ứng dụng quản lý dịch vụ

Ví dụ như:

Các dịch vụ đa phương tiện.

Các dịch vụ mua/bán (ví dụ thông tin, nguyên liệu của hàng hoá).

Các dịch vụ quản lý mạng thông minh.

- Các ứng dụng quản lý kinh doanh

Bao gồm:

Các dịch vụ tài chính.

Thương mại điện tử.

Quản lý luồng công việc.

Sự tự động hoá văn phòng.

Agent cũng rất phát triển trong lĩnh vực sản xuất công nghiệp như : những Agent điều khiển robot hay tương tác người máy.

- Ứng dụng vào lưới tính toán

Trong môi trường lưới, Agent có thể được áp dụng vào tìm kiếm và thương lượng tài nguyên. Tiềm năng của Agent trong tính toán lưới sẽ được tìm hiểu chi tiết trong phần sau.

6.2. Hệ đa Agent, điều phối hoạt động trong hệ đa Agent

6.2.1. Mô hình hệ đa Agent

Hệ đa Agent (Multi Agent System - MAS) được định nghĩa như là một môi trường bao gồm nhiều Agent, mà về bản chất là độc lập với nhau, phân tán và có thể không đồng nhất, cùng nhau giải quyết các công việc vượt quá khả năng của từng Agent riêng rẽ. Nghiên cứu trong

hệ đa Agent chủ yếu đề cập đến cách kết hợp thông minh giữa các Agent, cách chúng kết hợp về tri thức, mục đích, kỹ năng, và kế hoạch hoạt động cùng nhau hoặc để giải quyết các vấn đề. Sau đây ta sẽ đưa ra một cái nhìn tổng quát về hệ đa Agent.

6.2.1.1. Môi giới dịch vụ (Directory Facilitator - DF)

Một DF cung cấp *dịch vụ những trang vàng* cho các Agent. Nó có nhiệm vụ duy trì, cập nhật thông tin về các Agent. Nó đóng vai trò cung cấp những thông tin thiết yếu của các Agent trong AP (Agent Platform). Có ít nhất một DF trong một AP (gọi là DF mặc định). Tuy nhiên, một AP có thể hỗ trợ nhiều DF khác nhau.

Mỗi Agent muốn quảng bá dịch vụ của nó cho những Agent khác thì nó phải tìm được DF thích hợp, yêu cầu đăng ký (registration) với DF đó, sau đó gửi những thông tin mô tả dịch vụ của mình tới DF. Agent sau khi đăng ký một dịch vụ nào đó với DF, và một Agent khác thông báo muốn sử dụng dịch vụ này thì Agent ban đầu hoàn toàn có quyền từ chối thực hiện dịch vụ mà nó đã quảng cáo. Một dịch vụ được đăng ký với DF có những tham số bắt buộc mô tả về dịch vụ như: kiểu dịch vụ, tên dịch vụ, địa chỉ dịch vụ, các thuộc tính của dịch vụ.... Khi không còn cung cấp dịch vụ đó nữa, Agent hoàn toàn có thể từ chối thực hiện dịch vụ và khi đó thường là nó sẽ gửi yêu cầu ngừng đăng ký dịch vụ (deregistration) tới cho DF, hoặc có thể yêu cầu DF chỉnh sửa (modify) thông tin dịch vụ đã đăng ký.

Một Agent tìm kiếm một dịch vụ nào đó sẽ gửi thông tin tiềm kiếm tới DF. DF không bảo đảm thông tin nó cung cấp là chính xác, vì nó không có một hạn chế nào đối với những thông tin do các Agent đăng ký với nó. Tuy nhiên, DF có quyền có giới hạn truy xuất vào thư mục mô tả thông tin riêng của các Agent và kiểm tra tất cả các quyền truy xuất của Agent nhằm cập nhật thông tin về trạng thái Agent. DF ngầm định của AP có một AID như sau:

```
(Agent-identifier
   :name df@hap
   :addresses (sequence hap_transport_address))
```

Trong đó, trường :name là tên của DF, trường :addresses là địa chỉ của Platform chứa DF đó.

Mỗi DF có khả năng thực hiện một số hàm chức năng sau:

Register (nhận đăng ký dịch vụ của một Agent).

Deregister (ngừng đăng ký dịch vụ đối với một Agent).

Modify (sửa thông tin về dịch vụ của một Agent).

Search (tìm kiếm dịch vụ của Agent).

Trong mỗi DF có một cơ chế tìm kiếm. Đầu tiên, DF ưu tiên tìm kiếm cục bộ trong cơ sở dữ liệu của nó, sau đó sẽ mở rộng phạm vi tìm kiếm đến những DF khác nếu cần thiết và được phép. Cơ chế tìm kiếm được thực hiện theo giải thuật tìm kiếm sâu. Khi một DF nhận một yêu cầu tìm kiếm, nó có thể gọi đến những DF khác đã đăng ký với nó để nhờ tìm kiếm giúp.

Thông điệp yêu cầu tìm kiếm thông tin sẽ được gửi tiếp đi (forward) nếu tham số max-depth còn lớn hơn 1 và không có yêu cầu tìm kiếm khác cùng search-id có độ ưu tiên cao hơn. Nếu yêu cầu tìm kiếm được forward thì sẽ tuân theo luật sau: Nó sẽ không thay đổi giá trị của tham số search-id khi nhân rộng phạm vi tìm kiếm. Giá trị search-id nên là duy nhất. Trước mỗi lần nhân rộng phạm vi tìm kiếm, giá trị độ sâu tìm kiếm (max-depth) sẽ bị trừ đi một đơn vị.

6.2.1.2. Hệ điều hành Agent (Agent Management System- AMS)

Một AMS là một thành phần bắt buộc của mỗi AP, AMS tồn tại duy nhất trong mỗi AP và chịu trách nhiệm quản lý hoạt động của AP như tạo các Agent, xoá các Agent, theo dõi sự di trú của các Agent (trường hợp các Agent động- mobile Agent).

AMS chính là sự biểu hiện quyền lực của AP. Một AMS có thể yêu cầu một Agent thực hiện một chức năng quản lý nào đó, ví dụ như quit (ngừng mọi hoạt động trong AP) và AMS có quyền "bắt buộc" nếu Agent từ chối yêu cầu.

AMS duy trì bản danh sách tất cả các Agent hiện tại của nó, bao gồm cả định danh của Agent (Agent Identification- AID). Như vậy, một Agent nằm trong AP có nghĩa là trước đó nó phải đăng ký với AMS.

Mô tả về Agent có thể bị thay đổi vào bất cứ lúc nào vì một lý do nào đó. Quyền thay đổi này thuộc về AMS. Vòng đời của một Agent sẽ kết thúc khi nó gửi thông điệp ngừng đăng ký (deregistration) tới AMS. Sau khi ngừng đăng ký, AID của Agent có thể bị xoá khỏi AMS và những Agent khác có thể đăng ký với AMS để lấy AID này.Những mô tả về Agent có thể được AMS truy xuất trong thư mục của nó.

AMS có những chức năng cơ bản sau:

register: cho phép một Agent đăng ký vào AP.

deregister: cho phép một Agent rời khỏi AP.

modify: thay đổi thông tin của một Agent.

search: tìm kiếm thông tin một Agent.

get-descripstion: lấy thông tin về một Agent.

Ngoài ra, quyền lực của AMS trong hệ thống đối với Agent thể hiện qua những quyền sau:

Tạm dừng hoạt động của một Agent.

Dừng hẳn một Agent.

Tạo Agent.

Phục hồi Agent.

Yêu cầu Agent thực hiện một tác vụ.

Quản lý tài nguyên của AP.

AMS của một AP có một AID:

```
(Agent-identifier
```

```
:name ams@hap
:addresses (sequence hap_transport_address))
```

Trong đó, trường :name là tên của AMS,

6.2.1.3. Dịch vụ truyền thông điệp (Message Transport Service - MTS)

Dịch vụ truyền thông điệp (MTS) có nhiệm vụ truyền thông điệp giữa các Agent trong một AP và đến các Agent cư trú ở AP khác. Tất cả các Agent FIPA đều có thể sử dụng ít nhất một MTS và chỉ các thông điệp có địa chỉ mới có thể được gửi đến MTS.

6.2.1.4. Nền Agent (Agent Platform - AP)

AP cung cấp một cơ sở hạ tầng vật lý phục vụ cho Agent. AP gồm hệ điều hành, những tiến trình, những phần mềm hỗ trợ Agent, những thành phần trong một hệ đa Agent như: DF, AMS, MTS và các Agent. Cần chú ý rằng những Agent nằm trong một AP không có nghĩa là Agent đó nằm trong hệ thống vật lý đó, mà Agent có thể nằm ở vùng khác, nhưng đăng ký với AMS.

6.2.1.5. Phần mềm (Software)

Phần mềm ở đây gồm những chức năng mà Agent có thể truy cập đến ví dụ như Agent có thể truy xuất tới phần mềm, thêm dịch vụ mới, yêu cầu một giao thức truyền thông mới, một thuật toán,...

6.2.1.6. Định danh của Agent (Agent Identification - AID)

Mỗi Agent trong AP có một định danh riêng biệt. Mỗi Agent trong hệ thống được xác định bằng một định danh, đó là một tập các tham số gộp lại, gọi là Agent Identifier (AID). Một tập AID cung cấp những thông tin quan trọng, phù hợp với những yêu cầu trong hệ thống. Một AID bao gồm những tham số sau:

+ Tên (name): Là một tên đi liền với Agent, là duy nhất và có thể sử dụng nó để tham chiếu đến Agent thay cho địa chỉ. Một mẫu tên thường gặp là kiểu đặt tên giống như tên hòm thư, trong đó *CNTT* là tên gọi của Agent trong hệ thống, còn *Bachkhoa.com* là tên gọi của hệ thống (AP)

+ Địa chỉ (addresses): Là một tập những địa chỉ mà các thông điệp có thể được gửi tới. Địa chỉ ở đây là địa chỉ vật lý, và các Agent hỗ trợ cho nhiều phương thức giao tiếp có trong tập những địa chỉ có trong tham số address.

+ Tên chú giải (resolvers): Là tham số chứa tập những địa chỉ dịch vụ chú giải. Địa chỉ dịch vụ chú giải là một dịch vụ được cung cấp bởi AMS, thông qua hàm chức năng tìm kiếm search. Tham số resolver chứa một dãy những địa chỉ của các AID khác, mà thông qua chúng thông điệp có thể đến địa chỉ của các Agent hiện thời. Sau đây là ví dụ về một ví dụ của một mẫu tên với chú giải:

1. Giả sử Agent A muốn gửi một thông điệp cho Agent B có AID là :

```
(Agent-identifier
  :name AgentB@bar.com
  :resolvers (sequence
    (Agent-identifier
      :name ams@foo.com
      :addresses (sequence iiop://foo.com/acc))))
```

Và Agent A muốn biết các địa chỉ có thể truyền thông điệp cho Agent B.

2. Như vậy, Agent A có thể gửi một yêu cầu tìm kiếm tới Agent đầu tiên xác định trong tham số resolvers và nó thường là chính AMS. Trong ví dụ này, AMS nằm ở foo.com.

3.Nếu AMS ở foo.com và Agent B có đăng ký với nó thì nó sẽ trả lại một thông điệp kết quả chứa các mô tả của Agent B; nếu không, một thông báo "failed" sẽ được trả lại.

4.Dựa vào thông điệp kết quả, Agent A có thể trích ra tham số Agent-identifier của ams-Agent-description và sau đó dựa vào tham số addresses để xác định các địa chỉ truyền cho Agent B.

5.Agent A bây giờ có thể gửi thông điệp cho Agent B bằng việc chèn tham số addresses vào AID của Agent B.

6.2.1.7. Việc đăng ký Agent

Có 3 trường hợp một Agent có thể đăng ký với AMS:

+ Agent được tạo ra trong AP.

+ Agent di trú đến AP, đối với các AP này thường hỗ trợ Agent động.

+ Một Agent dứt khoát phải đăng ký với AP, giả sử rằng AP hỗ trợ viêc đăng ký động và sẵn sàng đăng ký một Agent mới. Việc đăng ký động là khi một Agent đã có một HAP (Home Agent Platform) nhưng vẫn muốn đăng ký đến một AP khác như một Agent địa phương.

Việc đăng ký Agent liên quan đến việc đăng ký một AID với AMS. Khi một Agent được tạo ra hoặc đăng ký động với một AP thì Agent đó phải đăng ký với AMS, ví dụ nó sẽ sử dụng hàm *register*. Trong ví dụ sau, một Agent được gọi là *discovery-Agent* đăng ký động với một AP có tên foo.com. Agent *discovery-Agent* được tạo ra trên AP có tên bar.com (đó là *discovery-Agent's* HAP) và yêu cầu đăng ký với AMS. Thông điệp mà Agent đó gửi đến AMS có dạng như sau:

```
(request
          :sender
      (Agent-identifier
          :name discovery-Agent@bar.com
          :addresses (sequence iiop://bar.com/acc))
```

```
        :receiver (set
      (Agent-identifier
         :name ams@foo.com
         :addresses (sequence iiop://foo.com/acc)))
         :ontology FIPA-Agent-Management
         :language FIPA-SL0
         :protocol FIPA-Request
         :content
  (action
  (Agent-identifier
       :name ams@foo.com
       :addresses (sequence iiop://foo.com/acc))
 (register
 (ams-description
       :name
  (Agent-identifier
       :name discovery-Agent@bar.com
       :addresses (sequence iiop://bar.com/acc))
       ...)))
```

6.2.2. Điều phối hoạt động trong hệ đa Agent

6.2.2.1. Khái niệm

Trong một hệ đa Agent chia sẻ chung một môi trường hoạt động, tương tác giữa các Agent sẽ tăng lên là một điều tất yếu. Một Agent có thể làm thay đổi một số đặc tính của môi trường, thay đổi này tác động đến các Agent khác ví dụ như nó có thể sử dụng một tài nguyên không chia sẻ của các Agent khác. Khi một quyết định của Agent này làm ảnh hưởng đến những gì mà các Agent khác có thể làm hoặc nên làm, các Agent trong hệ đa Agent này nên điều phối các quyết định của mình.

Định nghĩa 1: Điều phối là một cơ chế đảm bảo các Agent duy trì mối liên hệ giữa các hoạt động của nó (trình tự, thực thi...). Điều khiển các thông tin điều phối phải được thực hiện bởi 1 Agent thông minh.

Mức độ điều khiển là từ 0 (zero) đến tuyệt đối (total). Đối với Agent, không điều khiển tương ứng với tự chủ tuyệt đối, ngược lại điều khiển tuyệt đối tương ứng với không có tính tự chủ).

Định nghĩa 2: Trong một hệ đa Agent, điều phối là một quá trình mà trong đó mỗi Agent học về các hành động của nó và đoán trước các hành động của các Agent để cố gắng đảm bảo các hành động liên lạc với nhau trong một mối liên kết thống nhất. Liên kết có nghĩa các hoạt động của mỗi Agent đều được thực hiện tốt và không xung đột với các Agent khác.

6.2.2.2. Sự cần thiết của việc điều phối trong hệ đa Agent ?

- Ngăn chặn tình trạng hỗn loạn

Điều phối là một điều hết sức cần thiết vì trong các hệ thống hướng Agent, sự hỗn loạn là rất dễ xảy ra. Không một Agent nào có thể bao quát hết được môi trường, hệ thống mà nó thuộc vào. Điều này đơn giản không thực thi được vì mỗi cộng đồng Agent đều rất phức tạp. Ví dụ như một ông chủ không thể biết hết tất cả các hoạt động của tất cả các nhân công của mình. Mỗi Agent chỉ có khung nhìn cục bộ, do đó mục đích và kiến thức của chúng có thể xung đột nhau. Giống như bất kỳ một xã hội phức tạp nào, từng nhóm Agent phải được điều phối để đạt được mục đích chung.

- Thỏa mãn các ràng buộc chung

Thông thường mỗi nhóm Agent đều có những ràng buộc chung mà chúng phải thoả mãn nếu muốn công việc thành công. Ví dụ như một hệ thống đa Agent đang được xây dựng phải thoả mãn ràng buộc là nằm trong ngân sách định trước. Cũng tương tự như các Agent quản lý hệ thống mạng phải trả lời lỗi trong vòng vài giây và một số điều khác trong vòng vài giờ. Do đó các Agent cần điều phối hành vi của chúng để thoả mãn các điều kiện ràng buộc chung.

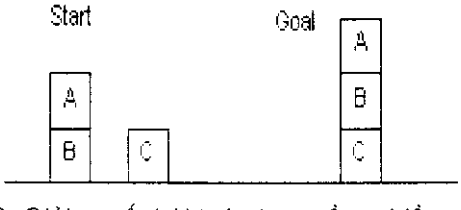
- Chia sẻ kinh nghiệm, nguồn tài nguyên và thông tin

Trong hệ đa Agent, các Agent có những khả năng và kiến thức khác nhau. Chúng có những nguồn về thông tin, tài nguyên, mức độ tin cậy, khả năng đáp ứng, các dịch vụ... khác nhau. Trong trường hợp này các Agent phải được điều phối cũng giống như trường hợp sau: Các chuyên gia về y học như bác sĩ gây mê, bác sĩ phẫu thuật, bác sĩ cấp cứu, y tá...đều cùng làm việc với nhau để cứu chữa cho các bệnh nhân bị tai nạn. Trong ví dụ này mỗi chuyên gia có kinh nghiệm trong lĩnh vực của mình nhưng họ đều chia sẻ thông tin chung về tình trạng của bệnh nhân, chia sẻ về nguồn tài nguyên như dụng cụ y tế.

- Độc lập giữa những hành động của các Agent (tránh xung đột)

Mục đích của các Agent thường hay phụ thuộc lẫn nhau. Hãy xét ví dụ :

Giả sử hai Agent phải giải quyết các khối công việc sau:



Hình 6-3: Giải quyết bài toán bao gồm nhiều công việc

Có hai Agent: Agent 1 có 2 khối công việc là A và B, Agent 2 có khối công việc là C. Mục đích của chúng là thực hiện một công việc có kết quả theo thứ tự A, B, C. Rõ ràng để thực hiện được mục đích này các Agent phụ thuộc vào nhau: Agent thứ 2 phải đợi Agent thứ nhất thực hiện xong thì nó mới có thể hoàn thành công việc của mình. Do đó, khi hành động của các Agent phụ thuộc vào nhau chúng cần phải được điều phối các hoạt động của mình.

- Đảm bảo hiệu năng cao

Ngay cả khi các chức năng của các Agent là độc lập nhau thì chúng vẫn cần phải điều phối vì thông tin của Agent này sẽ ảnh hưởng đến các Agent khác và kết hợp lại với nhau để giải quyết các bài toán nhanh chóng hơn, dễ dàng hơn.

Điều phối có thể bao gồm cả hợp tác, nhưng cần phải nhấn mạnh rằng hợp tác không phải là điều kiện cần thiết trong một tập hợp các Agent mà trong thực tế nó có thể là các hành vi không liên kết. Nếu các Agent không hiểu biết hết về nhau khi đó các hành vi không liên kết có thể là tốt. Điều phối cũng có thể không cần sự hợp tác. Ví dụ như có một người đang chạy về phía bạn nhưng bạn muốn thoát khỏi anh ta, khi đó bạn sẽ điều phối các hành động của mình đối với anh ta cho phù hợp mà không phải là hợp tác với anh ta.

Để đạt được sự điều phối, các Agent có thể phải liên lạc với nhau. Tuy nhiên, điều phối cũng có thể không cần sự liên lạc hay cung cấp các mô hình hành vi của mình cho nhau. Trong trường hợp này, sự điều phối đạt được bằng cách tổ chức. Song để điều phối giữa các Agent được dễ dàng, chúng nên hợp tác với nhau thông qua việc liên lạc, truyền thông để có thể biết được về mục đích, ý định, kết quả và trạng thái của nhau.

Ở trong môi trường phân tán điều phối còn có các tác dụng:

- Liên kết mạng: Tối đa hoá sự kết hợp tốt giữa các Agent trong hệ thống phân tán.

-Chỉ định nhiệm vụ và nguồn tài nguyên giữa các Agent.

- Nhận biết và giải quyết các mối xung đột giữa các mục đích, sự kiện, hiểu biết, khung nhìn và hành vi của các Agent.

- Quyết định về kiến trúc của một Agent được thiết lập.

Tuy nhiên danh sách các lý do trên để thấy được điều phối giữa các Agent là cần thiết vẫn chưa đầy đủ. Mức độ phối hợp được thể hiện trong phạm vi hoạt động của các Agent trong đó tránh thực hiện các hoạt động không cần thiết:

- Tránh livelock: Livelock là trường hợp mà hoạt động của các Agent được diễn ra liên tục nhưng không thu được sự tiến bộ nào.

- Tránh deadlock: Deadlock là trạng thái của sự việc mà các hành động sau đó của hai hay nhiều Agent là không thể thực hiện được.

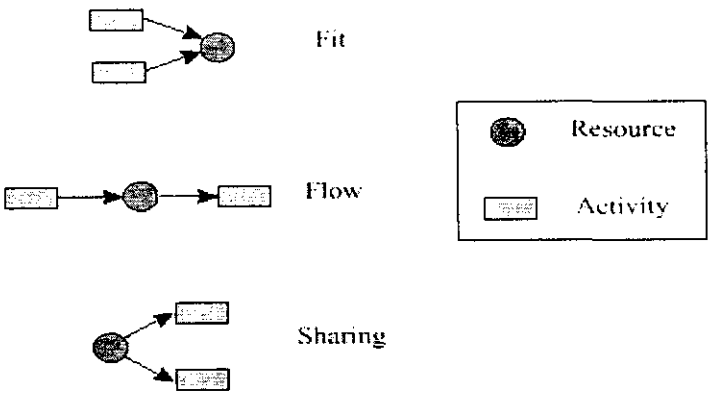
- Tối đa hoá khả năng của mỗi Agent.

- Đảm bảo khả năng an toàn cho hệ thống.

6.2.2.3. Các cơ chế điều phối trong hệ đa Agent

- Các cơ sở để phân loại cơ chế điều phối Agent:

Có rất nhiều tiêu chí để phân loại các cơ chế điều phối Agent như dựa vào sự phụ thuộc lẫn nhau giữa các thành phần, hoặc sự phụ thuộc giữa nhau giữa các hành động của Agent. Sau đây ta xét đến các cơ chế điều phối dựa trên sự phụ thuộc lẫn nhau giữa các hành động của Agent. Có ba loại phụ thuộc lẫn nhau giữa các hành động của các Agent là : fit, flow và sharing. Mô hình của chúng:



Hình 6-4: Các loại phụ thuộc lẫn nhau giữa các hành động.

+ Sự điều chỉnh phụ thuộc (Fit interdependency)

Mô tả tình huống trong đó có một vài hành động thực hiện cùng một thời điểm để tạo ra một sản phẩm hoặc một dịch vụ. Do đó, các đầu ra của các hoạt động này phải được điều phối để chúng phù hợp, tương thích với nhau để tạo ra một sản phẩm hoặc một dịch vụ liên kết. Khi một số sản phẩm đầu ra được tạo ra đồng thời, các hoạt động tạo ra chúng sẽ được điều phối đồng thời. Vì vậy các hoạt động điều phối trong quan hệ phụ thuộc fit interdependency chính là việc đặc tả đầu ra của các hoạt động như là một tiến trình.

+ Sự phụ thuộc luồng (Flow Interdependency)

Bao gồm một số các hành động được liên kết với nhau một cách tuần tự, đầu ra của một hành động này là đầu vào của một hành động khác. Sự điều phối các hoạt động này chính là việc đảm bảo có một luồng tài nguyên đáp ứng được xuyên suốt trình tự của các hoạt động. Nếu không các hoạt động sẽ không thể thực hiện vì không có tài nguyên đầu vào (nếu một hành động nào đó không có đầu ra đúng) hoặc các tài nguyên bị phủ lên nhau trong khi chờ được xử lý.

+ Phụ thuộc chia sẻ (Sharing Interdependency)

Nghĩa là hai hoạt động cùng dùng chung một nguồn tài nguyên. Sự điều phối ở đây chính là sự phân bổ tài nguyên đến các hoạt động này sao cho tạo ra giá trị kinh tế là lớn nhất hoặc đưa ra sự phân quyền ưu tiên cho các hành động để chúng sử dụng nguồn tài nguyên đó.

- Các cơ chế điều phối hệ đa Agent

+ Điều chỉnh qua lại (Mutual adjustment)

Các Agent chia sẻ thông tin và nguồn tài nguyên để đạt được một mục đích chung, mỗi Agent phải tự điều chỉnh hành vi của chúng cho phù hợp với hành vi của các Agent khác. Sự điều phối theo cơ chế điều chỉnh qua lại trong nhóm là ngang bằng nhau, không Agent nào có quyền điều khiển cao hơn và không Agent nào có quyền quyết định cho cả nhóm.

+ Giám sát trực tiếp (Direct Supervision)

Một Agent có một số mức độ điều khiển đối với một số Agent khác và nó có thể điều khiển thông tin, nguồn tài nguyên và hành vi. Thông thường, giám sát trực tiếp được thiết lập thông qua điều chỉnh qua lại.

+ Chuẩn hóa (Standardization)

Các thủ tục chuẩn hoá được thiết lập cho các Agent để tuân theo.

Trong cơ chế điều chỉnh qua lại được thực thi bằng sự chấp thuận

Trong cơ chế giám sát được thực thi bằng các yêu cầu.

+ Điều phối gián tiếp (Mediated Coordination)

Một Agent gián tiếp hoạt động như một nhân vật trung gian (facilitator- tìm kiếm thông tin), như một nhân vật môi giới (broker- cố vấn việc thương lượng nguồn tài nguyên), và một nhà giám sát (thực hiện một số mức độ giám sát trực tiếp). Nguyên tắc đầu tiên là bắt buộc nhưng những nguyên tắc khác là không bắt buộc.

+ Điều phối phản xạ (Reactive Coordination)

Agent phản ứng lại các tình huống đặc biệt với các hành vi đặc trưng. Với sự lựa chọn thích hợp từ các hành vi, các hoạt động mà nó học được khi phải phản ứng với các tình huống khác nhau, chúng sẽ lựa chọn được cách phản xạ tốt nhất.

- Mối quan hệ giữa sự phụ thuộc lẫn nhau của các hành động với các cơ chế điều phối

Với điều chỉnh tương hỗ, sự điều phối theo cơ chế này là không cần thiết đối với cả hai loại phụ thuộc flow và sharing bởi vì nó không cần điều phối các đầu ra cho các hoạt động này để phù hợp với các hoạt động khác (những sự phụ thuộc kiểu flow và sharing yêu cầu các chuẩn của đầu vào và đầu ra). Do đó, để phù hợp bằng kinh nghiệm người ta thấy rằng điều phối theo cơ chế điều chỉnh tương hỗ chỉ phù hợp với sự phụ thuộc lẫn nhau của các hành động theo kiểu fit.

Các cơ chế điều phối theo kiểu chuẩn, kiểu trung gian, kiểu phản xạ không thể áp dụng cho sự phụ thuộc giữa các hoạt động kiểu fit bởi vì các cơ chế điều phối này các yêu cầu đầu vào cần được chuẩn hoá giữa các hoạt động liên tiếp nhau. Các cơ chế điều phối trên theo kiểu định ra trước các tài nguyên được chuyển từ công việc này đến công việc khác, hoặc xác định giới hanh của số lượng các đầu ra được tạo ra, như vậy hoạt động của chúng chỉ phù hợp với các hoạt động phụ thuộc lẫn nhau kiểu flow và sharing.

Điều phối theo cơ chế giám sát trực tiếp có thể được áp dụng cho cả 3 loại phụ thuộc. Các giám sát viên có thể trực tiếp điều khiển cho các đầu ra của chúng phù hợp với nhau, các luồng xử lý được mịn hoặc ấn định các nguồn tài nguyên cho các hoạt động nếu chúng bị

dùng chung. Vì vậy, cơ chế giám sát trực tiếp là cơ chế điều phối được áp dụng rộng rãi nhất.

Tóm tắt các cơ chế điều phối ứng với những sự phụ thuộc lẫn nhau của các hành động:

Fit	Flow	Sharing
Giám sát trực tiếp	Giám sát trực tiếp	Giám sát trực tiếp
Điều chỉnh tương hỗ	Chuẩn hoá	Chuẩn hoá
	Điều phối trung gian	Điều phối trung gian
	Điều phối phản xạ	Điều phối phản xạ

- Một số điểm lưu ý

+ *Các cơ chế:* Điều chỉnh tương hỗ được thực hiện thông qua các tương tác ngang bằng, ngang cấp nhau. Giám sát trực tiếp sử dụng cách tiếp cận theo kiểu "chủ - tớ" (trong trường hợp lý tưởng không cho phép các tương tác ngang cấp). Một nhân vật trung gian hay các brokers điều chỉnh tương hỗ giữa các Agent và có thể thực hiện ở nhiều cấp độ giám sát khác nhau. Điều phối phản xạ phụ thuộc vào sự thích hợp của các mẫu hành vi đối với từng tình huống.

+ Các nhóm điều phối và Cấu trúc tổ chức (coordinated Groups and Organization Structure)

Điều khiển tương hỗ có thể thực hiện tốt trong các nhóm nhỏ nhưng số lượng kết nối thông tin và số lượng thông tin tăng nhanh theo kích thước.

Đối với các nhóm lớn, để việc điều phối có hiệu quả ta nên chia thành các nhóm nhỏ hơn nếu hầu hết các thông tin có thể chuyển sang dạng biểu diễn ở các nhóm nhỏ hơn được.

Điều phối một cách hiệu qủa đối với các nhóm nhỏ yêu cầu mỗi nhóm nhỏ có một điều phối viên, một giám sát viên, một nhà trung gian làm việc cùng nhau trong một hoặc nhiều nhóm điều phối.

Mỗi nhóm điều phối cũng có thể được điều phối trong nó như những nhóm Agent khác bởi các cơ chế như điều chỉnh qua lại hoặc giám sát trực tiếp, trung gian, hoặc các hành vi phản xạ.

Nếu chỉ điều phối bằng cơ chế giám sát trực tiếp sẽ tạo ra một cây phân cấp các nhóm Agent.

Nếu chỉ điều phối bằng cơ chế điều chỉnh qua lại sẽ tạo ra một cây giao cấp các Agent (không xuất hiện nhóm Agent).

6.2.2.4. Điều phối theo mô hình của sự uỷ thác, thoả hiệp, thoả hiệp cộng đồng và học

- Cơ sở của mô hình chung

Ta biết ba thành phần chính để thực hiện việc điều phối thành công:

+ Phải có cấu trúc dễ dàng cho các Agent tương tác với nhau một cách có dự đoán trước.

+ Phải mềm dẻo để các Agent có thể thao tác trong những môi trường động và có thể đối phó với những phần không liên kết của chúng và những khung nhìn không chính xác của cộng đồng.

+ Các Agent phải có đủ tri thức và khả năng lập luận để khai thác những cấu trúc và độ mềm dẻo sẵn có.

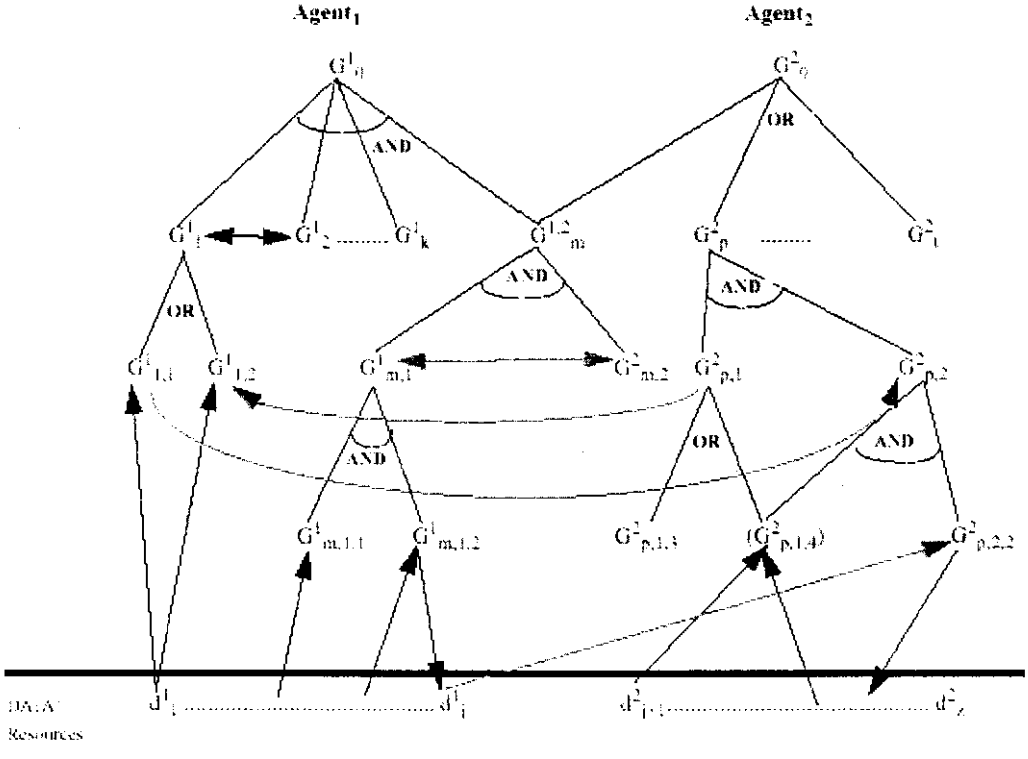
Những sự uỷ thác cung cấp cấu trúc cần thiết có những sự tương tác được đoán nhận trước, những sự thoả hiệp cung cấp sự mềm dẻo, linh động cần thiết để thao tác, thực hiện trong những môi trường động và những sự thoả hiệp cộng đồng cung cấp mức độ cần thiết cho việc hỗ trợ tương hỗ. Do đó:

Điều phối = Sự uỷ thác + Sự thoả hiệp + Sự thoả hiệp cộng đồng + Khả năng học

Những khẳng định ở trên được rút ra bằng những thử nghiệm phong phú về các tương tác cộng đồng xuất hiện như là kết quả của rất nhiều dạng mối quan hệ giữa các mục đích và sự phụ thuộc lẫn nhau của mục đích. Trong mỗi trường hợp đó, vai trò của sự uỷ thác, sự thoả hiệp, sự thoả hiệp cộng đồng lại càng nổi bật.

- Phân tích mô hình

+ Xem xét hệ đa Agent như việc tìm kiếm các mục đích phân tán.



Hình 6-5: Cây tìm kiếm mục đích phân tán

Mô hình trên mô tả mẫu của tìm kiếm mục đích phân tán (distribute goal search) như một tập điều khiển. Trong hình vẽ, hành động của $Agent_1$ và $Agent_2$ lần lượt giải quyết các mục đích G^1_0 và G^2_0 được biểu diễn dưới dạng cây tìm kiếm AND/OR. Cấu trúc cây theo kiểu cổ điển này cũng biểu diễn sự phụ thuộc lẫn nhau giữa các mục đích bởi vì chúng là chìa khoá cho sự điều phối trong các hệ thống đa Agent. Những nguồn tài nguyên cần thiết để đạt được các mục đích nhỏ nhất (các nút lá) cũng được thể hiện trong hình vẽ. Sự phụ thuộc lẫn nhau

có thể được thể hiện giữa các mục đích mức cao có quan hệ họ hàng với nhau, ví dụ như G^1_1 và G^1_2 hoặc chúng có thể xa nhau trong cây phân cấp như $G^1_{1,1}$ và $G^2_{p,2}$. Một trường hợp nữa, G^1_1 và G^2_p được gọi là các mục đích tương tác nhau nếu $G^1_{1,1}$ được sử dụng để giải quyết $G^2_{p,2}$. Những sự phụ thuộc gián tiếp giữa các mục đích là những sự phụ thuộc liên quan đến tài nguyên ($G^1_{m,1,2}$ và $G^2_{p,2,2}$ cùng dùng chung tài nguyên là d^1_j).

Sự phụ thuộc lẫn nhau có thể chia thành hai loại yếu hoặc mạnh (weak or strong) và là một chiều hoặc hai chiều. Mục đích A gọi là phụ thuộc mạnh vào mục đích B nếu muốn thực hiện mục đích A thì nhất thiết phải thực hiện mục đích B. Ngược lại, mục đích A gọi là phụ thuộc yếu vào mục đích B nếu mục đích A thực hiện có thể không cần đến mục đích B (tuỳ chọn). Trong hình trên G^1_1 phụ thuộc yếu vào G^1_1, $G^{1,2}_m$ phụ thuộc mạnh vào $G^1_{m,1}$. Sự phụ thuộc một chiều như $G^1_{1,1} \square G^2_{p,2}$ nghĩa là mục đích $G^2_{p,2}$ của Agent 2 phụ thuộc vào mục đích $G^1_{1,1}$ của Agent 1 (có thể là phụ thuộc mạnh hoặc yếu) nhưng $G^1_{1,1}$ không ảnh hưởng bởi $G^2_{p,2}$. Sự phụ thuộc hai chiều như giữa $G^1_{m,1}$ và $G^2_{m,2}$ các mục đích đều ảnh hưởng đến nhau.

Các mục đích nối (join goal) ví dụ như $G^{1,2}_m$ được coi là những hành động gắn liền với cộng đồng hoặc các mục tiêu mà một nhóm Agent được quyết định phải thực hiện theo nhóm. Mục đích nối chính là sự liên kết giữa các hành động riêng lẻ của mỗi Agent vào trong một mục đích chung làm tăng tính ảnh hưởng của mỗi Agent lên cộng đồng Agent. Ví dụ như $G^{1,2}_m$ là mục đích con của $Agent_1$ và $Agent_2$ và kết quả là $Agent_1$ thực hiện $G^1_{m,1}$ và $Agent_2$ thực hiện $G^2_{m,2}$. Nhờ các mục đích nối, tất cả các thành viên trong một đội đều có ảnh hưởng đến cả nhóm. Yếu tố thứ hai nói lên tầm quan trọng của mục đích nối là nó phân biệt giữa các *mục đích giống nhau và song song*. Ví dụ như cả x và y có mục đích là nấu mì ống do đó chúng có mục đích giống nhau, nhưng nếu cả hai Agent đều có mục đích là ăn mì ống (x có mục đích là x ăn mì ống và y có mục đích y ăn mì ống) thì chúng lại có các mục đích song song nhau. Hai loại mục đích này khác nhau là: các mục đích giống nhau làm tăng các mục đích nối nếu hai Agent quyết định làm chung trong một đội trong khi đó các mục đích song song lại tăng tính cạnh tranh và như vậy rõ ràng chúng không thể dẫn tới một hành động liên kết (hành động nối).

Toàn bộ cấu trúc mục tiêu không cần thiết phải cực kỳ tỉ mỉ để giải quyết bài toán từ đầu, nó có thể được xây dựng để giải quyết các tiến trình của một bài toán. Thực tế, xây dựng một đồ thị là một hoạt động xã hội rất phức tạp, liên quan đến thương lượng, giả thuyết, giải pháp với các sự xung đột hoặc có thể được giải quyết tập trung bởi một Agent. Sự xây dựng theo hướng từ trên - xuống với các mục đích ở mức cao, theo hướng từ dưới lên nếu được điều khiển bởi dữ liệu, hoặc hỗn hợp cả hai.

+ Điều phối trong việc tìm kiếm các mục đích phân tán.

Cách trình bày một hệ thống đa Agent như một bài toán tìm kiếm các mục đích phân tán cho phép các hoạt động được định nghĩa một cách rõ ràng, bao gồm:

Định nghĩa đồ thị mục đích (bao gồm nhận dạng và phân chia những sự phụ thuộc).

Ấn định các vùng đặc biệt của đồ thị thích hợp với các Agent.

Điều khiển việc quyết định diện tích của đồ thị để khai thác.

Thực hiện phân cấp mục đích.

Đảm bảo việc thực thi trong không gian tìm kiếm là thành công.

Một số các hoạt động trong các hoạt động trên được thực hiện trong một sự tương hỗ, một số thực hiện riêng lẻ. Quyết định để chọn cách tiếp cận phù hợp cho mỗi phase là một vấn đề của thiết kế hệ thống. Nó sẽ phụ thuộc vào:

Bản chất của lĩnh vực (trong các ứng dụng, các Agent có các trình độ, kinh nghiệm khác nhau do đó việc phân định các mục đích cho mỗi Agent là một công việc không đơn giản vì phụ thuộc vào khả năng hoàn thành công việc của mỗi Agent).

Dạng của các Agent có trong cộng đồng (ví dụ với các Agent tự chủ, tổng không gian tìm kiếm chính là hợp của các không gian tìm kiếm cục bộ và mỗi Agent làm việc dựa trên các mục đích riêng của chúng).

Yêu cầu các giải pháp đặc thù (để tăng tính đúng đắn của một kết quả quan trọng, các vùng tìm kiếm giống nhau trong không gian tìm kiếm có thể được ấn định một cách dư thừa với nhiều Agent trong khi đó phải thoả mãn yêu cầu tối ưu trong việc sử dụng Agent)

Giả thiết rằng có những sự phụ thuộc lẫn nhau giữa các mục đích hoặc giữa các yêu cầu tài nguyên của các Agent khác nhau, sự điều phối sẽ được yêu cầu nếu cộng đồng đó hoạt động trong một phương thức liên kết.

Bản chất của sự phụ thuộc xác định loại điều phối. Ví dụ nếu $Agent_1$ biết rằng $G^2_{p,2,2}$ yêu cầu nguồn tài nguyên d^1_j trước khi nó có thể bắt đầu (phụ thuộc mạnh, quan hệ một chiều) nên nó có thể quyết định thực hiện $G^1_{m,1,2}$ (để dinh ra tài nguyên cần thiết) trước khi $G^1_{m,1,1}$ nếu không có thêm thông tin nào khác tiêu biểu hơn để lựa chọn một trong hai. Thứ hai là, mối quan hệ giữa $G^1_{m,1}$ và $G^2_{m,2}$ có thể quy định cả hai hành động đều cần thực hiện đồng thời (phụ thuộc mạnh, quan hệ hai chiều) trong khi đó mỗi Agent thực hiện lần lượt. Cuối cùng là, nếu $Agent_1$ chọn $G^1_{1,1}$ như một phương thức để thoả mãn G^1_1 kết quả của hành động này sẽ cung cấp thông tin giá trị (phụ thuộc yếu, quan hệ một chiều) thông tin này có thể giúp cho $Agent_2$ thực hiện $G^2_{p,2}$. Biết được điều này, $Agent_2$ có thể bắt đầu với $G^2_{p,1}$. Tiếp theo, ta xét đến các thành phần trong mô hình này:

+ Sự uỷ thác

Cụm từ "uỷ thác" nghĩa là một sự đảm bảo hoặc một sự hứa hẹn. Các Agent có thể tạo ra sự đảm bảo cả về hành động và hiểu biết và những sự đảm bảo này có thể cả là quá khứ hoặc tương lai. Tuy nhiên với mục đích của sự điều phối, sự uỷ thác quan trọng nhất liên quan đến các hành động tương lai. Nếu một Agent tự uỷ thác cho mình thực hiện một hành động đặc biệt thì phải cung cấp cho nó các tình huống không thay đổi, nó sẽ cố gắng thực hiện vai trò của mình. Mỗi Agent sẽ quyết định thực hiện nhiệm vụ khi các ràng buộc của nó

vẫn thoả mãn có đủ tài nguyên giành cho hoạt động của nó. Nếu một Agent có nguồn tài nguyên vô hạn có thể cho bất cứ sự uỷ thác nào của nó thay đổi, định vị thì nó sẽ không có một ràng buộc nào. Tuy nhiên thực tế hầu hết nguồn tài nguyên là có hạn, bởi chúng bị ràng buộc bởi môi trường, một Agent bị giới hạn trong một số lượng hoặc loại uỷ thác mà chúng có thể tạo ra. Trong trường hợp này, mỗi Agent phải nhận thức được sự nhất quán với hiểu biết của nó. Sự uỷ thác cá nhân không được gây xung đột với các Agent khác, ví dụ một Agent không nên đảm bảo việc thực hiện hai mục đích đồng thời nếu chúng đều yêu cầu một nguồn tài nguyên không chia sẻ. Nếu các hành động dự định của một Agent được thực hiện trong một thể giới mà nó tin tưởng thì các trạng thái của công việc là đảm bảo.

Joint commitment (uỷ thác liên kết) có tất cả các tính chất của sự uỷ thác cá thể đã đề cập ở trên nhưng có thêm các điều kiện ràng buộc mà chúng liên quan đến nhiều hơn một Agent. Điều này có nghĩa là trạng thái của sự uỷ thác liên kết là phân tán. Ví dụ trạng thái của sự uỷ thác liên kết $G^{1,2}_{m}$ là sự phân tán giữa $Agent_1$ trong quá trình xử lý $G^{1}_{m,1}$ và $Agent_2$ trong quá trình xử lý $G^{2}_{m,2}$. Một cách lý tưởng tất cả các thành viên trong nhóm đều truy cập một trạng thái nhận thức chung liên quan đến uỷ thác liên kết, khi đó chúng sẽ đồng thời có những kinh nghiệm và những hiểu biết giống nhau, và như vậy là có sự đồng bộ giữa các thành viên trong nhóm. Tuy niên do hoạt động của nhóm được thực hiện một cách riêng biệt và không phải toàn thể nhóm. Do đó sự chia sẻ trạng thái nhận thức là không thể trừ khi tất cả các Agent sở hữu một cấu trúc đơn chung mà tất cả các bản ghi của sự hiểu biết của chúng về uỷ thác liên kết . Ví dụ như trong một nhóm tìm kiếm, nếu một Agent thoả mãn mục tiêu của cả nhóm và tìm ra khoản mục cuối cùng và đó là Agent duy nhất biết được uỷ thác liên kết đã được hoàn thành. Sau đó, Agent này thông báo với các Agent khác về thành quả của mình nghĩa là chúng chia sẻ thành quả chung, tuy nhiên trong khi đó những thành viên khác nhau dễ có sự hiểu biết khác nhau về sự uỷ thác liên kết. Với tất cả các khía cạnh khác, sự khác nhau giữa hai loại uỷ thác chính là số lượng.

+ Sự thoả hiệp

Một Agent hoàn thành sự uỷ thác của nó nếu được cung cấp những hoàn cảnh không thay đổi. Tuy nhiên, trong các kịch bản thực tế, các Agent ở trong một môi trường thay đổi theo thời gian, do đó mỗi Agent phải tự tiếp thu các thông tin mới, các Agent khác lại cố gắng tương tác với nó, và cứ như vậy... Do đó trong rất nhiều trường hợp, những sự hiểu biết của một Agent sẽ làm thay đổi giữa việc tạo ra một sự uỷ thác và nó thực tế thực hiện các quá trình, thời gian giữa hai sự kiện càng lớn càng có khả năng xuất hiện sự thay đổi.

Để thực hiện một cách thành công và thông minh, các Agent cần có các chính sách để kiểm soát việc thay đổi những sự uỷ thác của chúng. Những sự thoả hiệp này mô tả các trường hợp mà mỗi Agent nên cân nhắc kỹ những sự uỷ thác của chúng và chỉ định một chuỗi các hành động phù hợp với những sự uỷ thác còn lại cho đúng đắn. Khi đặc tả một sự thoả hiệp cần phải cân bằng giữa việc xem xét lại những sự uỷ thác (nó sẽ làm cho Agent phản hồi lại

một cách nhanh chóng với các tình huống thay đổi nhưng nó sẽ tốn thời gian để suy luận hơn là thời gian để thực hiện các công việc có ích) và không bao giờ xem xét lại những sự uỷ thác (Agent giành hết thời gian cho hoạt động nhưng lại phản ứng rất chậm với các tình huống thay đổi). Dựa vào các thí nghiệm về sự thoả hiệp tương ứng với sự thay đổi của môi trường rút ra được kết luận: Agent "dũng cảm" (không bao giờ kiểm tra lại kế hoạch của chúng) hoạt động tốt hơn Agent "bình thường". Theo kinh nghiệm đánh giá trong các tình huống thay đổi, sự thỏa hiệp đóng vai trò chủ chốt để đảm bảo các hoạt động trong cộng đồng theo phương pháp liên kết.

Cả hai danh sách các tình huống dưới sự uỷ thác cần được định lại và những hành động cần thiết trong các trường hợp đó đều có thể là rỗng. Do đó một Agent có thể luôn luôn uỷ thác một mục đích thậm chí nếu nó đã đạt được hoặc một Agent có thể không thực hiện hoạt động gì khi môi trường chứa nó thay đổi. Như trong minh hoạ dưới đây chỉ ra một sự thoả hiệp dựa trên mô hình hành vi cá thể, một Agent có thể từ bỏ việc thực hiện sự uỷ thác nếu nó biết rằng sự uỷ thác đã được thoả mãn hoặc không đạt được hoặc nếu không tồn tại sự thúc đẩy việc thực hiện sự uỷ thác.

+ Sự uỷ thác cộng đồng

Mặc dù những sự thoả hiệp đóng một vai trò quan trọng trong việc điều phối nhưng nó được xây dựng không có tính cộng đồng, tính xã hội. Sự thoả hiệp mô tả một Agent điều khiển sự uỷ thác của nó như thế nào, nhưng nó không đặc tả một Agent tác động đến các thành viên khác trong cộng đồng như thế nào khi nó thay đổi hoặc chỉnh sửa sự uỷ thác của nó. Với những mục đích mà chúng không liên quan các hoạt động khác thì sự thoả hiệp được coi là đầy đủ. Tuy nhiên với những mục đích phụ thuộc lẫn nhau cần phải thông báo sự thay đổi của mình đến tất cả các quá trình có liên quan khác, nếu cộng đồng Agent hoạt động theo phương thức liên kết. Một mặt việc cung cấp các thông tin cần thiết về sự thay đổi trong việc uỷ thác của Agent đến các thành viên trong cộng đồng một cách sớm nhất nếu có thể là rất quan trọng, một mặt các Agent không nên quảng bá thông tin về sự uỷ thác của chúng bất cứ khi nào chúng thay đổi vì điều này sẽ làm cho nguồn tài nguyên liên lạc sẽ bị quá tải và làm cho bên nhận dễ bị sao lãng một cách không cần thiết.

Một cách lý tưởng, những thành viên tham gia cần phải nhận thức được sự thoả hiệp nào điều khiển các mối tương tác của chúng. Sự nhận thức này là rất cần thiết nếu các Agent cực tiểu hoá những sự hợp tác không cần thiết, không chắc chắn và cực đại những lợi ích của việc liên kết hành động (joint action). Do đó, trở lại với ví dụ trên, nếu $Agent_2$ phải có nguồn tài nguyên d^1_j để hoàn thành $G^2_{p,2,2}$ thì nó sẽ yêu cầu sự hợp tác với $Agent_1$ để có được tài nguyên sẵn có. Tuy nhiên như vậy chưa đủ bởi vì $Agent_2$ còn muốn nguồn tài nguyên được sản xuất sử dụng một sự thoả hiệp cộng đồng thích hợp. $Agent_1$ có chấp nhận đề xuất của $Agent_2$ hay không tuỳ thuộc vào ý muốn cá nhân của nó và mối quan hệ giữa hai Agent. Nếu đề nghị của $Agent_2$ được $Agent_1$ chấp nhận hoặc nếu $Agent_2$ có thể buộc $Agent_1$

thì sự thoả hiệp sẽ được chấp thuận. Nếu đề nghị không được chấp thuận hai Agent sẽ cùng nhau thương lượng để quyết định đạt được sự chấp thuận. Cứ như vậy sự thoả hiệp cộng đồng cho mọi mục đích phụ thuộc lẫn nhau sẽ làm giảm tốc độ của các tiến trình khác, do đó khi thiết kế hệ thống phải đặt ra điều kiện khi hai Agent tương tác với nhau chúng phải sử dụng một sự thoả hiệp đặc biệt.

Vậy đặt ra một câu hỏi sự thoả hiệp cộng đồng cho một sự uỷ thác liên kết khác với sự thoả hiệp cộng đồng cho một sự uỷ thác cá thể như thế nào? Câu trả lời cho câu hỏi này dựa trên yêu cầu hỗ trợ lẫn nhau của các hành động liên kết. Mẫu tối thiểu của việc hỗ trợ tương hỗ là một nhóm các Agent hợp tác cung cấp cho các nhóm khác các thông tin chung như:

Trạng thái của sự uỷ thác đối với những mục tiêu chung

Trạng thái của sự uỷ thác đối với cả nhóm.

Nếu một Agent có biết về một trong hai loại thông tin trên thay đổi thì nó sẽ thể hiện một phần qua ngữ nghĩa trực quan của sự uỷ thác liên kết đến tất cả các thành viên khác trong nhóm. Cũng như nhiều hành động liên kết khác dựa trên sự tham gia của tất cả các thành viên, chỉ cần một sự thay đổi của sự uỷ thác của một Agent có thể làm ảnh hưởng nghiêm trọng đến hiệu năng của cả nhóm.

+ Sự uỷ thác trong các mối liên hệ mục đích - mục đích con

Nếu mục đích đang được thực hiện bởi một nhóm các Agent hợp tác trong mối quan hệ "VÀ" thì tất cả sự uỷ thác các mục đích con đều đóng vai trò quan trọng trong việc thực hiện thành công mục đích cha. Nếu một Agent bị từ bỏ thì các hành động của các Agent khác cũng sẽ bị ảnh hưởng. Do đó, các mục đích "AND" không thể đạt được bằng cách giải quyết bài toán hợp tác mà không có các ký hiệu của sự uỷ thác.

Nếu một mục đích được tạo ra bởi một số lượng các mục đích con hợp thành trong mối quan hệ hoặc (OR), và những Agent này hoạt động song song nhau, nếu một Agent gặp lỗi khi thực hiện sự uỷ thác của nó sẽ không ảnh hưởng nghiêm trọng đến kết quả của mục đích cha. Tuy nhiên nếu tất cả các Agent đều không thực hiện nhiệm vụ của chúng thì mục đích cha sẽ không đạt được, do đó phải đảm bảo ít nhất một Agent thực hiện sự uỷ thác thành công. Nếu các Agent điều phối các hoạt động của chúng gần giống nhau và chọn ra chỉ một mục đích con luân phiên nhau để thực hiện tại mỗi thời điểm (để tránh sự lặp lại không cần thiết) thì sự uỷ thác bị lỗi có thể ảnh hưởng nghiêm trọng tới toàn bộ tất cả các mức liên kết của hệ truyền thông.

+ Sự uỷ thác trong mối quan hệ phụ thuộc giữa các mục đích

Giả thiết rằng $Agent_1$ và $Agent_2$ lần lượt liên quan với các mục đích G^1_1 và G^2_1. Nếu có một sự phụ thuộc mạnh và hai chiều thì cả hai Agent đều phải thực hiện sự uỷ thác của chúng, nếu không sẽ không có Agent nào có thể đạt được mục tiêu của chúng. Nếu mối quan hệ là mạnh nhưng một chiều ($G^1_1 \square G^2_1$) thì nếu $Agent_1$ bị lỗi có nghĩa $Agent_2$ sẽ không thể thực hiện G^2_1 và sẽ không đạt được mục đích cha.

Với những sự phụ thuộc yếu, các Agent liên quan vẫn có thể tiếp tục hoạt động nhưng có thể đây là một phương pháp gần tối ưu.

6.2.2.5. Điều phối theo mô hình không gian 5D

Mô hình điều phối trong không gian 5D dựa trên sự điều khiển các tiến trình để thoả mãn các ràng buộc. Sự điều phối là một thành phần giữa (middle ware) và nó cũng có những ngôn ngữ riêng, mô hình riêng của nó. Chúng ta xét đến mô hình điều phối trong một hệ thống các Agent với các trạng thái luôn biến đổi, trong một cộng đồng Agent có thể có một điểm đơn (single point – macro model) hoặc một tập hợp các điểm (mỗi điểm đại diện cho một Agent). Trong không gian trạng thái đó, các tiến trình điều phối sẽ đưa ra các phương thức để luôn giữ cho không gian ấy luôn ở một quỹ đạo thích hợp. Tuy nhiên sự "thích hợp" ở đây dựa trên các tình huống, các loại điều phối khác nhau.

Trước khi mô tả mô hình coordination ta phải xét đến các mục đích quan trọng của một sự điều phối:

- Mục đích của sự điều phối theo mô hình 5D

Coi mô hình của sự điều phối bao gồm nhiều chiều, mỗi chiều được coi là một hướng, một khía cạnh.

+ Tính định lượng: Mỗi một khía cạnh của mô hình điều phối đều được thể hiện với một tỉ lệ nhất định có thể đo lường được tính đúng đắn của nó.

+ Kết nối trực tiếp để điều phối: Mỗi một hướng đều nên được kết nối trực tiếp với sự điều phối chất lượng, điều phối tiến trình.

+ Các quan hệ tương hỗ trực giao: Bất cứ sự phụ thuộc nào của một hướng dựa vào một hướng khác chỉ nên phụ thuộc vào cộng đồng Agent.

Những mục đích chính trên cho phép tạo ra các mô hình điều phối có thể thao tác được bằng các kỹ thuật toán học như tối ưu tuyến tính.

- Mô hình điều phối trong không gian 5D

Tiếp đến ta xét đến các khía cạnh trong điều phối là:

+ *Tính tự chủ:* Đó chính là sự tự do hoạt động của một Agent, không phụ thuộc vào các Agent khác ngay cả khi có sự hợp tác giữa chúng trên một vấn đề nào đó. Độ tự chủ được đo bằng mức độ mà các Agent bị điều khiển bởi các Agent khác. Tuy nhiên, mục đích của việc đưa khía cạnh này vào trong điều phối không phải là mức độ của độ tự chủ hay tính kết nối giữa các Agent mà nó lên sức mạnh và sự lựa chọn trong cộng đồng Agent.

+ *Kiến thức về các Agent khác:* Mỗi Agent trong cộng đồng đều xây dựng các mô hình về các Agent khác và sử dụng các mô hình đó để dự đoán trước các hành động của các Agent đó. Những mô hình này càng chính xác bao nhiêu thì càng có khả năng điều khiển được các yêu cầu của mình đối với cộng đồng, đưa ra các chiến lược phù hợp để tránh xung đột.

+ *Tính kết nối:* Tính kết nối của một Agent có thể được định nghĩa là số lượng các Agent khác mà nó có thể tương tác trực tiếp. Tính kết nối của các Agent càng cao thì chúng càng

dễ có được những thông tin, những sự giúp đỡ mà chúng cần, và có nhiều cơ hội hơn để điều phối có hiện quả, giải quyết các vấn đề gần đạt đến trạng thái tối ưu.

+ *Sự thoả thuận mục đích và xung đột:* Mối quan hệ giữa các mục đích của các Agent khác nhau rất quan trọng bởi vì những quan hệ này là nền tảng cho sự tương tác giữa các Agent (các Agent thường tương tác với nhau để đạt được mục đích của chúng). Mỗi Agent đều tìm kiếm để đạt được các tiện ích, các tài nguyên một cách tối đa để đạt được mục đích của nó đồng thời đó cũng là nguyên nhân của sự xung đột giữa các Agent.

+ *Kiến thức về bài toán, vấn đề cần giải quyết:* Kiến thức đặc biệt các Agent có, các tiến trình chúng có thể thực hiện, dữ liệu chúng truy cập, tất cả ảnh hưởng đến việc ấn định các nhiệm vụ của các Agent. Kiến thức về vấn đề bao gồm cả sự hiểu biết về những ràng buộc tài nguyên khi thiết kế để hoàn thành nhiệm vụ đó. Nếu có nhiều Agent có thể cùng thực hiện một nhiệm vụ, kiến thức bao phủ về vấn đề sẽ là tiêu chí để chọn một hoặc một số Agent nào có thể thực hiện chúng.

Từ đó, người ta xây dựng các khung làm việc 5 chiều (5D) liên quan đến Agent, cho phép ánh xạ một mô hình điều phối đến các mức của khung làm việc 5D đó. Hầu hết các cấu trúc tổ chức đều bao gồm tự chủ và kiến thức. Sự "tự chủ" trong mô hình liên quan đến sức mạnh của cấu trúc đó, tính kết nối và kiến thức liên quan đến số lượng các Agent "láng giềng" và các thông tin về chúng.

- Kiến trúc cơ bản của mô hình điều phối

Mô hình này là tập hợp các đối tượng tham gia vào nhiều tiến trình mà bị giới hạn bởi các ràng buộc nào đó.

+ Các thành phần của mô hình

Một mô hình của sự điều phối bao gồm các hệ thống con sau:

Một mô hình Agent: thể hiện các quá trình xử lý bên trong của các Agent, các quyết định, sự hiểu biết về các thực thể bên ngoài như môi trường, các Agent khác, các nhiệm vụ và toàn bộ cộng đồng. Sự đồng nhất của các Agent không phải là mục đích của việc mô hình hoá quá trình điều phối, sự hiểu biết, cách xử lý, các ràng buộc của mỗi Agent có thể khác nhau nhưng tất cả đều ảnh hưởng đến quá trình điều phối.

Một mô hình hệ thống: Thể hiện sự tương tác và thương lượng giữa các Agent (như việc thương lượng dùng mô thức nào để tương tác: broadcast hay point to point...)

Một mô hình môi trường: Thể hiện sự ảnh hưởng của môi trường và bất cứ các giới hạn nào bị ảnh hưởng bởi nó như lỗi kênh truyền, dữ liệu tắc nghẽn...

Một mô hình dữ liệu: Thể hiện việc phân bổ, truy cập, tổ chức, sự hoàn thiện và chính xác của bất cứ dữ liệu nào có thể được yêu cầu bởi các Agent. Dữ liệu là các thông tin được tồn tại độc lập với các Agent thậm chí nó được truy cập bởi các Agent.

Một mô hình nhiệm vụ: Thể hiện cấu trúc, độ phức tạp của nhiệm vụ và bất cứ một sự tương tác và ràng buộc nào giữa các nhiệm vụ khác nhau hoặc giữa các phần riêng biệt của một nhiệm vụ.

+ Các tiến trình

Một tiến trình trong một hệ thống đa Agent được định nghĩa như một chuỗi các hoạt động theo thứ tự thời gian thao tác trên các Agent (sự hiểu biết của chúng, trạng thái...), dữ liệu, các nhiệm vụ hoặc trên môi trường. (Thao tác trên các thực thể bao gồm mô hình hệ thống phụ tương ứng với các tổ chức tự thiết kế). Các tiến trình có thể được thực hiện bởi một hoặc nhiều Agent hoặc có thể là những sự thay đổi "không mang tính thông minh" của môi trường (như thời tiết). Ta có thể chia các tiến trình trong hệ thống đa Agent thành các loại chính sau:

Các tiến trình hợp tác: Được định nghĩa là các tiến trình được thực hiện bởi các Agent để hợp tác thực hiện các nhiệm vụ. Tiến trình hợp tác được thực hiện để tăng tính bền vững cho cộng đồng. Tính bền vững cộng đồng chính là tổng số các tiện ích cùng tham gia vào thực hiện một tiến trình.

Các tiến trình cạnh tranh: Được định nghĩa là một chuỗi các hành động để đạt được lợi ích cho một Agent.

Các tiến trình trung lập: Không liên quan đến cả hợp tác lẫn cạnh tranh.

+ Các ràng buộc

Các ràng buộc là những sự giới hạn, sự hạn chế trên Agent và các hoạt động hệ thống. Những ràng buộc này có thể được xuất phát từ những yêu cầu, từ những luật ngăn cấm hoặc giới hạn tài nguyên. Ta có thể chia các ràng buộc thành các dạng chính như sau:

Các ràng buộc đồng nhất: chỉ một số Agent được có thể được chọn cho sự hợp tác hoặc cạnh tranh dựa trên sự đồng nhất của chúng.

Các ràng buộc về khả năng: ở đó các Agent được chọn dựa trên khả năng của chúng hoặc kiến thức đặc biệt.

Các ràng buộc lớp: ở đó các Agent được chọn dựa trên mối quan hệ member trong một class nào đó.

Các ràng buộc dung lượng: là các giới hạn của kiến trúc hệ thống hoặc các giới hạn vật lý. Các ràng buộc luồng như hạn chế băng thông đường truyền khi gửi thông điệp cũng là ràng buộc dung lượng.

Các ràng buộc thời gian: là các ràng buộc mà các nhiệm vụ phải đạt được trong mối quan hệ tạm thời với các nhiệm vụ khác.

Mục đích của việc đưa ra các ràng buộc chính là để ánh xạ các hành vi của hệ thống sang khung làm việc 5D.

+ Sự kiểm tra cho cơ cấu Agent - Tiến trình- Ràng buộc

Nếu các tiêu chí để cho sự điều phối là tối ưu nhất có thể được diễn tả bằng các tiến trình, các ràng buộc, các biến trạng thái, nó sẽ có khả năng đi tới một học thuyết, một lý thuyết dễ hiểu về qui cách trạng thái của cộng đồng dưới những điều kiện khác nhau.

Ta coi mỗi ràng buộc cho mỗi quá trình bị ràng buộc bởi nó là một siêu bề mặt (hyper-surface) trong không gian 5D được định nghĩa bởi 5 chiều như đã nói trên. Khi tất cả các ràng buộc được thoả mãn, các thao tác hệ thống trong vùng khả thi được giới hạn bởi những siêu bề mặt này. Gọi C= $\{C_i\}$ là tập các ràng buộc,

P = $\{P_j\}$ là tập các tiến trình, do đó tập các siêu bề mặt H sẽ được định nghĩa như sau: H= $\{H_{ij} \mid$ Ci các ràng buộc Pi ^ $H_{ij} = F(C_i, P_j)\}$

Trong đó $F(C_i, P_j)$ là hàm chức năng của ràng buộc trong không gian 5D. Giả sử cho I tiến trình và J các ràng buộc có thể sẽ tạo ra I.J các siêu bề mặt. Tuy nhiên trong thực tế rất ít trường hợp tất cả các ràng buộc đều được áp cho tất cả các tiến trình. Như vậy công thức này đưa ra một cách ánh xạ của tiêu chí một sự điều phối tối ưu sang một không gian 5D. Ta có thể ánh xạ ngược lại sang mô hình ràng buộc và tiến trình và sự phân bổ tài nguyên cho các tiến trình có thể được điều chỉnh...sang sự điều phối tối ưu.

- Phương pháp điều phối trong không gian 5D

Phương pháp của mô hình điều phối dựa trên sự phân tích về các tiến trình, các ràng buộc, sự xây dựng và thực hiện của hệ thống. Các bước chính:

+ Mô tả các tiến trình và các ràng buộc về sự thực thi trong hệ thống đa Agent. Đây chính là bước yêu cầu định danh của các hệ thống phụ.

+ Định danh các biến trạng thái và mô tả mối quan hệ giữa các biến trạng thái sinh ra bởi các tiến trình và các ràng buộc. Bước này liên quan đến việc làm mịn các tương tác dựa trên kiến trúc của hệ thống.

+ Mô tả tiêu chí cho sự điều phối tối ưu theo các tiến trình, các ràng buộc và các biến trạng thái.

+ Quyết định các phép biến đổi tối ưu của hệ thống dựa vào sự phối hợp tối ưu giữa các biến trạng thái.

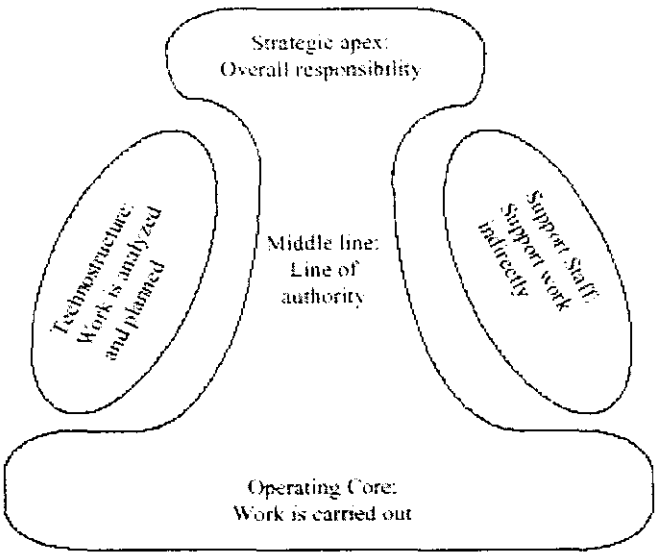
+ Biến đổi ngược các phép kết hợp tối ưu của các tiến trình và các ràng buộc để quyết định các cơ chế điều phối phù hợp cho các tiến trình/ các ràng buộc.

6.2.2.6. Điều phối theo mô hình Mintzberg

Mô hình của Mintzberg dựa trên cấu trúc của các tổ chức, các cơ chế điều phối. Mô hình của Mintzberg bao gồm 5 thành phần:

- Operating core (nhân điều hành) là thành phần cơ bản, nền tảng của mô hình, là nơi thực hiện trực tiếp các công việc.
- Strategic apex (điểm chiến lược) là thành phần đỉnh (top) của mô hình, điều hành, có trách nhiệm đưa ra các quyết định chiến lược và hướng dẫn phương hướng của tổ chức điều phối

- Middle line (đường nằm giữa) là một dây chuyền, một chuỗi các sự điều hành việc thực thi các quyết định bằng việc giám sát các mức dưới quyền và thông báo lại cho những giám sát viên của chúng.
- Technostructure (Kỹ thuật cấu trúc): Phân tích và tổ chức để thực hiện các công việc.
- Support staff: bao gồm tất cả các yếu tố hỗ trợ công việc một cách gián tiếp.



Hình 6-6: Điều phối theo mô hình Mintzberg

Mô hình trên được giải thích bởi 5 cơ chế sau:

+ Điều chỉnh tương hỗ
+ Giám sát trực tiếp
+ Chuẩn hoá công việc (Standardization of work)
+ Chuẩn hoá các đầu ra (Standardization of outputs)
+ Chuẩn hoá các kỹ năng (Standardization of skillls)

6.2.2.7. Các kỹ thuật điều phối

- Cấu trúc có tổ chức (Organizational Structure) – long term

Một phương thức điều phối giữa các Agent là thông qua cấu trúc có tổ chức của mạng Agent. Trong ngữ cảnh của các hệ thống trí tuệ nhân tạo phân tán, một cấu trúc có tổ chức được xem như một mẫu thông tin và điều khiển mối quan hệ giữa các cá thể. Những cấu trúc điều khiển này với các dạng như phân lớp, giao lớp, phẳng (flat) tương ứng với mối quan hệ phân quyền giữa các Agent và phù hợp với dạng tương tác cộng đồng có thể xảy ra. Ví dụ như khi xây dựng một hệ thống các Agent để chẩn đoán sự cố mạng điện, nhà thiết kế hệ thống phải đặc tả mô hình chức năng (Agent 1 làm việc với các sự cố cao thế trong khi Agent 2 làm việc ở mức hạ thế) hoặc đặc tả mô hình không gian (Agent1 chịu trách nhiệm với các sự cố ở vùng 1, Agent 2 chịu trách nhiệm với các sự cố điện ở vùng 2). Đối với các Agent phân tán theo không gian, giả sử các vùng đó chồng lên nhau, khi đó sẽ gây ra sự dư

thừa hành động của các Agent. Do đó, chúng ta có thể cung cấp toàn bộ sự điều phối bằng cách đặc tả hành động nào mà một Agent sẽ phải đảm nhiệm và có thể tránh như thế nào các công việc dư thừa được thực hiện bởi các Agent khác.

Việc đặc tả mối quan hệ bằng cách cung cấp các cấu trúc có tổ chức là thông tin dài hạn (long-term) về các Agent và cộng đồng xung quanh nó. Nó hỗ trợ tiến trình điều phối bằng sự đặc tả các hành động mà mỗi Agent phải đảm nhiệm thông qua phương tiện phân chia không gian tìm kiếm. Mức độ chi tiết của các kiểu mẫu tổ chức Agent được gọi là luật cộng đồng. Những Agent trong cộng đồng ấy phải tuân theo tập luật này. Mỗi nhà lập trình được uỷ thác tuân theo tập luật này khi xây dựng Agent riêng của anh ta. Quá trình thiết kế khá đơn giản vì giả định rằng tất cả các Agent khác sẽ tuân theo luật đã định ra đó.

Khi một Agent đóng một vai trò đặc biệt trong tổ chức của nó, trong thực tế là tạo ra một mức độ uỷ thác cao về loại hành động mà nó sẽ phải theo đuổi. Lấy ví dụ với trường hợp quản lý điện, nếu Agent1 đảm nhiệm vai trò phát hiện sự cố điện cao áp, các Agent khác sẽ kỳ vọng nó đảm nhiệm công việc trong vùng này. Chúng sẽ đưa ra các quyết định sau đó để giải quyết bài toán của chúng dựa trên sự thừa nhận rằng Agent 1 sẽ thực sự phải đối đầu với mọi sự cố liên quan đến mạng cao áp. Mặc dù đây được coi là một cấu trúc dài hạn, nhưng nó được biểu diễn dưới các tổ chức khác nhau để thích hợp với các tình huống bài toán và yêu cầu hiệu năng khác nhau. Do đó với một tình huống phát sinh, cộng đồng có thể định lại tổ chức của nó theo chu kỳ để quyết định xem cấu trúc đó vẫn phù hợp hay phải sắp xếp lại để đạt được lợi ích tối đa. Lại xét ví dụ quản lý mạng điện, tổ chức có thể quyết định tốt nhất là thay thế Agent chịu trách nhiệm phát hiện, dự đoán sự cố điện cao áp bằng một vài các Agent phân tán theo không gian để làm giảm tải và độ trông cậy vào các Agent.

- Sự siêu trao đổi thông tin (Meta-Level Information Exchange) – medium term

Sự siêu trao đổi thông tin liên quan đến việc các Agent gửi cho nhau các thông tin điều khiển về những thuộc tính, sự quan tâm, định hướng hiện thời của chúng. Một nút truyền đi các mục đích và các giả thuyết đến các nút khác, là những nút nó cho rằng quan tâm đến các thông tin nó gửi đi. Nó cũng nhận được các thông tin về mục đích, về giả thuyết từ các Agent khác. Các hình thái giải quyết một bài toán cục bộ của một Agent liên quan đến các thông tin mà nó nhận được từ các Agent khác. Do đó, khả năng giải quyết một bài toán của một nút được cân bằng bởi sự nhận thức của nó về cách giải quyết thích hợp với các hoạt động của các nút khác nó cho rằng quan trọng. Siêu trao đổi thông tin là một quá trình trung hạng của sự nhận thức về sự uỷ thác của một Agent- ngắn hơn cấu trúc có tổ chức nhưng dài hơn lập kế hoạch đa Agent. Nó cải thiện mức độ chính xác của sự điều phối vì với những thông tin không chính xác đôi khi mang lại hậu quả xấu hơn là việc không có thông tin. Do đó, nếu mỗi Agent thông báo cho nhau về sự thay đổi dù là rất nhỏ trong các sự uỷ thác của chúng có thể sẽ gây ra phản ứng dây chuyền cho toàn hệ thống. Vì vậy, các Agent tuân theo một sự thoả hiệp trong đó chúng chỉ thông báo khi sự thay đổi thực sự có ý nghĩa.

Durfee đã phát triển một cơ chế siêu trao đổi thông tin gọi là kế hoạch tổng thể từng phần (Partial Global Planning-PGP) trong đó các Agent xây dựng và chia sẻ các kế hoạch. Các PGP được trao đổi bởi các Agent như là một phương tiện xây dựng sự thể hiện của các hành động quen thuộc của chúng- chúng chỉ ra những mục đích nào sẽ phải đạt được, theo thứ tự nào, ảnh hưởng và thời gian để đạt được mục đích này. Sự điều phối sẽ dựa trên việc kết hợp các PGP riêng lẻ thành các PGP lớn hơn cung cấp cách nhìn hoàn thiện hơn về các hoạt động của nhóm.

Các Agent sinh ra các PGPs và các PGPs này sẽ định ra các định hướng cho mỗi Agent.

- Lập kế hoạch đa Agent (Multi-Agent planning) - short term

Để điều phối trong hệ đa Agent, các Agent thường lập kế hoạch để mô tả các hành động và tương tác của chúng trong tương lai để đạt được một mục tiêu đặc biệt nào đó. Cụ thể là trước khi thực hiện một công việc trong thực tế, các Agent quyết định không gian tìm kiếm, con đường đi của chúng. Lập kế hoạch đa Agent để tránh việc thực hiện các hành động trái ngược hoặc đụng độ nhau đặc biệt là việc tranh giành nguồn tài nguyên khan hiếm. Ví dụ như đối với việc điều khiển các phương tiện đường không, mỗi phương tiện (Agent) gửi đi thông tin của điều phối viên về các hành động sắp tới. Điều phối viên sẽ xây dựng một kế hoạch để mô tả các hoạt động của từng phương tiện để tránh đụng độ.

Lập kế hoạch đa Agent khác với cấu trúc có tổ chức, siêu trao đổi thông tin ở mức độ chi tiết việc đặc tả mọi hoạt động của tất cả các Agent. Với phương pháp này, các Agent biết chính xác những hành động chúng sẽ thực hiện, những tương tác sẽ xuất hiện. Các kế hoạch chỉ được thực hiện trong một khoảng thời gian ngắn vì các sự kiện trong môi trường là động và không biết trước. Khi xây dựng kế hoạch phải thống kê tất cả các trường hợp có khả năng xảy ra với các Agent nên yêu cầu tài nguyên tính toán và truyền thông nhiều hơn hai kỹ thuật trên.

Có hai cách thực hiện lập kế hoạch đa Agent là *tập trung* và *phân tán*. Với việc lập kế hoạch đa Agent tập trung, mỗi Agent trong hệ thống gửi cho điều phối viên thông tin về các hành động mong muốn của mình. Điều phối viên sẽ xây dựng một kế hoạch mô tả toàn bộ hoạt động của tất cả các Agent trong đó có cả các hành động của một số nút khác nên làm để tránh đụng độ. Với việc lập kế hoạch đa Agent phân tán, kế hoạch được phát triển bởi một số Agent và gửi đến cho điều phối viên để điều chỉnh. Điều này có nghĩa là không một Agent nào có thể biết được toàn bộ các hoạt động của cộng đồng mình. Tuy nhiên, tình huống có thể thay đổi rất nhanh chóng giữa lúc lập kế hoạch và thực thi, nếu kế hoạch đã được thực hiện thì lợi ích của việc lập kế hoạch trở nên không đáng kể thậm chí còn có tác động tiêu cực. Trong trường hợp này, phải thêm vào pha lập lại kế hoạch để tạo ra một kế hoạch có lợi. Do vậy, cần phải có một sự thoả hiệp khi nào thì quyết định lập lại kế hoạch, khi nào tái sử dụng lại kế hoạch đã được lập và khi nào một kế hoạch hoàn toàn mới được xuất hiện.

6.2.2.8. Thiết kế ngôn ngữ điều phối cho hệ đa Agent

- Các giả định về Agent:

+ Những Agent tự chủ có kế hoạch riêng lẻ, cá thể.

+ Các Agent có thể nhận thức về các Agent khác và có thể cho các thông điệp đi qua chúng.

+ Các Agent không thể đoán trước được hành vi của các Agent khác, nhưng có thể dự tính trước được các Agent sẽ làm gì.

+ Các kế hoạch của các Agent có thể bị thay đổi trong quá trình thực thi.

- Sự giao tiếp trong điều phối giữa các Agent

Sử dụng ngôn ngữ KQML để định dạng các thông điệp, còn các dạng hoạt động liên lạc, truyền thông được tạo ra bởi người phát triển chúng. Ta hãy xét đến cách các Agent trao đổi với nhau qua việc gửi thông điệp trong môi trường chúng chạy như thế nào.

Cấu trúc của mỗi Agent có dạng: bao gồm một tên (name) và một thông dịch viên (interpreter) để điều khiển sự trao đổi của Agent đó. Mỗi Agent được điều khiển bởi một bộ phận quản lý sự trao đổi (conversation manager).

Mỗi bộ phận quản lý sự trao đổi bao gồm:

Một danh sách các Agent mà nó quản lý

Một hàm điều khiển để lựa chọn các Agent để thực thi và điều khiển tài nguyên

Môi trường để "chạy" Agent với mục đích: quản lý việc trao đổi thông điệp và đưa ra lịch biểu thực thi của Agent. Môi trường có thể được phân chia để các tiến trình chạy trên các máy tính khác nhau (môi trường phân tán). Trong môi trường đó có dịch vụ cho phép chuyển các thông điệp đến các máy tính khác nhau

Các Agent có thể chuyển thông điệp, thay đổi trạng thái và thực hiện các hoạt động bộ phận trong quá trình trao đổi. Luật trao đổi dựa trên giá trị của các biến đặc biệt gọi là "biến trạng thái". Những sự trao đổi này được thể hiện thành các "lớp trao đổi" (conversation classes) hoạt động để mô tả các kế hoạch của Agent để đạt được một mục đích. Các lớp trao đổi được liên kết với các Agent riêng biệt để quyết định xem mỗi Agent có thể tương tác theo kiểu nào.

Quá trình trao đổi:

Đầu tiên mỗi Agent tạo ra một thể hiện của sự trao đổi.

Khi một thể hiện của sự trao đổi được thực hiện thì thông điệp sẽ được gửi từ một Agent và sẽ nhận được thông điệp từ các Agent khác. Mỗi một thông điệp có một vùng để định danh các Agent có liên quan đến thông điệp đó.

Mỗi một Agent có thể đưa ra các chính sách của mình để lựa chọn thông điệp nào để tiếp tục công việc. Một Agent có thể cùng một lúc thực hiện vài cuộc trao đổi. Một Agent đưa ra quyết định lựa chọn cuộc trao đổi nào tiếp theo để điều khiển dựa trên sự ưu tiên của chúng (phụ thuộc vào tập luật)

Các Agent dựa vào các chính sách của mình về sự trao đổi để đưa ra quyết định xem Agent đó có nên tiếp tục bắt đầu một cuộc trao đổi mới hay không.

Những điểm cần chú ý trong khi thực hiện các cuộc trao đổi:

Các cuộc trao đổi có thể hoạt động như các luồng

Mỗi cuộc trao đổi có thể bị dừng (hoãn) trong khi Agent thực hiện một cuộc trao đổi khác.

Có nhiều cuộc trao đổi có thể được khôi phục lại khi những cuộc trao đổi khác kết thúc.

6.2.2.9. Kết luận

Trên đây là toàn bộ những lý thuyết liên quan đến điều phối trong hệ đa Agent. Sự điều phối thành công là chìa khoá chính cho việc thiết kế hệ thống đa Agent. Trong tương lai, người ta hướng đến những công cụ trợ giúp thiết kế việc điều phối cũng như có một độ đo định lượng để đo thế nào là một sự điều phối tốt.

6.3. Quản lý tài nguyên trong tính toán lưới

6.3.1. Các thách thức trong quản lý tài nguyên lưới

Quản lý tài nguyên lưới bao gồm việc khám phá, định vị, thương lượng, bao quát và quản lý yêu cầu của người sử dụng nhằm cung cấp chất lượng dịch vụ tốt nhất. Trong kiến trúc hướng dịch vụ, tài nguyên lưới có thể bao gồm cả nguồn tài nguyên truyền thống (Những dịch vụ tính toán, băng thông mạng hoặc hệ thống lưu trữ ...) và tài nguyên ảo (Cơ sở dữ liệu, dịch vụ truyền dữ liệu hay dịch vụ mô phỏng ...). Không làm mất tính tổng quát, thuật ngữ quản lý tài nguyên (Resource Management) được sử dụng để mô tả tất cả các khía cạnh của quá trình định vị, thoả thuận, sử dụng và theo dõi trạng thái của tài nguyên.

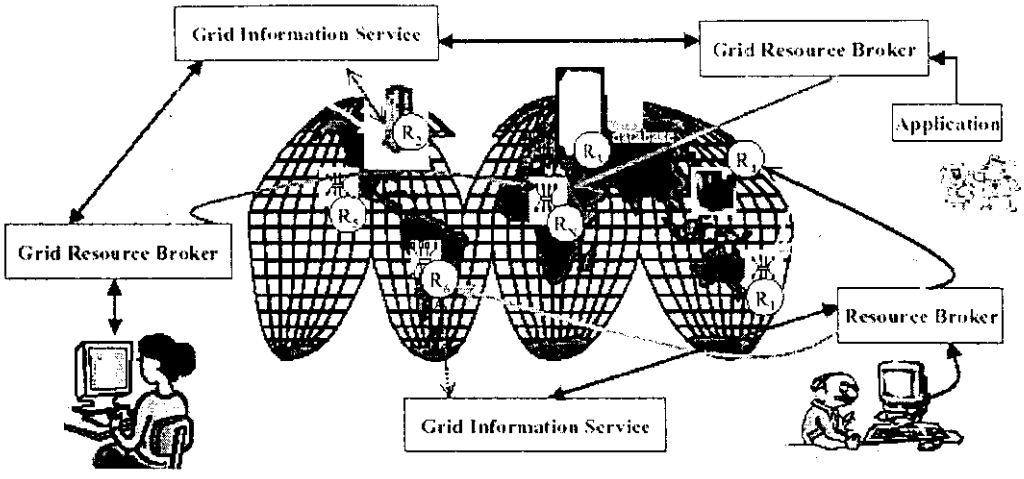
Trong hệ thống tính toán truyền thống, quản lý tài nguyên được xem là một khâu của bộ lập lịch, hay của bộ máy quản lý luồng công việc hoặc của hệ điều hành. Hệ thống quản lý tài nguyên này được thiết kế và hoạt động với giả thiết rằng chúng nắm quyền điều khiển toàn bộ tài nguyên. Nhờ đó, chúng có thể thực hiện đầy đủ những cơ chế và các chính sách cần thiết để sử dụng hiệu quả tài nguyên. Nhưng trong lưới thì giả thiết này không đúng, vì chúng không còn giữ quyền điều khiển tuyệt đối, ta phải phát triển các phương thức để quản lý trên các vùng khác nhau với tập nguồn tài nguyên không đồng nhất.

Một đặc điểm nổi bật của tài nguyên lưới là sự không đồng nhất ngay cả ở trong cùng lớp hay cùng kiểu tài nguyên. Vì vậy, việc quản lý tài nguyên lưới tập trung vào giải quyết vấn đề không đồng nhất của tài nguyên. Hơn nữa, tổ chức khác nhau có các chính sách quản lý tài nguyên khác nhau. Mặt khác, mục đích của người dùng tài nguyên và các nhà cung cấp tài nguyên có thể khác nhau hoặc thậm chí có thể mâu thuẫn nhau. Hầu hết các ứng dụng yêu cầu sử dụng đồng thời nhiều tài nguyên ở nhiều nơi khác nhau. Ví dụ, một chương trình mô phỏng dự báo thời tiết sẽ yêu cầu ít nhất là tập tài nguyên lưu trữ cho tập dữ liệu đầu vào, đầu ra, nguồn tài nguyên mạng để truyền dữ liệu, nguồn tài nguyên tính toán để chạy sự mô

phỏng và các nguồn tài nguyên này phải phối kết hợp để hoàn thành công việc. Hiện nay, trong kiến trúc hướng dịch vụ lưới, ngày càng tăng các ứng dụng thương mại. Vì vậy, một trong các mục đích của tính toán lưới là cung cấp chất lượng dịch vụ tốt nhất. Một thách thức nữa đối với quản lý tài nguyên là vấn đề bảo mật tài nguyên. Vì hệ thống lưới là phân tán cả về địa lý và tổ chức. Các tổ chức khác nhau thì có các chính sách về bảo mật khác nhau và phải được điều hòa để cho phép sử dụng đồng thời tài nguyên qua các miền khác nhau. Một lưới có đa dạng nguồn tài nguyên, mỗi tài nguyên lại yêu cầu các mức bảo mật khác nhau. Kiểu tài nguyên phổ biến trong lưới là là tài nguyên tính toán và tài nguyên lưu trữ. Đối với tài nguyên tính toán, người quản trị có thể điều khiển tốc độ CPU, dung lượng đĩa trống, tải, băng thông mạng, ...Còn tài nguyên lưu trữ thì đơn giản là quyền gắn đĩa tới một nút tính toán nào đó...Hệ thống bảo mật phải cho phép truyền thông tin nhận dạng để sử dụng trong thanh toán ...

6.3.2. Định vị tài nguyên lưới

Khi có yêu cầu của người dùng, bộ quản lý tài nguyên sẽ tìm tài nguyên từ dịch vụ chỉ mục (Index Service) và sau đó định vị tài nguyên đến nút lưới phù hợp. Thường thì sự định vị này có thể chia ra hai giai đoạn: Đầu tiên tài nguyên này sẽ được định vị đến một nút cụ thể nào đó trong lưới và tại nút này thì tài nguyên sẽ được lập lịch sử dụng. Ở đây, ta chỉ quan tâm đến giai đoạn đầu tiên đó là định rõ nguồn tài nguyên cần thiết. Khi một ứng dụng đang chạy, bộ quản lý tài nguyên cần theo dõi trạng thái tài nguyên và thông báo trở lại cho bộ lập lịch và hệ thống kế toán. Khi có 2 yêu cầu được đệ trình đến lưới cùng lúc thì cả 2 sẽ cùng được xử lý. Vấn đề xung đột này sẽ được giải quyết trong hệ thống định vị tài nguyên bằng các quy luật thị trường. Tức là người bán có thể bán vượt số lượng dự trữ. Nhưng chất lượng dịch vụ sẽ bị giảm và như vậy sẽ làm mất uy tín đối với khách hàng. Do đó, người cung cấp sẽ không làm như vậy nữa trong những lần phục vụ sau. Để tuân theo luật thị trường, ta sử dụng một quy ước. Quy ước này hoạt động như một hàng đợi: Khi một ứng dụng yêu cầu sử dụng tài nguyên mà hiện tại tài nguyên đó đang phục vụ cho một ứng dụng khác thì nó sẽ được xếp vào hàng đợi cho đến khi tài nguyên đó được sử dụng xong và sẵn sàng phục vụ. Như đã đề cập ở trên, môi trường lưới phân tán về địa lý và tài nguyên lưới là không đồng nhất. Do đó, để định vị đúng tài nguyên, ta cần phải thiết kế một hệ thống quản lý tài nguyên phù hợp. Với cách tiếp cận truyền thống là sử dụng chính sách tập trung để theo dõi và quản lý tài nguyên không thể sử dụng trong môi trường rộng lớn như lưới. Vì vậy, ta phải chuyển sang hướng tiếp cận đa tầng và tổ chức tài nguyên phi tập trung. Bên trong cách tiếp cận này, có thể tồn tại nhiều mô hình kinh tế khác nhau để quản lý và điều tiết sự cung cầu của nguồn tài nguyên. Ở đây, ta dùng một bộ môi giới tài nguyên đóng vai trò trung gian giữa nhà cung cấp tài nguyên và người sử dụng tài nguyên.

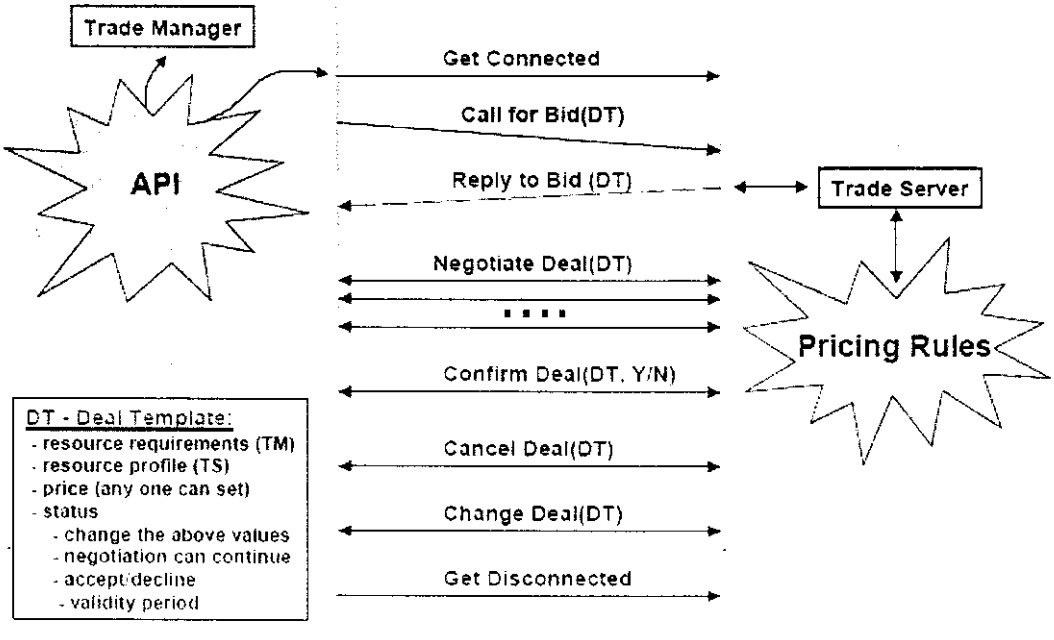


Hình 6-7: Tổng quan về quản lý tài nguyên lưới

Khách hàng sẽ tương tác trực tiếp với bộ môi giới tài nguyên lưới. Sau khi nhận được yêu cầu, bộ môi giới sẽ tìm kiếm tài nguyên qua dịch vụ thông tin lưới, thương lượng với các nhà cung cấp, lập lịch và theo dõi trạng thái của nguồn tài nguyên đó.

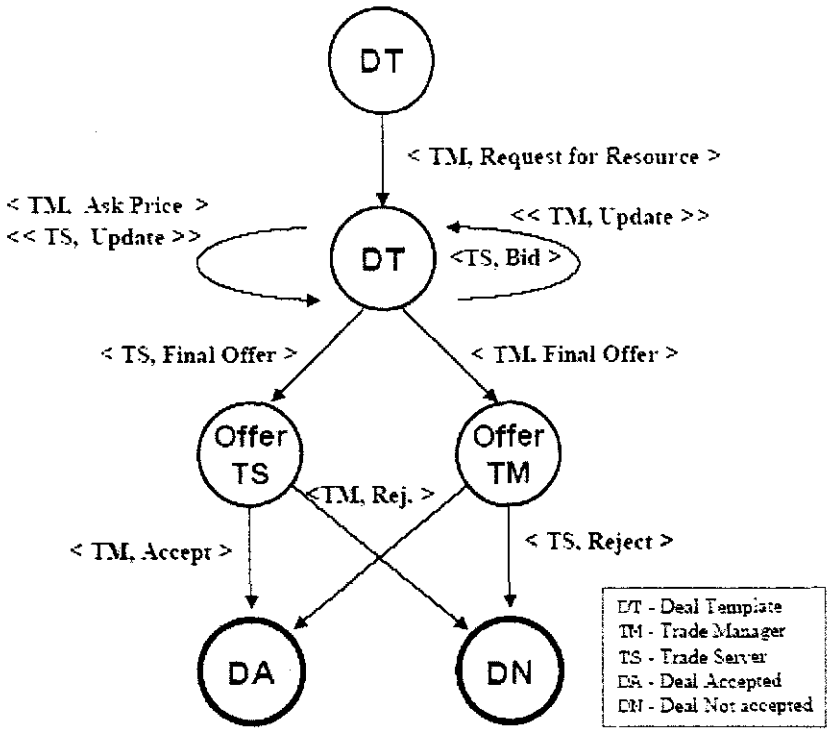
6.3.3. Vấn đề thương lượng tài nguyên lưới

Quá trình thương lượng tài nguyên lưới dựa trên các giao thức hay các luật trong kinh doanh để chuyển đổi các lệnh buôn bán giữa người sử dụng tài nguyên và các nhà cung cấp tài nguyên. Hình sau minh hoạ các giao thức thương lượng mà cả hai phía người mua và người bán cần trong quá trình mặc cả.



Hình 6-8: Mô hình thương lượng tài nguyên lưới

Đầu tiên bên phía khách hàng kết nối với nhà cung cấp, Sau khi nhận được giá tài nguyên, Cả hai bên bán và mua sẽ tiến hành thương lượng. Khi thương lượng thành công, bên phía khách hàng sẽ yêu cầu ngừng kết nối và sử dụng tài nguyên đó. Bộ máy hữu hạn trạng thái biểu diễn các giao thức trong quá trình mặc cả có thể được minh hoạ như sau:



Hình 6-9: Bộ máy hữu hạn trạng thái biểu diễn các giao thức thương lượng

Trong mô hình này, Bên môi giới (TM) liên lạc với chủ sở hữu tài nguyên (TS) để hỏi giá cả. TM sẽ xác định rõ yêu cầu tài nguyên từ mẫu thoả thuận (DT), mẫu này được thiết kế thành các trường đơn giản với ngôn ngữ đặc tả riêng. Nội dung của DT bao gồm: Thời điểm bắt đầu, khoảng thời gian sử dụng, dung lượng yêu cầu...TM sẽ căn cứ vào DT, cập nhật nội dung và gửi trở lại cho TS. Cuộc thương lượng giữa TM và TS tiếp tục cho đến khi một trong hai bên thoả mãn yêu cầu của mình. Đồng thời, một bộ phận khác cũng quyết định chấp nhận (DA) hay huỷ bỏ thoả thuận (DN) này. Nếu chấp thuận, thì cả hai sẽ tuân theo mẫu thoả thuận đã đưa ra. Mô hình đấu giá cũng là một trong nhiều kiểu thương lượng giữa một nhà cung cấp dịch vụ và nhiều khách hàng để đưa ra một giá cao nhất cho nguồn tài nguyên đó. Người điều khiển sàn đấu giá sẽ đưa ra các luật đấu giá. Cuộc đấu giá sử dụng luật thị trường để thương lượng giá cho dịch vụ. Trong thế giới thực, đấu giá được sử dụng phổ biến đặc biệt là trong kinh doanh hang hoá. Thông thường thì trên sàn đấu giá gồm các tác nhân: Chủ sở hữu tài nguyên, người điều khiển, khách hàng. Trong môi trường lưới, nhà cung cấp có thể sử dụng các giao thức đấu giá để quyết định giá cả của dịch vụ. Các bước diễn ra trong quá trình đấu giá bao gồm:

Các nhà cung cấp quảng bá dịch vụ của họ và ngã giá khởi điểm.

Môi giới đại diện cho người sử dụng sẽ đưa ra giá của mình.

Bước hai tiếp tục cho đến khi không có giá cao hơn được đưa ra.

Nhà cung cấp sẽ trao dịch vụ của mình cho khách hàng ngã giá cao nhất. Và kết thúc cuộc đấu giá.

Tóm lại, vấn đề quản lý tài nguyên trong môi trường lưới còn rất nhiều thách thức. Vì đây là môi trường phân tán về địa lý, nguồn tài nguyên không đồng nhất, và tồn tại nhiều chính sách quản lý của các tổ chức khác nhau. Để giải quyết các khó khăn này, ta sẽ sử công nghệ Agent và tìm hiểu những tiềm năng mà công nghệ này mang lại.

6.4. Ứng dụng công nghệ Agent vào quản lý tài nguyên lưới

6.4.1. Tiềm năng của công nghệ Agent

Hai công nghệ lưới và Agent tập trung vào hai khía cạnh khác nhau của một vấn đề chung, đó là chia sẻ tài nguyên giữa các tổ chức ảo. Lưới được coi như sức mạnh về cơ bắp (Brawn) còn Agent được coi như sức mạnh của trí tuệ (Brain). Trong trường hợp của lưới, sự quan tâm chính tập trung vào cách tổ chức và hoạt động mang tính cơ học như các giao thức chuẩn, các chính sách hoạt động trong lưới, hay các thành viên trong tổ chức nhận ra nhau như thế nào, lời cam kết giữa các thành viện trong tổ chức được định rõ ra sao...Tức là công nghệ này cho phép tạo ra những hệ thống lớn với tập các hoạt động ổn định.

Để phân biệt giữa Sức mạnh cơ bắp và Sức mạnh trí tuệ chúng ta xét các ví dụ sau: Sử dụng những công cụ lưới để truy cập dữ liệu trên những hệ thống lưu trữ khác nhau nhưng không tích hợp ngữ nghĩa của dữ liệu. Cho phép truy cập dịch vụ và trạng thái tài nguyên nhưng không đoán trước được sự thay đổi của trạng thái này... Trong trường hợp này thì lưới chỉ có tính Sức mạnh cơ bắp nhưng không có tính sức mạnh trí tuệ.

Agent có khả năng ra quyết định một cách linh hoạt nhờ có khả năng phán đoán trước các tình thế có thể xảy ra. Nhưng những hành động này của Agent chỉ có ý nghĩa khi được áp dụng vào môi trường phân tán thực tế. Ví dụ, framework Agent cung cấp những khả năng linh hoạt cho Agent nhưng lại không cung cấp sự bảo mật các hoạt động tương tác hoặc tìm kiếm dịch vụ. Do đó có thể nói Agent chỉ có tính Sức mạnh trí tuệ nhưng không có tính sức mạnh cơ bắp.

Như vậy, để lưới làm việc có hiệu quả thì cần nhúng vào lưới khả năng linh hoạt, và khả năng ra quyết định phi tập trung. Cũng như vậy Agent cần có một nền tính toán phân tán đủ mạnh để cho phép chúng tìm kiếm, khám phá, liên minh phục vụ cho việc ra quyết định. Có một vài gợi ý nhỏ cho việc ứng dụng công nghệ Agent vào lưới như lựa chọn nguồn tài nguyên phù hợp dựa trên công nghệ Agent. Trong trường hợp này Agent sử dụng hoạt động lưới để giám sát tình trạng, tìm kiếm nguồn tài nguyên và đệ trình công việc, ngược lại Agent sẽ cung cấp sự đánh giá cho các hoạt động này, cùng với một hệ đa Agent sẽ thực hiện việc

quản lý nguồn tài nguyên toàn cầu rộng lớn mà theo cách tiếp cận khác không thể đạt tới được. Ví dụ thứ 2 là sử dụng công nghệ thương lượng tự động để định rõ vị trí nguồn tài nguyên trong hệ thống lưới. Ở đây, người thiết kế sẽ đánh giá hiệu quả của cả thị trường hàng hoá và những giao thức đấu giá để xác định vị trí nguồn tài nguyên trong hệ thống phân tán.

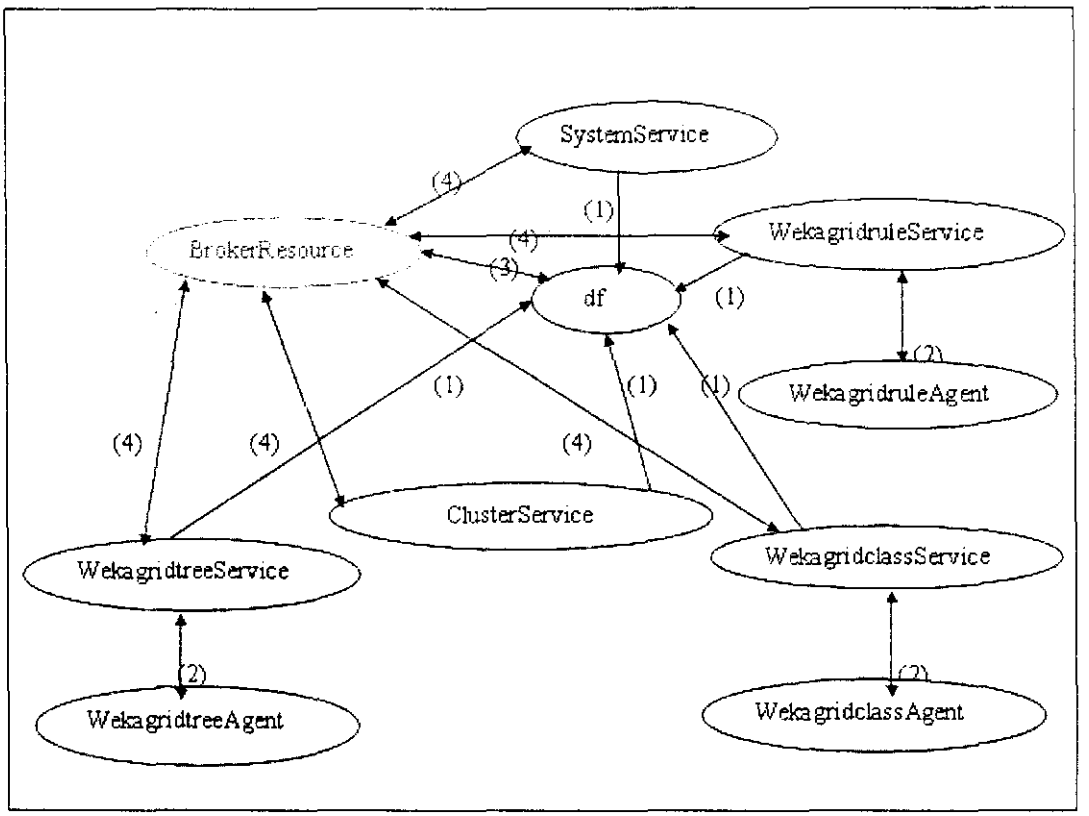
6.4.2. Hệ đa Agent quản lý tài nguyên lưới

Trong lưới Agent thích hợp với việc quản lý, tìm kiếm và định vị nguồn tài nguyên. Ở đây ta sẽ tìm hiểu xem có thể sử dụng hệ thống đa Agent để phát triển một thị trường kinh doanh nguồn tài nguyên tính toán lưới. Như đã đề cập, vấn đề quản lý tài nguyên lưới vẫn được xem là một thách thức lớn. Nó bao gồm việc quản lý yêu cầu người sử dụng, tìm kiếm những nguồn tài nguyên phù hợp nhất với các yêu cầu và lập lịch cho các nguồn tài nguyên đó. Những thách thức ở đây là: quản lý tài nguyên không đồng nhất, các vấn đề về bảo mật, sự thương lượng giữa những người sử dụng tài nguyên và những người cung cấp tài nguyên...

Agent có thể đáp ứng được những khó khăn đó nhờ vào các khả năng linh hoạt, tự động, thông minh. Trong đó, các Agent có thể tương tác, hợp tác hoặc thương lượng để đạt được mục đích riêng và mục đích tổng thể. Chúng ta đưa ra một môi trường Agent mà trong đó có những Agent đại diện cho khách hàng có thể tìm kiếm những Agent đại diện cho dịch vụ lưới, thương lượng với chúng về những điều kiện để sử dụng dịch vụ. Công nghệ Agent cho phép thương lượng linh hoạt bằng cách truyền messages. Hiện nay, có rất nhiều công cụ cho phép phát triển hệ đa Agent. Một trong những công cụ đó ta phải kể đến Jade.

6.4.3. Thiết kế hệ đa tác tử trong BKGrid2005

6.4.3.1. Kịch bản tương tác của hệ đa Agent



Hình 6-10: Kịch bản tương tác trong hệ đa Agent trong hệ thống BKGrid2005

Do hạn chế về cơ sở vật chất nên tài nguyên trong lưới BKGrid chỉ gồm các máy PCs, cluster và các dịch vụ Weka. Hệ đa Agent bao gồm: SystemService cung cấp thông tin về dịch vụ hệ thống; WekagridtreeService, WekagridclassService, WekagridruleService cung cấp thông tin về dịch vụ Weka tương ứng. ClusterService cung cấp dịch vụ Cluster. Hoạt động của hệ như sau:

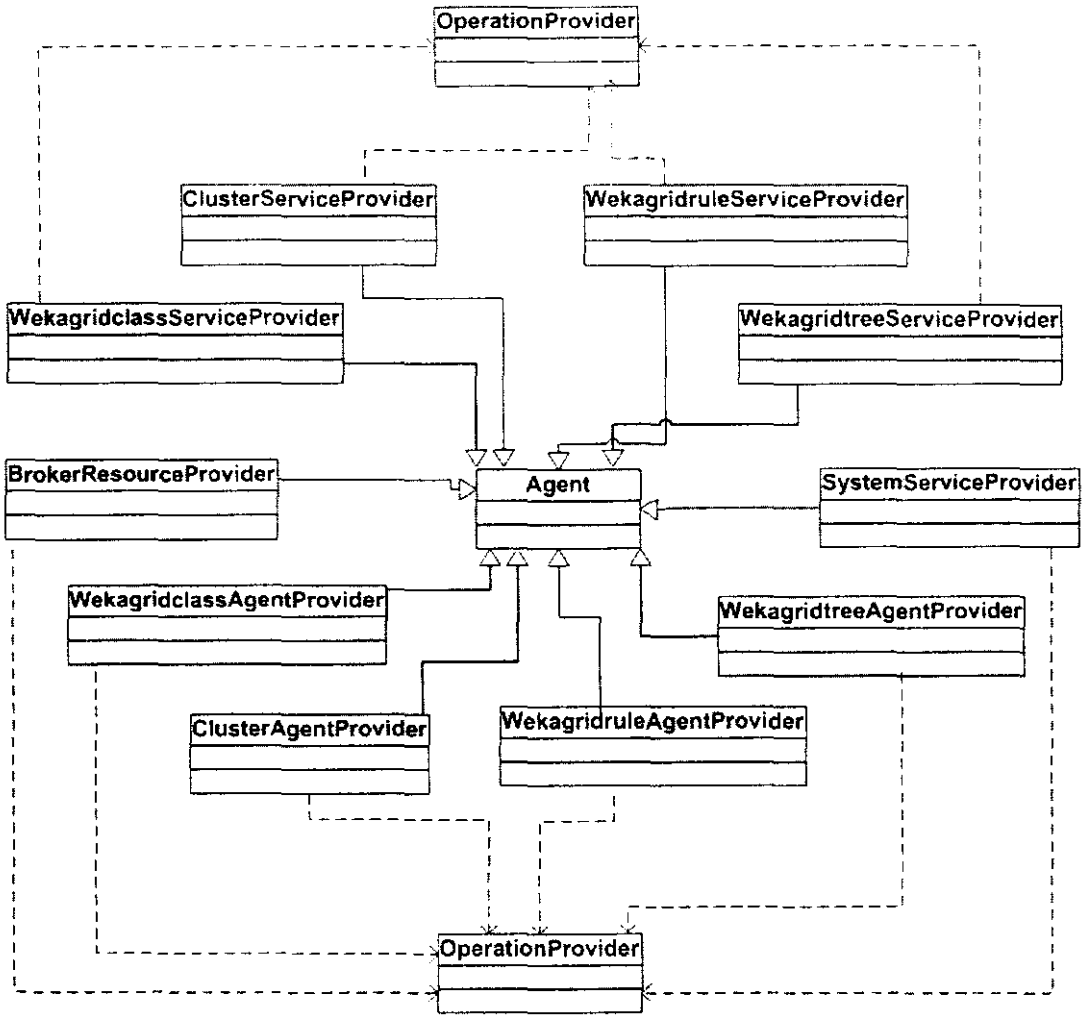
(1) Các Agent đại diện cho nhà cung cấp dịch vụ, khi tham gia vào thị trường lưới sẽ đăng kí dịch vụ mình với trang vàng DF (bản chất cũng là một Agent).

(2) Các Agent cung cấp thông tin dịch vụ thường xuyên cập nhật thông tin về số Instance của Agent Weka.

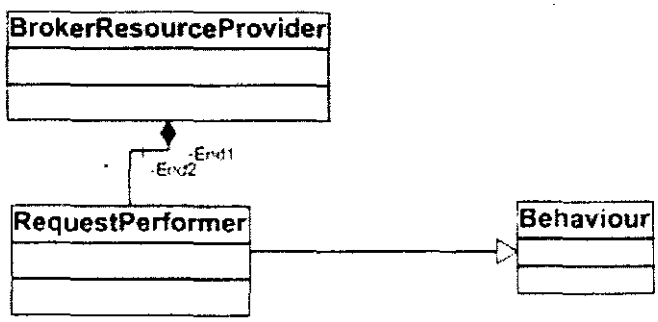
(3) BrokerResource tìm dịch vụ từ trang vàng DF

(4) BrokerResource gửi các Messages thương lượng đến các Agent dịch vụ để sử dụng tài nguyên phù hợp nhất (tối ưu theo giá hoặc tốc độ). Và trả lại tài nguyên khi sử dụng xong.

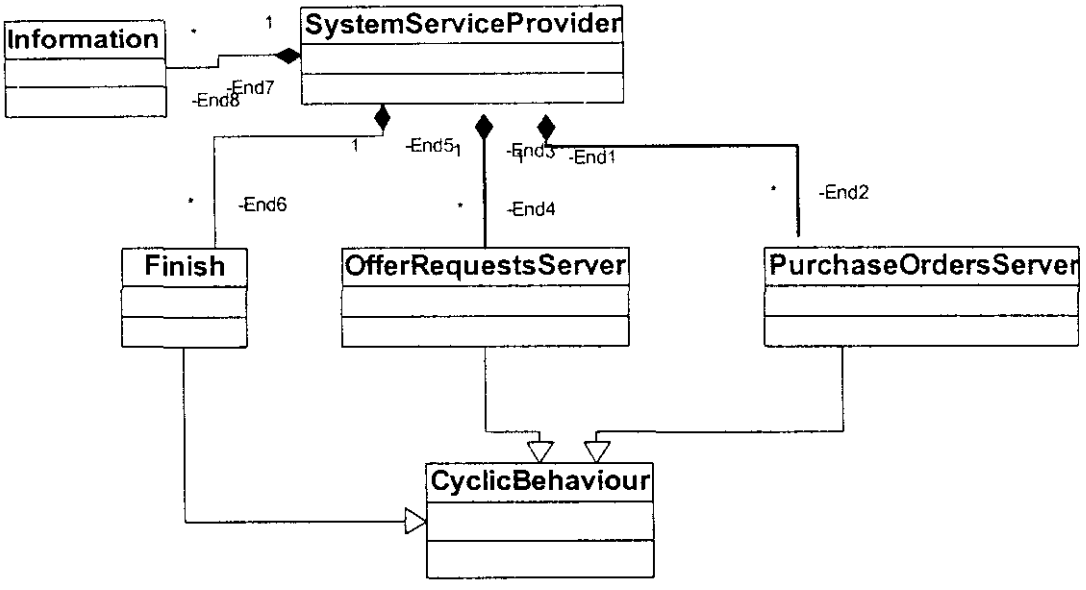
6.4.3.2. Thiết kế lớp



Hình 6-11: Sơ đồ lớp toàn hệ thống

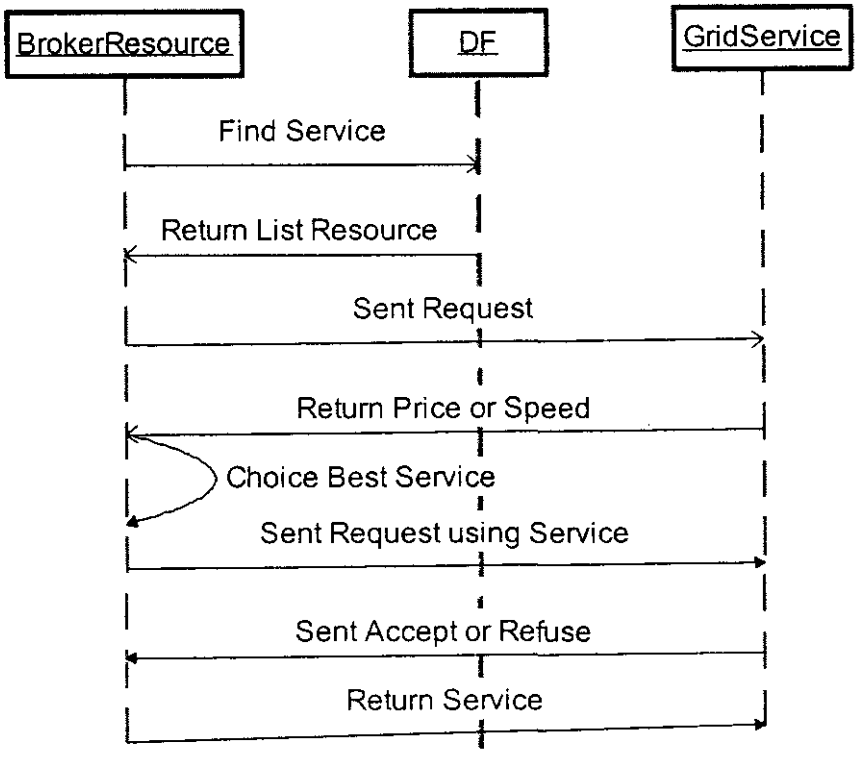


Hình 6-12: Sơ đồ lớp BrokerResourceProvider



Hình 6-13: Sơ đồ lớp SystemServiceProvider

6.4.3.3. Kịch bản thương lượng tài nguyên:



Hình 6-14: Kịch bản tìm kiếm và thương lượng tài nguyên

BrokerResource sẽ truy vấn trên trang vàng DF để có một danh sách các Agent cung cấp dịch vụ mà nó đang tìm kiếm. Sau đó, nó sẽ gửi message yêu cầu đến tất cả các Agent dịch vụ. Các Agent dịch vụ sẽ phản hồi lại cùng với giá và tốc độ của dịch vụ. Căn cứ vào đó, Agent môi giới sẽ lựa chọn tài nguyên phù hợp nhất và gửi message thương lượng đến

Agent cung cấp dịch vụ tốt nhất đó. Nếu được nó sẽ nhận được message đồng ý nếu không nó sẽ bị từ chối. Sau khi dùng xong, Agent môi giới sẽ trả lại dịch vụ.

Chương 7: Bộ lập lịch

Xét từ phía ứng dụng, bộ lập lịch và môi giới tài nguyên là một trong những thành phần trung tâm của BK Grid, bảo đảm chất lượng dịch vụ và hiệu năng hệ thống khi cung cấp cho người dùng. Nhiệm vụ của bộ lập lịch và môi giới tài nguyên bao gồm:

Khám phá tài nguyên.

Chọn tập tài nguyên phù hợp để thực hiện yêu cầu của người dùng theo những tiêu chí tối ưu mà người dùng đặt ra.

Theo dõi quá trình thực hiện ứng dụng và có những xử lý cần thiết khi có những tình huống bất thường xảy ra. Ví dụ: tài nguyên bị lỗi, có tài nguyên mới trong quá trình thực hiện.

Ghi lại những thông tin về lịch sử xử dụng của người dùng làm cơ sở cho việc thanh toán.

Hiển thị thông tin chi tiết về quá trình thực hiện ứng dụng

Trong BK Grid 2005 bộ lập lịch được tích hợp chặt chẽ với ứng dụng cụ thể là ứng dụng Weka. Là thành phần giữ vị trí trung tâm, các bộ lập lịch và môi giới tài nguyên có các mối liên hệ với dịch vụ thông tin hệ thống và các dịch vụ Weka.

Các dịch vụ thông tin trong BK Grid được triển khai trên từng tài nguyên nhằm cung cấp các thông tin về chính tài nguyên đó cũng như các dịch vụ được triển khai trên nó. Việc thu thập các thông tin trên toàn bộ các tài nguyên sẽ do một tập các Agent đảm nhiệm. Trong BK Grid bộ lập lịch và môi giới tài nguyên không tương tác trực tiếp với các dịch vụ thông tin mà thông qua hệ đa Agent này để lấy thông tin, như dịch vụ thông tin hệ thống (SystemService), nhóm các dịch vụ lưới Weka (ClassifierService, ClusterService, AsociationRuleService), ...

Người dùng khi sử dụng các dịch vụ khai phá dữ liệu của Weka sẽ không tương tác trực tiếp với các dịch vụ tính toán đầu cuối này mà sẽ thông qua đại diện của mình là bộ lập lịch và môi giới tài nguyên. Bộ lập lịch nhận các yêu cầu từ người dùng cùng một tập các dữ liệu đầu vào sau đó lựa chọn các dịch vụ tính toán phù hợp để thực hiện sao cho tối ưu được các tiêu chí mà người dùng đã đặt ra.

7.1. Bài toán lập lịch trong tính toán lưới

7.1.1. Bài toán lập lịch trong các môi trường tính toán truyền thống

Bài toán lập lịch hay lập kế hoạch là một bài toán ta rất hay gặp trong thực tế. Có thể kể ra như lập thời khóa biểu, lập kế hoạch quản lý dự án... đều là các bài toán lập lịch tiêu biểu và rất kinh điển. Nội dung của bài toán là tìm ra một bản kế hoạch thực hiện các công việc một cách tối ưu nhất dựa vào tất cả các thông tin đầu vào có được từ tình trạng hiện tại của hệ thống, tuy nhiên tùy theo bài toán cụ thể mà yêu cầu và phương pháp giải quyết của các bài toán này rất khác nhau.

Trong ngành khoa học máy tính thuật ngữ lập lịch xuất hiện từ thời kỳ đầu. Ở thế hệ máy tính thời kỳ đầu việc lập lịch cho máy tính hoạt động do con người đảm nhiệm, thường là do người quản trị trung tâm máy tính. Máy tính thời kỳ này cực kỳ đắt, khả năng tính toán rất hạn chế, không có hệ điều hành, điều khiển rất phức tạp, việc chuẩn bị chương trình cho nó cũng rất công phu. Các chương trình cho máy tính thời kỳ này chủ yếu là các chương trình tính toán không cần tương tác với người dùng. Quản trị viên trung tâm máy tính sẽ gom các yêu cầu chạy chương trình của nhiều người sử dụng lại lập thành một lô (batch) cho máy tính thực hiện tuần tự và trả về kết quả cho người dụng. Thứ tự thực hiện các công việc này do quản trị viên quyết định. Anh ta có thể thực hiện một vài thuật toán đơn giản để xem thứ tự thực hiện nào là tốt nhất ứng với một tiêu chí nào đó, ví dụ thời gian chờ trung bình nhỏ nhất.

Các máy tính thế hệ thứ 2 và 3 đã mạnh hơn rất nhiều so với máy tính thời kỳ đầu. Chúng có hệ điều hành phụ trách công việc lập lịch. Hệ điều hành nhận yêu cầu chạy chương trình từ nhiều người dùng và sắp xếp thứ tự thực hiện các công việc này sao cho phù hợp với kiểu máy tính. Ví dụ máy tính đơn bộ xử lý thường áp dụng thuật toán quay vòng để đáp ứng yêu cầu của nhiều người dùng ; với các máy tính đa bộ xử lý, hệ điều hành có thể phân phối các tiến trình lên các bộ xử lý khác nhau để chạy song song thực sự. Các tiêu chí đánh giá thường là thời gian hồi đáp, thông lượng...

Khi công nghệ mạng đã phát triển mạnh mẽ thì nhu cầu kết nối máy tính lại với nhau nhằm tạo ra sức mạnh tính toán lớn hơn xuất hiện, cùng với đó là nhu cầu xây dựng các hệ quản lý tài nguyên phân tán để quản lý hiệu quả hơn các nguồn tài nguyên này. Các hệ quản lý tài nguyên nổi tiếng có thể kể ra như PBS, LSF, Condor, Legion... Bộ lập lịch là một thành phần của hệ quản lý tài nguyên có trách nhiệm phân phối các nguồn tài nguyên đến các yêu cầu của người dùng. Bộ lập lịch này hoạt động ở mức cao hơn các bộ lập lịch trong các tài nguyên địa phương và thường là hướng hệ thống.

7.1.2. Bài toán lập lịch trong tính toán lưới

Công nghệ tính toán lưới cũng là tính toán phân tán nhưng ở mức phát triển cao hơn nữa, thể hiện ở chỗ nó liên kết các nguồn tài nguyên không chỉ ở trong một tổ chức mà từ nhiều tổ chức cùng tham gia. Bộ lập lịch cũng là một thành phần của hệ quản lý tài nguyên (hướng hệ thống) hoặc một thành phần được nhúng ngay trong ứng dụng (hướng ứng dụng) và nhiệm vụ của nó là kết hợp các nguồn tài nguyên không chỉ từ một mà từ nhiều tổ chức, do vậy nó cũng có một số điểm khác biệt sau:

- Trong các mô hình tính toán truyền thống bộ lập lịch thay mặt người dùng là chủ sở hữu tài nguyên do vậy có toàn quyền sử dụng các tài nguyên theo nhu cầu; trong tính toán lưới người dùng chỉ là một thành viên của lưới do vậy bị hạn chế về khả năng sử dụng tài nguyên.

- Trong các mô hình tính toán truyền thống thông tin cung cấp cho bộ lập lịch thường là tương đối đầy đủ, chính xác và cập nhật. Trong tính toán lưới do đặc điểm phức tạp của môi trường lưới nên nhìn chung các bộ lập lịch buộc phải hoạt động trong môi trường thiếu thông tin đầy đủ và chính xác.
- Mô hình ứng dụng trong tính toán lưới cũng đa dạng hơn rất nhiều so với các mô hình tính toán truyền thống do vậy người ta thường nghiên cứu những thuật toán cụ thể để đáp ứng nhu cầu cho những dạng ứng dụng cụ thể.
- Do quy mô của lưới thường là rất lớn nên việc lập lịch trong lưới có thể được chia ra làm nhiều cấp độ khác nhau, có những bộ lập lịch làm việc ở mức cao (Meta-scheduler) điều phối những bộ lập lịch hoạt động ở mức độ thấp hơn (Local – Scheduler).

Hiện nay việc nghiên cứu các thuật toán lập lịch cho lưới là một vấn đề khá mới mẻ không chỉ ở Việt Nam mà ngay cả ở trên thế giới. Lý do khá đơn giản là vì bản thân tính toán lưới cũng đang ở trong giai đoạn phát triển liên tục và hiện nay có rất nhiều hướng nghiên cứu cho vấn đề này. Trong phần tiếp chúng ta sẽ cũng tìm hiểu một cách tiếp cận khá mới mẻ cho vấn đề lập lịch trong lưới, đó là cách tiếp cận hướng đến mục tiêu kinh tế.

7.2. Mô hình lưới mang tính kinh tế và bộ lập lịch mang tính kinh tế

7.2.1. Một số mô hình tính toán truyền thống.

7.2.1.1. Máy tính lớn

Đây là mô hình tính toán và quản lý tài nguyên tập trung rất điển hình. Xuất hiện ngay từ những ngày đầu của ngành tin học, những chiếc máy tính này hết sức đắt và được dành để giải những bài toán quan trọng. Mọi tài nguyên đều nằm tập trung tại trung tâm máy tính dưới sự quản lý tập trung của nhân viên điều hành. Công việc được xử lý chủ yếu là theo chế độ lô (batch), khả năng tương tác hết sức hạn chế. Việc quyết định các công việc được thực hiện theo thứ tự nào hoàn toàn do người quản lý quyết định nghĩa là việc lập lịch hoàn toàn do con người đảm nhiệm. Tuy rất đơn giản nhưng có những phương pháp lập lịch trong thời kỳ này đã được áp dụng vào những mô hình tính toán tiên tiến hơn. Ví dụ như việc xây dựng các hàng đợi và xác định độ ưu tiên của người dùng, việc sắp xếp các công việc sao cho thời gian quay vòng hay thời gian chờ là nhỏ nhất có thể...

Một bước phát triển hơn của mô hình này là mô hình tính toán trong đó một máy tính phục vụ nhiều người dùng thông qua nhiều thiết bị đầu cuối (terminal). Các thiết bị đầu cuối về cơ bản không có khả năng xử lý mà chỉ đóng vai trò hiển thị các kết quả và chuyển các yêu cầu của người dùng về trung tâm để xử lý. Máy tính thời kỳ này được quản lý bởi các hệ điều hành đa nhiệm, đa người dùng mà nổi tiếng nhất là hệ điều hành UNIX. Trong mô hình này, hệ điều hành nắm quyền quản lý toàn bộ tài nguyên của hệ thống và phân phối các tài nguyên này đến người dùng tùy theo yêu cầu, tính chất công việc và độ ưu tiên của người

dùng. Hệ điều hành làm nhiệm vụ sắp xếp lịch thực hiện cho các tiến trình của người dùng sao cho khả năng tận dụng tài nguyên hệ thống là tối đa. Ví dụ như khi có một tiến trình đang để chờ nhập xuất dữ liệu nó có thể bị block lại để cho tiến trình khác đang cần CPU được chạy, hoặc như các tiến trình chạy ở chế độ lô có thể có độ ưu tiên thấp hơn các tiến trình chạy ở chế độ tương tác...

7.2.1.2. Máy tính cá nhân

Cùng với sự phát triển của công nghệ điện tử thời kỳ bùng nổ của các máy tính cá nhân. Những máy tính cá nhân phổ biến nhất thuộc về các máy tính theo kiến trúc IBM-PC chạy trên hệ điều hành MS DOS. Hệ điều hành MS DOS thời kỳ đầu thậm chí còn là một hệ điều hành đơn nhiệm đơn người sử dụng. Mãi đến khi phát triển các phiên bản Windows cho đến tận Windows NT, Windows XP, Windows 2000 Microsoft mới phát triển nó thành một hệ thống đa nhiệm đa người dùng đầy đủ. Tài nguyên tính toán ở đây tất nhiên được quản lý hoàn toàn tập trung bởi hệ điều hành và người sử dụng hầu như không thể can thiệp gì vào quá trình này. Ví dụ: lượng tài nguyên dành cho tiến trình nào là bao nhiêu, nó được chạy vào thời điểm nào...

7.2.1.3. Mạng máy tính

Sự phổ biến rộng rãi của máy tính cá nhân đã dẫn đến nhu cầu tất yếu là phải kết nối các hệ thống này lại để trao đổi thông tin và các nguồn tài nguyên. Các mạng máy tính đã phát triển từ các mạng cục bộ thành mạng toàn cầu Internet như hiện nay.

Công nghệ mạng máy tính phát triển đã sinh ra nhiều mô hình tính toán mới: mô hình tính toán phân tán dựa trên mô hình client – server, mô hình tính toán bó (cluster), mô hình tính toán dựa trên Web... và mới nhất hiện nay là mô hình tính toán lưới.

Tài nguyên trong các mô hình tính toán này có thể được quản lý theo kiểu tập trung hoặc phân tán. Ví dụ như trong một doanh nghiệp các phòng ban có tài nguyên tính toán riêng và tương tác với nhau theo kiểu mô hình client – server. Mô hình tính toán lưới ra đời cho phép phân bổ tài nguyên hiệu quả giữa các phòng ban chức năng trong doanh nghiệp, cho phép tận dụng hiệu quả hơn các nguồn tài nguyên rỗi.

Tuy tài nguyên phân tán về mặt địa lý và được quản lý bởi nhiều đơn vị khác nhau nhưng trong môi trường lưới chúng lại được quản lý một cách khá tập trung theo nghĩa mọi nguồn tài nguyên rỗi đều được quản lý tập trung và mọi yêu cầu phân bổ tài nguyên cũng được tập trung lại để xử lý.

7.2.2. Mô hình tính toán lưới hướng mục tiêu kinh tế

7.2.2.1. Giới thiệu

Tính toán lưới là một công nghệ rất hứa hẹn cho phép giải quyết những bài toán lớn trong khoa học, kỹ thuật và kinh tế. Nó cho phép chia sẻ và kết hợp hàng triệu tài nguyên phân tán về mặt địa lý thuộc về nhiều tổ chức. Lưới được tạo nên từ những tài nguyên không đồng

nhất từ phần cứng (PC, workstation, cluster, super computer...), phần mềm quản trị hệ thống (hệ điều hành, các hệ theo kiểu hàng đợi, ví dụ: PBS...) đến các phần mềm ứng dụng (khoa học, kỹ thuật, kinh tế) với những đòi hỏi khác nhau về tài nguyên (CPU, không gian lưu trữ, đường truyền mạng...). Các tài nguyên thuộc sở hữu bởi những tổ chức khác nhau với những chính sách quản lý tài nguyên riêng biệt, chính sách định giá cho người dùng khác nhau ở những thời điểm khác nhau. Cũng như vậy, sự sẵn sàng của tài nguyên và tải trên chúng cũng biến động theo thời gian.

Trong một môi trường lưới như vậy người cung cấp (người sở hữu tài nguyên và người tiêu thụ (người dùng) có những mục tiêu và chiến lược hoạt động khác nhau. Quan trọng hơn, cả người cung cấp và người sở hữu đều nằm phân tán về mặt địa lý với những múi giờ khác nhau. Trong một môi trường phức tạp như vậy những cách tiếp cận truyền thống trong quản lý tài nguyên với mục tiêu tối ưu hóa các tham số đo hiệu năng của toàn bộ hệ thống không thể hoạt động hiệu quả được. Các phương pháp truyền thống sử dụng các chính sách quản lý tập trung cần phải có đầy đủ thông tin về toàn bộ hệ thống và một chính sách quản lý tài nguyên chung hoặc là chính chính sách quản lý phân tán một phần. Trong môi trường lưới rất khó có thể xác định được các tham số hiệu năng cho toàn bộ hệ thống và một chính sách quản lý tài nguyên chung.

Từ những khó khăn như vậy một số các nhà khoa học trên thế giới đã đề xuất ra một số mô hình mới trong việc quản lý tài nguyên lưới và một trong những mô hình rất hứa hẹn đó là mô hình lưới hướng đến mục tiêu kinh tế (GRACE – Grid Architecture for Computation Economy) do trường đại học Monach của Australia phát triển. Chúng ta sẽ cùng tìm hiểu về mô hình này.

Hầu hết các hệ thống quản lý tài nguyên và lập lịch (như Legion, Condor, AppleS PST, NetSolve...) đều áp dụng một chiến lược truyền thống: bộ lập lịch lựa chọn công việc nào sẽ được thực hiện trên tài nguyên nào dựa vào những hàm lượng giá được đánh giá bởi các tham số hướng hệ thống. Chúng tập trung vào việc cải thiện hiệu năng của toàn hệ thống được đánh giá bởi các tiêu chí như thông lượng, khả năng tận dụng CPU và thời gian hoàn thành ứng dụng sớm nhất. Các tham số như giá sử dụng tài nguyên không được tính đến, điều đó có nghĩa là giá của việc chạy ứng dụng sẽ là như nhau vào mọi thời điểm – mà điều này có vẻ không đúng trong thực tế: giá của việc thực hiện ứng dụng phải cao hơn khi có những sức ép về chất lượng thực hiện vi dụ: thời gian hoàn thành ứng dụng. Người dùng không muốn phải trả giá cao nhất mà muốn thương lượng một mức giá nhất định dựa vào nhu cầu, độ ưu tiên của ứng dụng và kinh phí của họ. Trong cách tiếp cận hướng kinh tế, quyết định lập lịch được thực hiện động trong thời gian chạy và chúng được định hướng bởi nhu cầu của chính người sử dụng. Trong khi các cách tiếp cận truyền thống thường định giá qua các tài nguyên phần cứng và phần mềm cần thiết để chạy ứng dụng thì trong mô hình kinh tế, người dùng chỉ phải trả giá cho phần lợi ích thực mà họ được hưởng. Việc định giá

dựa trên nhu cầu của người dùng và khả năng cung ứng tài nguyên sẽ là động lực cạnh tranh chính trong mô hình lưới kinh tế: người dùng cạnh tranh với người dùng trong việc giành quyền sử dụng tài nguyên còn các nhà cung cấp sẽ cạnh tranh với nhau trong việc cung cấp tài nguyên và thu hút người sử dụng. Việc sử dụng mô hình kinh tế sẽ mang lại nhiều động lực cho cả người sử dụng và nhà cung cấp để họ trở thành một thành viên của cộng đồng lưới. Tuy hiện nay việc ứng dụng công nghệ lưới mới chỉ ở mức phổ biến trong giới làm khoa học nhưng việc ứng dụng các mô hình kinh tế vẫn mang lại những lợi ích nhất định. Người quản trị hệ thống có thể đặt ra những giới hạn về sử dụng tài nguyên (Quota) cho từng người dùng và định giá các tài nguyên khác nhau với số lượng thẻ bài (token) khác nhau. Những chính sách như vậy sẽ khiến cho người dùng có ý thức hơn trong việc sử dụng hợp lý các nguồn tài nguyên hiện có tùy theo nhu cầu, mục đích của mình. Ví dụ nếu một bài toán có tầm quan trọng không cao và không có ràng buộc chặt chẽ về mặt thời gian có thể được sắp xếp để được chạy trên những tài nguyên giá rẻ hơn. Điều này mang lại lợi ích cho chính người sử dụng đó và cả những người sử dụng khác có những yêu cầu cao hơn về tài nguyên được sử dụng những tài nguyên phù hợp hơn với họ.

- Một số vấn đề mới cần giải quyết khi triển khai mô hình lưới kinh tế:

Một thư mục thị trường trong đó có thông tin về các tài nguyên lưới.

Mô hình đánh giá giá trị tài nguyên và tính giá sử dụng cho người dùng.

Các giao thức thương lượng sử dụng tài nguyên

Các hệ thống thanh toán.

Các bộ lập lịch và môi giới tài nguyên hướng đến yêu cầu và lợi ích của người dùng.

- Một số lợi ích của mô hình lưới hướng kinh tế:

Nó khuyến khích người sở hữu tài nguyên đóng góp những nguồn tài nguyên nhàn rỗi của họ vào lưới và được hưởng lợi từ đó.

Nó cho phép cân bằng giữa cung và cầu tài nguyên.

Khuyến khích người dùng sử dụng các nguồn tài nguyên phù hợp với nhu cầu của họ. Ví dụ: dùng các nguồn tài nguyên chậm và rẻ cho các bài toán có độ ưu tiên thấp và do đó ưu tiên những nguồn tài nguyên mạnh để giải quyết các bài toán quan trọng.

Nó cho phép loại bỏ sự cần thiết của bộ quản lý tài nguyên tập trung.

Cho phép người dùng đặc tả những yêu cầu và mục tiêu của họ.

Cho phép phát triển một chiến lược lập lịch hướng người dùng hơn là hướng hệ thống.

Cho phép định ra một cơ chế hiệu quả cho việc định vị và quản lý tài nguyên.

Cho phép cả người sử dụng và người cung cấp tài nguyên có nhiều quyền quyết định hơn với các nguồn lực mà họ hiện có sao cho đạt được lợi ích tối đa.

Trong mục tiếp theo đây chúng ta sẽ lần lượt tìm hiểu từng thành phần trong một mô hình lưới hướng kinh tế.

7.2.2.2. Các thành phần của mô hình lưới hướng kinh tế

Trong một mô hình lưới hướng kinh tế có các thành phần chính sau:

Nhà cung cấp dịch vụ/tài nguyên.

Người tiêu thụ tài nguyên/dịch vụ

Các dịch vụ trung gian và dịch vụ gia tăng

- Nhà cung cấp dịch vụ/tài nguyên

Đây là những người đóng góp một số loại tài nguyên, dịch vụ nhất định nào đó cho cộng đồng lưới. Như ta đã biết, tính toán lưới tập trung vào việc chia sẻ và sử dụng tài nguyên trên một quy mô hết sức rộng lớn và thành viên của lưới bao gồm nhiều tổ chức khác nhau hoạt động tương đối độc lập. Trong mô hình lưới hướng kinh tế, mức độ độc lập này được đẩy lên một mức cao hơn nữa: các tổ chức hoạt động độc lập và còn cạnh tranh với nhau trong việc mang lại cho người dùng chất lượng dịch vụ tốt nhất với giá cả phải chăng nhất. Mục tiêu của người cung cấp dịch vụ là thu được lợi nhuận tối đa từ những nguồn tài nguyên dịch vụ mà họ cung cấp.

Các ứng dụng của người dùng có những yêu cầu khác nhau về tài nguyên tùy thuộc vào công việc tính toán được thực hiện và thuật toán được sử dụng. Một số ứng dụng yêu cầu nhiều sức mạnh tính toán trong khi đó một số khác lại thiên về trao chuyển dữ liệu hoặc tổng hợp cả hai. Ví dụ một ứng dụng yêu cầu nhiều tính toán (CPU Intensive) có thể chỉ cần tính phí thời gian sử dụng CPU trong khi miễn phí các thao tác vào ra. Tuy nhiên mô hình này không thể áp dụng được cho các ứng dụng dạng vào ra dữ liệu (I/O Intensive Applications).

Các nhà cung cấp có thể cung cấp và tính phí các loại tài nguyên như:

Thời gian sử dụng CPU.

Dung lượng bộ nhớ sử dụng.

Thông lượng đường truyền sử dụng.

Các phần mềm và thư viện hệ thống sử dụng.

Việc truy cập đến các tài nguyên này có thể là đơn lẻ hay tổ hợp. Truy nhập theo kiểu kết hợp sẽ đòi hỏi thành lập một ma trận giá để tính phí truy nhập tài nguyên.

- Người sử dụng dịch vụ/tài nguyên lưới

Người sử dụng dịch vụ, tài nguyên lưới là những người dùng có các bài toán cần được giải quyết bằng các nguồn tài nguyên, dịch vụ mà lưới cung cấp. Thông qua các dịch vụ thông tin trong môi trường lưới người sử dụng có thể tự tìm các nguồn tài nguyên phù hợp để giải quyết bài toán của họ. Trong trường hợp này người dùng trực tiếp làm công việc của một bộ lập lịch và môi giới tài nguyên. Tuy nhiên trong thực tế điều này thường không xảy ra. Người dùng sẽ ủy nhiệm cho một phần mềm làm công việc này, đó chính là các bộ lập lịch và môi giới tài nguyên. Bộ lập lịch và môi giới tài nguyên này đảm nhận toàn bộ các công việc từ khâu nhận yêu cầu của người dùng cho đến khi trả lại các kết quả.

Mục tiêu của bộ lập lịch này là mang lại nhiều lợi ích nhất có thể được cho người dùng trong những giới hạn về ngân sách và thời gian mà họ đặt ra. Và đây cũng chính là nhiệm vụ của tài liệu này: xây dựng một bộ lập lịch và môi giới tài nguyên hướng kinh tế.

- Các dịch vụ trung gian và các dịch vụ gia tăng

Để tăng thêm các tiện ích sử dụng lưới cho người dùng, trong lưới người ta xây dựng thêm một số dịch vụ trung gian và dịch vụ gia tăng. Sau đây là một số dịch vụ tiêu biểu:

+ Dịch vụ thông tin thị trường lưới

Đây là một trong những dịch vụ quan trọng nhất của lưới. Người cung cấp các tài nguyên và dịch vụ muốn quảng bá các tài nguyên, dịch vụ mà mình cung cấp sẽ thực hiện điều này nhờ các dịch vụ thông tin. Thông qua dịch vụ này nhà cung cấp sẽ cung cấp các thông tin cơ bản về tài nguyên như kiểu loại tài nguyên (kiến trúc phần cứng, hệ điều hành, tốc độ xử lý...), kiểu loại dịch vụ cung cấp (loại dịch vụ gì, chức năng gì...) và chi phí sử dụng các loại tài nguyên dịch vụ này. Đến lượt mình người sử dụng thông qua đại diện là bộ lập lịch và môi giới tài nguyên sẽ truy cập vào danh mục thông tin này để lấy ra các thông tin về tài nguyên dịch vụ mà mình đang quan tâm, từ đó lựa chọn ra các tài nguyên dịch vụ phù hợp nhất với những yêu cầu và mục tiêu của mình.

+ Các cổng truy cập dịch vụ lưới, cổng ứng dụng (Portal)

Đây là các dịch vụ gia tăng được xây dựng trên nền các dịch vụ cơ bản của lưới. Các cổng truy cập lưới cung cấp một giao diện tích hợp cho phép người dùng sử dụng toàn bộ các dịch vụ lưới và cả các ứng dụng đã được xây dựng sẵn trên Portal. Ví dụ trong BK Grid 2005 người dùng vừa có thể trực tiếp sử dụng các dịch vụ khai phá dữ liệu Weka vừa có thể sử dụng các ứng dụng khai phá dữ liệu được xây dựng sẵn trên Portal. Trong cách thứ hai này người dùng có nhiều lựa chọn hơn vì các ứng dụng này đã được tích hợp sẵn bộ lập lịch và môi giới tài nguyên. Hơn nữa các ứng dụng này cũng cung cấp những dịch vụ tinh vi và phức tạp hơn do chúng kết hợp nhiều loại dịch vụ cơ bản sẵn có mà lưới cung cấp.

+ Các dịch vụ thanh toán

Đây là những dịch vụ hết sức quan trọng góp phần hiện thực hóa một mô hình lưới hướng kinh tế. Ở mức đơn giản nhất từng tài nguyên và dịch vụ của mình sẽ ghi lại chi tiết tất cả các giao dịch của từng người sử dụng lưới và thanh toán trực tiếp với người dùng. Ở các mức cao hơn sẽ có các thành phần trung gian như ngân hàng lưới làm nhiệm vụ thanh toán kết hợp với các cơ chế thanh toán điện tử. Tuy nhiên nhìn chung vấn đề thanh toán trong lưới là một vấn đề khá phức tạp và các nghiên cứu trên thế giới cũng chỉ mới ở trong giai đoạn khởi đầu.

Trong thời gian tới các cơ chế thanh toán trong môi trường lưới sẽ được phát triển để hỗ trợ nhiều thể thức thanh toán linh hoạt như:

Trả trước – người mua phải mua được quyền sử dụng trước khi sử dụng tài nguyên.

Trả sau – chi tiết về việc sử dụng tài nguyên của người dùng được ghi lại và sau đó việc thanh toán được tiến hành theo định kỳ.

Trả trong khi sử dụng (Pay as you go)

Mỗi nhà cung cấp dịch vụ có thể trực tiếp tính phí người dùng và thực hiện mọi xử lý liên quan đến quá trình tranh toán. Tuy nhiên điều này sẽ mang lại rất nhiều bất tiện cho cả người cung cấp lẫn người sử dụng khi ở trong môi trường lưới quy mô lớn. Để đơn giản hóa vấn đề này cần thiết phải có các cơ chế trung gian như các ngân hàng lưới. Các bộ môi giới của người dùng sẽ tự động thông báo cho nhà cung cấp các thông tin về tài khoản của người dùng trong ngân hàng lưới để họ tự động chuyển khoản bằng các cơ chế điện tử. Điều đó cũng có nghĩa là cơ sở hạ tầng thanh toán trong lưới phải phát triển đến mức có thể tích hợp được các cơ chế thanh toán điện tử đang được sử dụng rộng rãi hiện nay như:

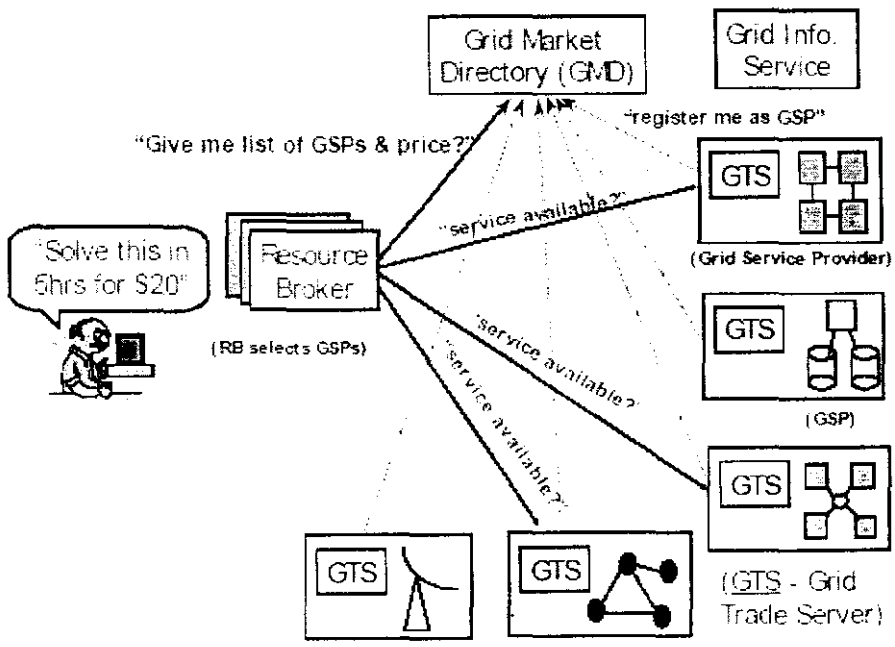
NetCheck – Người dùng đăng ký với máy chủ NetCheck có thể viết các sec điện tử và gửi cho nhà cung cấp. Tiền sẽ được chuyển khoản tự động từ tài khoản của người dùng đến tài khoản của nhà cung cấp.

NetCash – Cơ chế thanh toán bằng tiền điện tử.

Paypal – hệ thống thanh toán dựa trên thẻ tín dụng

7.2.2.3. Cơ chế hoạt động của một mô hình lưới hướng kinh tế

Hình vẽ sau minh họa cơ chế hoạt động của một mô hình lưới hướng kinh tế.



Hình 7-1: Cơ chế hoạt động của một mô hình lưới hướng kinh tế

- Để gia nhập thị trường lưới trước hết cả người sử dụng lẫn người cung cấp dịch vụ đều phải là thành viên của môi trường lưới. Việc ra nhập lưới đòi hỏi phải cài đặt các phần mềm

phù hợp với các phần mềm cơ bản dùng để xây dựng lưới đó. Hiện nay các bộ công cụ xây dựng lưới đang được thiết kế theo hướng chuẩn hóa xung quanh mô hình kiến trúc lưới hướng dịch vụ OGSA giúp cho các lưới có thể dễ dàng giao tiếp được với nhau. Ngoài ra một điều hết sức quan trọng khác là phải có các nền tảng bảo mật cần thiết. Ví dụ: các giấy chứng nhận điện tử. Đây là nền tảng cho mọi giao dịch trong môi trường lưới, tạo cơ sở tin tưởng giữa người cung cấp và người sử dụng dịch vụ và là cơ sở thanh toán sau này.

- Sau khi đã ra nhập môi trường lưới, để quảng bá dịch vụ mình cung cấp người cung cấp dịch vụ phải thông qua các thư mục thị trường (Grid Market Directory - GMD). Đây cũng thực chất là các dịch vụ thông tin trong môi trường lưới tuy nhiên có thêm nhiều thông tin về các khía cạnh kinh tế có liên quan đến việc sử dụng dịch vụ. Ví dụ: giá cả, cách tính phí,...
- Người sử dụng, thông qua đại diện là các bộ lập lịch và môi giới tài nguyên của mình khi muốn sử dụng tài nguyên lưới trước tiên sẽ phải tìm đến tra trong các thư mục thị trường để tìm ra được các tài nguyên và dịch vụ phù hợp. Trong các GMD sẽ có các bộ truy vấn thông tin phục vụ cho công việc tìm kiếm của người dùng. Các cơ sở để truy vấn phụ thuộc vào tính chất công việc và các yêu cầu cụ thể của người dùng về cách thức thực hiện công việc đó. Ví dụ: tìm các dịch vụ để giải quyết một bài toán X với mức chi phí là a$ và tốc độ thực hiện b giờ.
- Sau khi đã có trong tay danh sách các tài nguyên thỏa mãn các tiêu chí ban đầu, bộ lập lịch và môi giới tài nguyên sẽ liên hệ trực tiếp với các tài nguyên để biết thêm thông tin chi tiết về tình hình hiện tại của tài nguyên. Ví dụ: mức tải hiện thời... Trong giai đoạn này sẽ có một số bước thương lượng giữa người cung cấp và người sử dụng để đạt được quyền sử dụng tài nguyên ví dụ: thương lượng về mức giá thực, chất lượng dịch vụ...
- Cuối cùng sau khi đã thành công trong việc thương lượng – thuận mua vừa bán –công việc của người dùng sẽ được thực hiện trên tài nguyên của người cung cấp. Mọi thông tin chi tiết về quá trình thực hiện công việc sẽ được lưu lại để làm cơ sở cho quá trình thanh toán và giải quyết các tranh chấp nảy sinh sau này.

7.2.2.4. Chi tiết hoạt động của một bộ lập lịch và môi giới tài nguyên

Trong mô hình lưới hướng kinh tế bộ lập lịch và môi giới tài nguyên là một trong những thành phần quan trọng nhất quyết định tính hiệu quả của toàn bộ hoạt động của lưới. Bộ lập lịch được nghiên cứu và phát triển trong tài liệu này là một bộ lập lịch hướng người dùng và hướng ứng dụng, điều đó có nghĩa là:

- Bộ lập lịch được thiết kế để phục vụ riêng cho từng người dùng. Mỗi người dùng sẽ có một bộ lập lịch của riêng mình chịu trách nhiệm thực hiện yêu cầu của riêng người dùng đó và các bộ lập lịch này hoạt động độc lập với nhau. Có thể nói ý tưởng này không hề mới tuy nhiên nó thường không được áp dụng cho các mô hình lập lịch truyền thống. Ta sẽ phân tích cụ thể hơn các lý do dẫn đến việc áp dụng ý tưởng này trong môi trường tính toán lưới:

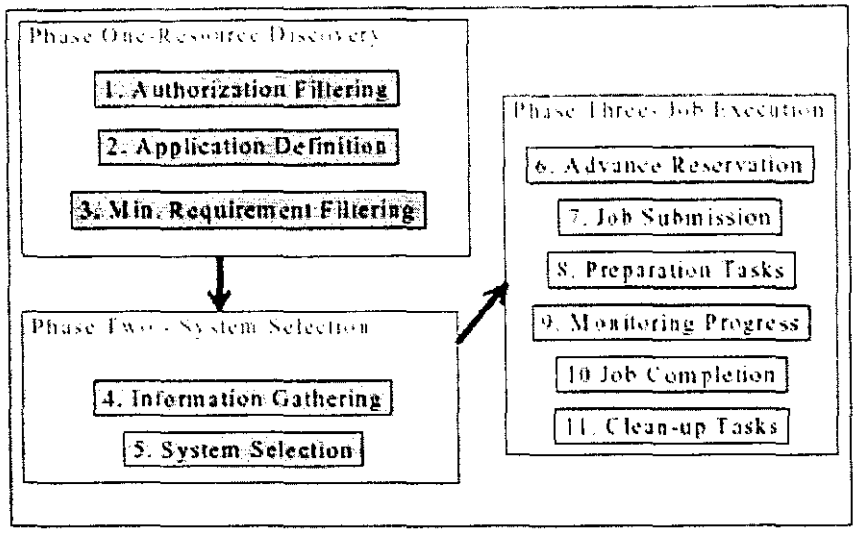
Trong các môi trường tính toán truyền thống tài nguyên thường thuộc về một tổ chức hay cá nhân. Người sở hữu tài nguyên có toàn quyền quyết định cách thức sử dụng tài nguyên sao cho phù hợp với các mục tiêu của họ. Mặt khác việc quản lý thông tin trong các môi trường tính toán truyền thống thường tương đối hiệu quả và tin cậy, và do vậy trong các hệ thống tính toán truyền thống người ta thường áp dụng các chiến lược lập lịch tập trung hoặc phân tán với mục tiêu chủ yếu là tập trung vào việc tăng hiệu năng của toàn bộ hệ thống.

Ngược lại, môi trường lưới có sự góp mặt của nhiều cá nhân hay tổ chức hoạt động vì những mục tiêu khác nhau. Việc xây dựng một hệ thống quản lý theo kiểu tập trung với mục tiêu làm tăng hiệu năng của toàn bộ hệ thống sẽ không mang lại hiệu quả cao nữa. Do tính chất rộng lớn và phức tạp của môi trường lưới mà việc xây dựng một hệ thống quản lý như vậy sẽ cực kỳ phức tạp và tốn kém mà hiệu quả mang lại không cao do phải hoạt động trong môi trường kém tin cậy hơn rất nhiều so với các môi trường tính toán truyền thống. Như vậy có lẽ hiệu quả hơn là ta sẽ giao cho từng thành viên của lưới toàn quyền tự quyết hành vi của mình. Đó cũng chính là ý tưởng của mô hình lưới hướng kinh tế trong đó các thành viên lưới hoạt động như trong một nền kinh tế thị trường. Điều này hứa hẹn mang lại sự năng động cho từng thành viên trong hoạt động để đạt được mục đích của mình.

Phần này tập trung vào thiết kế một bộ lập lịch và môi giới tài nguyên phía người dùng, nghĩa là cố gắng mang lại nhiều lợi ích nhất cho người dùng từ chi phí mà họ phải bỏ ra.

- Bộ lập lịch để thiết kế cho từng ứng dụng hay tổng quát hơn là từng lớp ứng dụng: các mô hình ứng dụng trong môi trường tính toán lưới đa dạng hơn rất nhiều so với các mô hình tính toán truyền thống. Ta lấy một ví dụ cụ thể và điển hình. Trong môi trường tính toán hiệu năng cao, cụ thể là mô hình tính toán bó (cluster) đang được sử dụng phổ biến trên thế giới hiện nay, các ứng dụng được bó gọn trong mô hình đơn chương trình đa dữ liệu (Single Instruction Multiple Data – SIMD), và hệ lập lịch và quản lý tài nguyên của Cluster, lấy một ví dụ điển hình là PBS, khi lập lịch cho các ứng dụng của người dùng đơn thuần chỉ là quan tâm đến các yếu tố chung nhất về ứng dụng như thời gian chạy, số nút tính toán yêu cầu... mà không phải quan tâm đến các chi tiết cụ thể của ứng dụng (vì chỉ có một kiểu ứng dụng). Trong khi đó mô hình ứng dụng trong tính toán lưới đa dạng hơn, có thể là ứng dụng dạng tuần tự, có thể là ứng dụng song song như kiểu cluster, cũng có thể là các ứng dụng dạng nghiên cứu tham số (Parameter Sweep) trong đó các công việc độc lập với nhau hoặc là ứng dụng theo kiểu luồng công việc (Workflow) trong đó các tác vụ có quan hệ thứ tự với nhau. Như vậy một bộ lập lịch khi tối ưu hóa việc thực thi của các ứng dụng nhất thiết phải quan tâm đến các chi tiết cụ thể của ứng dụng để có các chiến lược lập lịch phù hợp.

Sau đây là chi tiết các giai đoạn để lập lịch cho các ứng dụng lưới. Các giai đoạn được đề xuất ra ở đây là đủ tổng quát cho mọi mô hình ứng dụng lưới và mọi phần mềm nền tảng dùng để xây dựng lưới.



Hình 7-2: Các bước lập lịch cho một ứng dụng lưới

- Giai đoạn 1: khai phá tài nguyên

Giai đoạn đầu tiên trong mọi công việc liên quan đến lập lịch là xác định xem tài nguyên nào khả dụng đối với người dùng hiện tại. Pha khai phá tài nguyên liên quan đến việc lựa chọn một tập các tài nguyên để xem xét chi tiết hơn trong giai đoạn 2. Ở thời điểm khởi đầu giai đoạn 1 tập các tài nguyên tiềm năng là một tập rỗng. Đến cuối giai đoạn này, tập các tài nguyên tiềm năng sẽ chứa các tài nguyên thỏa mãn những yêu cầu tối thiểu. Giai đoạn khai phá tài nguyên được thực hiện theo ba bước:

+ Bước 1: Lọc phân quyền người dùng

Bước đầu tiên trong việc khai phá tài nguyên là xác định tập tài nguyên mà người dùng có đủ thẩm quyền truy nhập tới. Xét từ khía cạnh này thì việc chạy ứng dụng trên lưới cũng không có gì khác so với việc đệ trình công việc từ xa đến một nút tính toán đơn lẻ: không có thẩm quyền thì người dùng sẽ không chạy được ứng dụng trên lưới. Ở cuối bước này người dùng sẽ có trong tay một tập các máy hoặc tài nguyên mà họ có quyền truy nhập. Tùy vào từng bộ công cụ nền tảng để xây dựng lưới mà cơ chế phân quyền truy nhập được cài đặt khác nhau nhưng phổ biến nhất là dưới dạng các danh sách điều khiển truy nhập (Access Control List – ACL).

- *Bước 2:* Để pha khai phá tài nguyên được tiến hành hiệu quả, người dùng phải có khả năng định ra một tập các yêu cầu tối thiểu để thực hiện công việc. Điều này sẽ giúp cho việc lọc các tài nguyên hiệu quả hơn nữa. Tập các yêu cầu để thực hiện công việc có thể rất rộng và khác nhau rất nhiều cho từng loại công việc yêu cầu. Nó có thể có các thông tin tĩnh như hệ điều hành hay kiến trúc phần cứng hoặc cũng có thể bao gồm các thông tin động như lượng bộ nhớ tối thiểu cần dùng, khả năng kết nối hay bộ nhớ tạm. Các thông tin này càng nhiều và càng chi tiết thì càng có nhiều khả năng tìm được tài nguyên phù hợp nhất.

Hiện tại người dùng xác định các thông tin về yêu cầu công việc này thông qua dòng lệnh hoặc các kịch bản đệ trình công việc ví dụ như trong PBS hoặc LSF. Trong môi trường lưới tình hình trở nên phức tạp hơn bởi yêu cầu của ứng dụng sẽ thay đổi tương ứng với hệ thống mà chúng sẽ được thực hiện. Ví dụ, tùy thuộc vào kiến trúc máy tính và các thuật toán mà lượng bộ nhớ yêu cầu sẽ thay đổi cũng như các thư viện cần cài đặt hoặc lượng đĩa trống.

+ Bước 3: Lọc những yêu cầu tối thiểu

Cho một tập các tài nguyên mà người dùng có quyền truy cập và ít nhất là một tập nhỏ các yêu cầu của ứng dụng, bước 3 trong giai đoạn khai phá tài nguyên này sẽ lọc bỏ tất cả những tài nguyên không đáp ứng được những yêu cầu tối thiểu. Đến cuối bước này người sử dụng sẽ có trong tay một tập các tài nguyên để tìm hiểu chi tiết hơn ở giai đoạn sau.

Các hệ thống thông tin dịch vụ lưới hiện nay được thiết lập để chứa cả các thông tin tĩnh và động. Rất nhiều trong số chúng lưu trữ dữ liệu tại vùng đệm kèm theo một giá trị thời gian tồn tại (time-to-live) cho phép chúng đáp ứng nhanh hơn với những dữ liệu có thời gian tồn tại lâu bao gồm những thông tin cơ sở về tài nguyên như hệ điều hành, các phần mềm sẵn có và cấu hình phần cứng. Do các tài nguyên nằm phân tán và việc thu thập dữ liệu để đưa ra quyết định lập lịch có thể rất lâu nên bước này thường dùng những thông tin cơ bản và thường là tĩnh để đánh giá xem một tài nguyên nào đó có đáp ứng được những yêu cầu cơ bản hay không. Người dùng đơn giản là duyệt qua danh sách các tài nguyên và loại bỏ những tài nguyên không đáp ứng được các yêu cầu của công việc. Ví dụ: một phần mềm tính toán viết bằng Java yêu cầu phải có máy ảo Java để chạy được.

- Giai đoạn 2: Lựa chọn tài nguyên

Cho một tập tài nguyên khả dụng (thường là một tập con của tập tất cả các tài nguyên) đáp ứng được những yêu cầu cơ bản của công việc đề ra. Giai đoạn chọn lựa thường được tiến hành theo 2 bước: lấy thông tin chi tiết và đưa ra quyết định.

+ Bước 4: thu thập thông tin động

Để có thể đưa ra quyết định hợp lý nhất khi gán công việc cho các tài nguyên thì hệ lập lịch phải có được thông tin động về tài nguyên. Do thông tin này có thể thay đổi tùy thuộc vào ứng dụng đang được lập lịch và tài nguyên đang được kiểm tra nên không thể có giải pháp tổng quát nào cho vấn đề này. Bước thu thập thông tin động bao gồm việc xác định xem thông tin nào sẵn có và người dùng có thể truy nhập đến nó như thế nào. Thông tin sẵn có phụ thuộc vào từng tài nguyên và thông thường người dùng sẽ có 2 nguồn thông tin chính là các bộ thông tin tài nguyên lưới (Grid Information Service -GIS) và bộ lập lịch tài nguyên địa phương. Một vấn đề khá quan trọng khi tương tác với các tài nguyên đa miền quản trị đó là các chính sách quản lý tài nguyên địa phương. Một chính sách phổ biến là các nơi cung cấp tài nguyên sẽ hạn chế số phần trăm tài nguyên (dưới dạng dung lượng, tải hay một số tham

số khác) mà họ có thể cung cấp cho lưới. Những chi tiết này cũng phải được cân nhắc trong quá trình thu thập thông tin động.

+ Bước 5: Lựa chọn tài nguyên

Sau khi đã có đầy đủ thông tin về tài nguyên người dùng sẽ lựa chọn những tài nguyên phù hợp nhất cho yêu cầu và mục đích của họ. Bước này thường do bộ lập lịch và quản lý tài nguyên thay mặt người dùng đảm nhận tự động. Bước lựa chọn tài nguyên thường liên quan đến việc giải một bài toán tối ưu tổ hợp – bài toán lập lịch sẽ được trình bày kỹ hơn ở phần sau.

- Giai đoạn 3: Thực thi công việc

Giai đoạn này liên quan đến một loạt các bước, một số trong đó đã được chuẩn hóa.

+ Bước 6: Đặt trước tài nguyên (tùy chọn)

Để có thể sử dụng tốt nhất một hệ thống nào đó, một phần hoặc toàn bộ tài nguyên phải được đặt trước. Tùy thuộc vào từng tài nguyên mà việc đặt trước có thể dễ hay khó, thậm chí phải nhờ cả vào các tác động kỹ thuật của con người. Thêm vào đó, việc đặt chỗ trước có thể bị quá hạn và có thể phải trả phí cho việc đặt chỗ.

+ Bước 7: Đệ trình công việc

Sau khi đã chọn được tài nguyên ứng dụng cần phải được đệ trình lên tài nguyên đó để thực hiện. Việc thực hiện công việc có thể đơn giản giống như việc chạy một dòng lệnh đơn hoặc cũng có thể phức tạp như chạy một dãy các kịch bản và có thể còn cần phải có nhiều bước chuẩn bị.

Trong các hệ thống lưới việc đệ trình công việc tưởng như đơn giản có thể bị phức tạp hóa do thiếu các chuẩn. Một số hệ thống như Globus GRAM bao bên trên các bộ lập lịch địa phương tuy nhiên lại phụ thuộc quá nhiều vào các tham số địa phương. Hiện tại trên diễn đàn lưới toàn cầu (GGF – Global Grid Forum) cũng đã có các nỗ lực nhằm giải quyết vấn đề này.

+ Bước 8: Các công việc chuẩn bị

Trong bước này phía người dùng sẽ làm các công việc cần thiết để ứng dụng có thể chạy được. Ví dụ: dùng GridFTP để chuyển các file dữ liệu cần thiết đến địa điểm nơi công việc sẽ chạy.

+ Bước 9: Theo dõi tiến độ

Tùy thuộc vào ứng dụng và thời gian chạy của nó mà người dùng có thể muốn theo dõi tiến độ và có thể sẽ thay đổi ý định của họ về việc công việc sẽ được thực hiện ở đâu và như thế nào.

Trước đây những theo dõi như thế thường được thực hiện bằng cách liên tục truy vấn tài nguyên về thông tin hiện tại của nó tuy nhiên hiện nay có nhiều lựa chọn hơn. Tài nguyên cũng có thể chủ động thông báo về tình trạng hiện tại của nó. Nếu một công việc không thực hiện đủ tiến độ, nó có thể phải được lập lịch lại để được thực hiện trên tài nguyên khác. So

với các hệ thống đơn việc lập lịch lại trong lưới phức tạp hơn rất nhiều vì thiếu các cơ chế quản lý. Có thể có các công việc khác cũng được đệ trình để chạy trên tài nguyên đó và công việc hiện tại bị ngắt. Điều này nói chung nằm ngoài tầm kiểm soát của bộ lập lịch lưới. Để giải quyết tình trạng này phải có các giao thức để đảm bảo chất lượng dịch vụ giữa người dùng và nhà cung cấp dịch vụ.

+ Bước 10: hoàn thành công việc

Khi công việc kết thúc thì cần phải báo cho người sử dụng bằng một hình thức nào đó. Ví dụ một số hệ thống tính toán khi đệ trình công việc có thêm tham số về địa chỉ email để thông báo. Khi người dùng sử dụng bộ lập lịch và môi giới tài nguyên thì bộ lập lịch sẽ thường truy vấn để biết tình trạng công việc từ lúc bắt đầu cho đến khi kết thúc.

+ Bước 11: dọn dẹp và kết thúc

Sau khi một công việc đã được chạy người dùng sẽ phải chuyển các file kết quả về, xóa các file tạm. Người sử dụng có thể làm công việc này một cách thủ công hoặc mô tả nó trong kịch bản đệ trình của họ.

7.2.3. Các thuật toán lập lịch

Trong phần này ta sẽ đi vào nghiên cứu phần cốt lõi của các bộ lập lịch và môi giới tài nguyên, đó là các thuật toán lập lịch. Chính các thuật toán này sẽ quyết định hành vi của các bộ lập lịch và môi giới tài nguyên, quyết định độ ổn định cũng như tính hiệu quả và kinh tế của chúng. Như đã trình bày ở phần trên, bộ lập lịch ở đây hướng đến người dùng và được thiết kế tối ưu cho từng mô hình ứng dụng chạy trên môi trường lưới nên trước hết ta tìm hiểu về mô hình ứng dụng trong BK Grid 2005.

7.2.3.1. Mô hình ứng dụng

- Giới thiệu về Weka

Weka là một dự án xây dựng các công cụ phần mềm phục vụ cho lĩnh vực khai phá dữ liệu (Data mining) đã được phát triển từ khá lâu bởi một đội ngũ đông đảo các nhà khoa học trên toàn thế giới. Các sản phẩm của Weka bao gồm:

Các thư viện tính toán trong lĩnh vực data mining

Các chương trình data mining trên máy đơn

Các chương trình data mining song song trên nhiều máy (Weka parallel)

Các chương trình data mining trên môi trường lưới (Weka Grid) mới được nghiên cứu và phát triển gần đây.

Các chương trình Weka trên máy đơn sử dụng các thuật toán tuần tự để xử lý dữ liệu. Weka Parallel sử dụng cùng một lúc nhiều máy tính để xử lý song song dữ liệu. Trong Weka Parallel đã có sẵn các chức năng lập lịch, chọn tài nguyên... tự động. Việc của người sử dụng còn lại chỉ là cấu hình và cài đặt sau đó Weka Parallel sẽ tự động điều khiển các máy tính này hoạt động song song như một Cluster. Tuy Weka Parallel hoạt động như một

cluster nhưng nó hoàn toàn không sử dụng bất kỳ một thư viện lập trình song song nào (ví dụ:MPI, PVM...) và cũng không sử dụng phần mềm quản lý tài nguyên và phân tải nào (ví dụ: PBS) mà tất cả được thực hiện hoàn toàn bằng các thư viện lập trình của Java.

Gần đây có một nhóm nghiên cứu ở New Zealand tuyên bố đã phát triển các chương trình Weka cho môi trường lưới. Gói phần mềm này có một số dac diem nhu sau:

+ Weka Grid có giao diện rất giống với Weka Parallel.

+ Weka Grid không sử dụng bất cứ framework phát triển ứng dụng lưới nào (ví dụ: Globus), tất cả vẫn hoàn toàn được thực hiện bằng các thư viện lập trình của Java. Nhóm phát triển tuyên bố là trong thời gian tới họ sẽ tiếp tục phát triển để Weka Grid tuân theo chuẩn dịch vụ lưới OGSA.

+ Weka Grid được phát triển theo hướng mổ xẻ các thuật toán trong Weka sau đó viết lại theo kiểu song song hóa cho môi trường lưới.

- Hướng phát triển ứng dụng Weka trong BKGrid 2005

Mục đích phát triển ứng dụng Weka trong BK Grid 2005 cũng là "lưới hóa" ứng dụng Weka tuy nhiên không giống như cách làm của Weka Grid, thể hiện ở chỗ:

"Lưới hóa" ở đây hiểu theo nghĩa tạo giao diện lưới cho các chức năng của Weka, giúp cho người sử dụng lưới có thể dễ dàng truy nhập và sử dụng các dịch vụ tính toán của Weka trên môi trường lưới chứ không phải đi sâu vào phân tích mổ xẻ rồi song song hóa lại các thuật toán của Weka trên lưới. Lý do của cách làm này là:

+ Hạn chế về kiến thức chuyên ngành: các thuật toán của Weka đều rất sâu về lĩnh vực data mining, do vậy muốn nắm vững các thuật toán này (ở dạng tuần tự) sau đó song song hóa chúng là vấn đề rất khó khăn.

+ Không sát với lĩnh vực nghiên cứu: Mục tiêu của nhóm là nghiên cứu công nghệ tính toán lưới và các vấn đề làm cơ sở nền tảng cho việc phát triển ứng dụng chứ không phải là bản thân việc phát triển ứng dụng. Các ứng dụng triển khai chỉ mang tính chất minh họa. Muốn phát triển ứng dụng lưới cho lĩnh vực chuyên môn nào đó đều đòi hỏi có sự hợp tác chặt chẽ của các chuyên gia trong lĩnh vực nghiên cứu đó, do vậy có thể việc phát triển thêm nhiều ứng dụng mang tính chuyên môn và tính thực tiễn cao có thể sẽ là đề tài phát triển cho các năm sau khi lưới của ta đã phát triển về phạm vi và kêu gọi được nhiều tổ chức cùng tham gia với các bài toán thực tế của họ (ví dụ: Viện Vật Lý Kỹ Thuật với các bài toán về công nghệ nano)

+ Không tận dụng được các công cụ sẵn có của Weka. Một số thuật toán của Weka đã được song song hóa bởi Weka Parallel.

Như vậy phương hướng phát triển của nhóm ứng dụng là nghiên cứu việc tận dụng tối đa các chức năng sẵn có của Weka, bao chúng bằng các dịch vụ lưới giúp cho người dùng có thể dễ dàng sử dụng từ xa.

Người dùng sẽ được hưởng lợi từ công nghệ tính toán lưới thể hiện ở những điểm sau:

+ Cho phép chạy ứng dụng từ xa: nếu tài nguyên địa phương không đủ đáp ứng yêu cầu xử lý có thể thuê thêm các nguồn tài nguyên từ các tổ chức khác với giá cả phù hợp. Đặc biệt có thể thuê thêm cả một cụm cluster với sức mạnh tính toán rất lớn (Trên cluster này có Weka Parallel được bao bởi một dịch vụ lưới)

+ Khả năng chạy song song các công việc khi người dùng có một số lượng lớn các công việc độc lập nhau. Vd :một số lượng lớn các file đầu vào độc lập cần Weka phân tích. Khi đó các file này sẽ được phân phối đến nhiều tài nguyên để xử lý song song.

+ Song song hóa các công việc có khả năng chia thành nhiều phần thực hiện độc lập.

Việc chọn các tài nguyên như thế nào cho phù hợp để đáp ứng được các yêu cầu của người dùng (về giá cả, thời gian thực hiện...) sẽ là nhiệm vụ của bộ lập lịch và bộ môi giới tài nguyên.

- Mô hình ứng dụng Weka trong BK Grid 2005

Ứng dụng Weka trong BK Grid 2005 gồm 3 phần:

+ Phần back-end là các dịch vụ tính toán đầu cuối. Phần này cài đặt các thuật toán của Weka được bao bởi các dịch vụ lưới. Các thuật toán được sử dụng có thể ở dạng tuần tự hoặc song song. Khi triển khai dưới dạng tuần tự thì tài nguyên của ta là một máy tính đơn.Ở dạng song song thì tài nguyên tính toán của ta là 1 cluster, khi đó dịch vụ lưới sẽ bao lên toàn bộ cluster này và được cài ở nút chủ. Dịch vụ lưới khi đó sẽ tương tác với Weka Parallel còn lại mọi công việc nội bộ trong cluster hoàn toàn do Weka Parallel đảm nhiệm.

+ Phần middle-tier chịu trách nhiệm phối hợp các dịch vụ tính toán Weka đầu cuối để tạo ra ứng dụng Weka hoàn chỉnh. Các chức năng chính của phần này là môi giới tài nguyên, lập lịch và theo dõi hoạt động của ứng dụng một cách tổng thể.

+ Phần front-end nằm trên portal chịu trách nhiệm giao tiếp với người dùng. Phần này chịu trách nhiệm nhận các yêu cầu từ người dùng, chuyển các yêu cầu này đến tầng middle-tier của ứng dụng, nhận kết quả trả về từ tầng này và cuối cùng là hiện thị kết quả cho người dùng.

- Các kịch bản sử dụng Weka trong BKGrid 2005

Khi người dùng sử dụng ứng dụng Weka họ sẽ được cấp phát riêng một thể hiện (instance) của tầng middle-tier (có thể coi là một instance của bộ lập lịch và môi giới tài nguyên). Bộ lập lịch này sẽ nhận các yêu cầu của người dùng và chịu trách nhiệm thực hiện ứng dụng theo các tiêu chí tối ưu do duy nhất người dùng đó đặt ra. Các kịch bản sử dụng bao gồm:

+ Người dùng chỉ có một file đầu vào duy nhất cần phân tích: khi này nhiệm vụ của bộ lập lịch là chọn 1 tài nguyên phù hợp nhất để đáp ứng yêu cầu này dựa vào các thông tin về tình trạng hiện tại của lưới và các yêu cầu mà dùng đặt ra. Ví dụ: giới hạn về kinh phí.

+ Người dùng có một tập nhiều file đầu vào: Nhiệm vụ của bộ lập lịch lúc này trở nên phức tạp hơn rất nhiều do phải dùng song song cùng lúc nhiều tài nguyên để đáp ứng yêu cầu của người sử dụng. Điểm khác biệt so với khi chạy các ứng dụng Weka thông thường thể hiện ở

chỗ người dùng có khả năng đặt thêm các giới hạn về thời gian hay kinh phí cho việc thực hiện.

7.2.3.2. Tư tưởng thiết kế chung của các thuật toán lập lịch

Việc đưa các tham số kinh tế vào hệ thống lập lịch có ảnh hưởng rất lớn đến việc lựa chọn các tài nguyên thỏa mãn yêu cầu của người sử dụng. Người sử dụng sẽ đưa ra ứng dụng cùng các yêu cầu của họ. Bộ lập lịch dựa trên sự ủy nhiệm của người dùng sẽ cố gắng hoàn thành công việc đã được giao với khoản thời gian và kinh phí đã được định trước. Để đi đến một quyết định lập lịch hệ thống lập lịch phải cân nhắc dựa trên rất nhiều yếu tố như:

- Kiến trúc tài nguyên và cấu hình của chúng.
- Khả năng của tài nguyên (ví dụ tốc độ xung, kích thước bộ nhớ...)
- Trạng thái tài nguyên (ví dụ tải CPU, dung lượng bộ nhớ còn trống, dung lượng đĩa còn trống...)
- Yêu cầu tài nguyên của ứng dụng.
- Tốc độ truy nhập
- Số nút tính toán còn rỗi (ví dụ trong các tài nguyên tính toán dạng cluster).
- Độ ưu tiên của người dùng.
- Dạng hàng đợi và độ dài hàng đợi
- Thông lượng đường truyền mạng và độ trễ (nếu các công việc có yêu cầu truyền thông).
- Độ tin cậy của tài nguyên và các liên kết.
- Sự lựa chọn của người sử dụng.
- Thời hạn phải hoàn thành ứng dụng.
- Khả năng thanh toán của người dùng cho việc sử dụng tài nguyên.
- Giá sử dụng tài nguyên, mức giá này có thể thay đổi với những thời điểm sử dụng khác nhau. Ví dụ vào những giờ cao điểm mức giá có thể cao hơn và vì vậy khuyến khích người dùng sử dụng tài nguyên vào những thời điểm khác khi tài nguyên rỗi.
- Những thông tin về lịch sử sử dụng tài nguyên và tốc độ hoàn thành công việc.

Những tham số kinh tế chính tác động đến quyết định của bộ lập lịch là:

- Giá tài nguyên (được người sở hữu đặt ra).
- Ngân sách của người dùng.
- Thời hạn phải hoàn thành ứng dụng.

Như đã trình bày ở trên, ứng dụng được triển khai trên BK Grid thuộc dạng ứng dụng nghiên cứu tham số bao gồm một tập lớn các công việc độc lập trên tập các dữ liệu độc lập. Mô hình này cũng giống như mô hình lập trình đơn chương trình đa dữ liệu (Single Program Multiple Data – SPMD). Việc lập lịch và điều phối hoạt động của các ứng dụng dạng này trên môi trường tính toán phân tán trên diện rộng tưởng chừng khá đơn giản tuy nhiên các vấn đề phức tạp sẽ nảy sinh khi người dùng đặt ra một số các tham số về chất lượng dịch vụ như thời hạn kết thúc ứng dụng và giới hạn về kinh phí. Những đảm bảo như thế về chất

lượng dịch vụ là rất khó thực hiện trong một môi trường như môi trường lưới khi mà các tài nguyên lưới phân tán, không đồng nhất, được sở hữu bởi nhiều tổ chức khác nhau với những chính sách và cơ chế tính giá khác nhau. Thêm vào đó các thuật toán lập lịch phải được thiết kế để thích ứng được với sự thay đổi tải và sự sẵn dùng của tài nguyên cùng lúc với việc phải đảm bảo được thời hạn và kinh phí.

Tư tưởng chủ đạo của việc thiết kế các thuật toán lập lịch cho lưới là: không sử dụng các thuật toán tối ưu truyền thống vẫn hay thường được sử dụng trong các bài toán lập lịch mà chủ yếu sử dụng các heuristic. Lý do của cách làm này ở chỗ:

Các thuật toán tối ưu tổ hợp cần phải có các thông tin chính xác để có thể làm việc hiệu quả tuy nhiên điều này rất khó đạt được trong môi trường lưới. Thêm vào đó có một số loại thông tin như thời gian chạy của các công việc hay tải của tài nguyên là không thể có được một cách chính xác mà chỉ là những con số ước đoán. Như vậy có thể nói lời giải đạt được có thể sẽ không có ý nghĩa lắm khi mà dữ liệu đầu vào có độ chính xác không cao. So sánh với các thuật toán lập lịch trên các hệ tính toán song song như cluster ta thấy rằng trên các hệ này thông tin được cung cấp một cách rất đầy đủ và đáng tin cậy, các tài nguyên được kiểm soát hoàn toàn (do chúng cùng thuộc một tổ chức) nên việc khống chế và đo tải khá dễ dàng. Do vậy thuật toán lập lịch trên các hệ này thường là các thuật toán tối ưu tổ hợp cổ điển như Tabu Search, Simulated Anealing hay là giải thuật di truyền (genetic algorithm)...

Tài nguyên trên lưới thường có những biến động lớn và liên tục xảy ra ngoài tầm kiểm soát. Ví dụ: một tài nguyên không thể truy cập được do lỗi của bản thân tài nguyên đó, do đường truyền mạng hoặc do chủ sở hữu tài nguyên đang cần phải sử dụng nó để thực hiện công việc nội bộ của tổ chức... Do vậy việc lập lịch lại sẽ phải tiến hành khá thường xuyên. Vì lý do đó chúng ta cần một thuật toán lập lịch hoạt động rất nhanh để đảm bảo tính hiệu quả của bộ lập lịch.

Phần tiếp theo đây sẽ trình bày một số thuật toán lập lịch mang tính kinh tế đã được nghiên cứu và phát triển cho môi trường lưới.

7.2.3.3. Thuật toán tối ưu hóa về thời gian

Thuật toán tối ưu về thời gian cố gắng hoàn thành công việc một cách nhanh nhất có thể với điều kiện giới hạn trong ngân sách hiện có. Sau đây là đoạn giả mã mô tả phần lõi của thuật toán:

- B1: Khai phá tài nguyên

Xác định các tài nguyên cùng khả năng của chúng bằng dịch vụ thông tin tài nguyên lưới GIS. Với mỗi tài nguyên, xác định giá và khả năng xử lý công việc ứng với mỗi đơn vị giá. Ví dụ: Mb/$

- B2: Lập lịch

Lặp lại các bước sau nếu còn các công việc chưa được xử lý với điều kiện thời gian và chi phí hiện tại chưa vượt quá giới hạn thời gian và ngân sách người dùng đặt ra. Việc lập lịch

lại được tiến hành theo định kỳ hoặc mỗi khi có một sự kiện nào đó xảy ra. Ví dụ: có công việc mới hoàn thành hoặc công việc bị lỗi...

a. Với mỗi tài nguyên: dự đoán tốc độ hoàn thành công việc hoặc dung lượng tài nguyên được chia sẻ thông qua đo đạc và ngoại suy có tính đến thời gian cần thiết để hoàn thành các công việc trước đó.

b. Nếu tài nguyên nào không thể xử lý hết khối lượng công việc được giao trước thời hạn hoàn thành thì chuyển bớt một khối lượng công việc thích hợp ra ngoài hàng đợi các công việc chưa được lập lịch. Điều này giúp cho việc cập nhật lịch trình thực hiện dựa vào những thông tin mới nhất về tài nguyên.

c. Lặp lại các bước sau cho mỗi công việc trong hàng đợi công việc chưa được lập lịch:

Chọn 1 công việc chưa được lập lịch từ hàng đợi.

Tạo 1 nhóm các tài nguyên chứa những tài nguyên có thể sử dụng được trong giới hạn ngân sách hiện có. Ví dụ: những tài nguyên mà giá nhỏ hơn hoặc bằng ngân sách còn lại cho mỗi công việc.

Với mỗi tài nguyên trong nhóm tài nguyên kể trên, tính toán và dự báo thời gian hoàn thành công việc có tính đến các công việc đã được gán trước đó, tốc độ hoàn thành công việc và tải tài nguyên hiện tại.

Sắp xếp các tài nguyên trong nhóm theo thứ tự tăng dần của thời gian hoàn thành.

Gán công việc cho tài nguyên đầu tiên trong nhóm và xóa nó khỏi hàng đợi công việc nếu thời gian hoàn thành tiên đoán sớm hơn giới hạn về thời gian mà người dùng đã đặt ra.

- B3. Phân phối công việc đến các tài nguyên theo kế hoạch đã lập ra.

Một số nhận xét về thuật toán:

+ Thuật toán tối ưu hóa về thời gian cố gắng hoàn thành công việc trong thời hạn sớm nhất có thể với chi phí nằm trong giới hạn cho phép đã được ấn định bởi người dùng. Bộ lập lịch sẽ cố gắng dùng những tài nguyên nhanh nhất có thể để đạt được mục đích tối ưu về thời gian. Điều đó có nghĩa là với ngân sách càng lớn thì khả năng người dùng đạt được độ tối ưu về thời gian càng cao.

+ Nếu người dùng có một lượng ngân sách không hạn chế thì khi đó chúng ta trở về với một bài toán tối ưu cổ điển: cho một tập M tài nguyên và một tập N công việc, tìm cách thực hiện nhanh nhất. Cách giải quyết bài toán này theo phương pháp cổ điển là dùng một trong các thuật toán tối ưu tổ hợp đã được nghiên cứu khá kỹ như Tabu Search, Simulated Aneaning, giải thuật di truyền... Tuy nhiên điều này trong thực tế hiếm khi xảy ra vì người dùng ít khi đạt được tình trạng dư giả về tài chính như vậy. Mặt khác như đã phân tích ở trên do tính chất bất định của môi trường lưới mà việc áp dụng các thuật toán tối ưu cổ điển sẽ trở nên kém hiệu quả.

+ Độ phức tạp tính toán:

Giả sử số công việc đầu vào là M và số tài nguyên là N. Thông thường ta có M>>N. Thân thuật toán có một vòng lặp chính duyệt qua tất cả các tài nguyên và các công việc:

Vòng lặp ngoài: duyệt tuần từ qua danh sách công việc độ dài M.

Vòng lặp trong: duyệt tuần tự qua từng tài nguyên để thực hiện một số thao tác tính toán sau đó sắp xếp tập tài nguyên. Thời gian tính toán chủ yếu tập trung vào phần xắp xếp. Nếu áp dụng giải thuật Quicksort thì độ phức tạp tính toán là O(NlogN)

Như vậy độ phức tạp tính toán của giải thuật là O(MNlogN)

7.2.3.4. Thuật toán tối ưu hóa về chi phí thực hiện.

Thuật toán tối ưu hóa chi phí thực hiện cố gắng hoàn thành công việc trong giới hạn thời gian cho phép với mức chi phí thấp nhất có thể. Sau đây là phần mô tả thuật toán:

- B1: Khai phá tài nguyên

+ Xác định các tài nguyên cùng khả năng của chúng bằng dịch vụ thông tin tài nguyên lưới GIS. Với mỗi tài nguyên, xác định giá và khả năng xử lý công việc ứng với mỗi đơn vị giá. Ví dụ: Mb/$

+ Sắp xếp tài nguyên theo thứ tự tăng dần về giá

- B2: Lập lịch

Lặp lại các bước sau nếu còn các công việc chưa được xử lý với điều kiện thời gian và chi phí hiện tại chưa vượt quá giới hạn thời gian và ngân sách người dùng đặt ra. Việc lập lịch lại được tiến hành theo định kỳ hoặc mỗi khi có một sự kiện nào đó xảy ra. Ví dụ: có công việc mới hoàn thành hoặc công việc bị lỗi...

a. Với mỗi tài nguyên: dự đoán tốc độ hoàn thành công việc hoặc dung lượng tài nguyên được chia sẻ thông qua đo đạc và ngoại suy. Tính toán khối lượng công việc mà tài nguyên có thể hoàn thành trước thời hạn cho phép.

b. Với mỗi tài nguyên theo thứ tự đã xắp xếp:

Nếu khối lượng công việc hiện thời được gán cho tài nguyên nhỏ hơn khối lượng công việc đã được dự đoán mà nó có thể hoàn thành trước thời hạn thì gán thêm công việc cho nó từ hàng đợi công việc chưa được lập lịch. Chỉ gán công việc khi còn đủ ngân sách.

Ngược lại, nếu một tài nguyên không thể xử lý hết khối lượng công việc đã được gán cho nó trước thời hạn hoàn thành thì chuyển những công việc này về hàng đợi các công việc chưa được lập lịch.

- B3. Phân phối công việc đến các tài nguyên theo kế hoạch đã lập ra.

Một số nhận xét về thuật toán:

+ Thuật toán tối ưu hóa về chi phí cố gắng sử dụng ít chi phí nhất có thể trong giới hạn thời gian đã đặt ra. Giả sử nếu người dùng không hề có một giới hạn nào về mặt thời gian, nghĩa là các công việc có thể được thực hiện trong bao lâu tùy ý thì giải pháp là rất rõ ràng: thực hiện tuần tự mọi công việc trên tài nguyên rẻ nhất trong lưới. Tuy nhiên cũng như ở trên trong thực tế điều này thường không xảy ra. Người dùng thường muốn công việc của họ

được hoàn thành trong một giới hạn nào đó về mặt thời gian và vì thế cũng đặt ra thêm ràng buộc cho bộ lập lịch.

+ Khi người dùng càng nới lỏng yêu cầu về thời gian thực hiện công việc, nghĩa là thời gian thực hiện càng lớn thì khả năng họ đạt được sự tối ưu về mặt chi phí càng cao bởi khi đó bộ lập lịch có thể sử dụng những tài nguyên giá rẻ (và tất nhiên là tốc độ chậm) để thực hiện công việc mà vẫn nằm trong giới hạn thời gian cho phép.

+ Độ phức tạp tính toán:

Giả sử số công việc là M và số tài nguyên là N. Phần chính của thuật toán gồm 2 vòng lặp lồng nhau:

Duyệt qua danh sách các tài nguyên đã được xắp xếp độ dài N.

Duyệt qua dánh sách các công việc để tìm số lượng công việc phù hợp gán cho tài nguyên

Như vậy độ phức tạp của thuật toán này là O(MN).

7.2.3.5. Thuật toán tối ưu hóa về thời gian – chi phí thực hiện

Thuật toán tối ưu hóa về thời gian – chi phí thực hiện là một mở rộng của thuật toán tối ưu hóa chi phí để tối ưu hóa về thời gian mà không phải trả thêm chi phí . Điều này được thực hiện bằng cách áp dụng thuật toán tối ưu hóa về thời gian để lập lịch cho các công việc trên những tài nguyên có cùng giá. Sau đây là mô tả thuật toán:

- B1: Khai phá tài nguyên

Xác định các tài nguyên cùng khả năng của chúng bằng dịch vụ thông tin tài nguyên lưới GIS. Với mỗi tài nguyên, xác định giá và khả năng xử lý công việc ứng với mỗi đơn vị giá. Ví dụ: Mb/$

- B2: Lập lịch

Lặp lại các bước sau nếu còn các công việc chưa được xử lý với điều kiện thời gian và chi phí hiện tại chưa vượt quá giới hạn thời gian và ngân sách người dùng đặt ra. Việc lập lịch lại được tiến hành theo định kỳ hoặc mỗi khi có một sự kiện nào đó xảy ra. Ví dụ: có công việc mới hoàn thành hoặc công việc bị lỗi...

a. Với mỗi tài nguyên: dự đoán tốc độ hoàn thành công việc hoặc dung lượng tài nguyên được chia sẻ thông qua đo đạc và ngoại suy có tính đến thời gian cần thiết để xử lý các công việc trước đó.

b. Sắp xếp các tài nguyên theo thứ tự tăng dần về giá. Nếu hai tài nguyên có cùng giá thì sắp xếp sao cho các tài nguyên mạnh hơn sẽ được ưu tiên hơn trong quá trình chọn lựa. Ví dụ: theo thứ tự giảm dần về tốc độ thực hiện công việc hoặc nếu là lần đầu thực hiện thì dựa trên tốc độ lý thuyết của tài nguyên. Ví dụ: MIPS

c. Tạo danh sách các nhóm tài nguyên có cùng giá.

d. Sắp xếp các nhóm tài nguyên theo thứ tự tăng dần về giá.

e. Nếu tài nguyên nào đã được gán một số công việc cho nó trong lần lập lịch trước mà chưa được chuyển đến tài nguyên để xử lý và có những biến động về tải của tài nguyên mà

nó không thể xử lý hết khối lượng công việc được giao trước thời hạn hoàn thành thì chuyển bớt một khối lượng công việc thích hợp ra ngoài hàng đợi các công việc chưa được lập lịch. Điều này giúp cho việc cập nhật lịch trình thực hiện dựa vào những thông tin mới nhất về tài nguyên.

f. Lặp lại các bước sau với mỗi nhóm tài nguyên khi còn công việc chưa được lập lịch:

Lặp lại các bước sau cho mỗi công việc trong hàng đợi tùy thuộc và o giá xử lý và ngân sách hiện tại:

Chọn một công việc từ danh sách công việc chưa được lập lịch

Với mỗi tài nguyên dự đoán thời hạn hoàn thành công việc có tính đến những công việc đã được gán trước đó và tốc độ hoàn thành công việc.

Sắp xếp các tài nguyên theo thứ tự tăng dần của thời gian thực hiện

Gán công việc cho tài nguyên đầu tiên và loại bỏ nó ra khỏi danh sách công việc chưa lập lịch nếu thời gian dự đoán hoàn thành nhỏ hơn giới hạn thời gian.

- B3. Phân phối công việc đến các tài nguyên theo kế hoạch đã lập ra.

Độ phức tạp tính toán: vòng lặp chính của thuật toán này tương tự như thuật toán tối ưu hóa về thời gian. Độ phức tạp tính toán là O(MNlogN)

7.2.4. Một số vấn đề khó trong lập lịch

7.2.4.1. Vấn đề xác định độ dài của các công việc

Như ta đã thấy trong các thuật toán trình bày ở trên để bộ lập lịch hoạt động động được hiệu quả thì cần thiết phải ước lượng được tương đối chính xác thời gian cần thiết để thực hiện một công việc nào đó. Tuy nhiên điều này rất khó thực hiện và hiện trên thế giới cũng có rất ít công trình đề cập đến vấn đề này. Công cụ chủ yếu là các đánh giá theo kiểu heuristic hoặc là các đánh giá dựa vào lich sử thực hiện các công việc được tiến hành trước đó. Việc đánh giá độ dài công việc phải dựa vào kiểu thuật toán cụ thể dùng để xử lý và loại tài nguyên cụ thể dùng để xử lý. Lấy một ví dụ cụ thể: Bài toán yêu cầu sắp xếp một mảng số độ dài N.

- Ta biết rằng để sắp xếp một mảng số có thể dùng khá nhiều thuật toán khác nhau với độ phức tạp tính toán khác nhau. Ví dụ nếu dùng Quick Sort thì độ phức tạp tính toán trung bình là O(NlogN) còn nếu dùng Bubble Sort thì độ phức tạp tính toán lại là $O(N^2)$. Đây có thể là những thông tin quan trọng để ta đưa ra những đánh giá về độ dài 1 công việc khi đã biết độ dài của một công việc bất kỳ.

- Nếu sử dụng một tài nguyên tính toán song song để thực hiện công việc sẽ khác với thực hiện trên tài nguyên tính toán đơn lẻ vì độ phức tạp tính toán của các thuật toán sắp xếp song song sẽ khác với độ phức tạp tính toán của các thuật toán sắp xếp tuần tự.

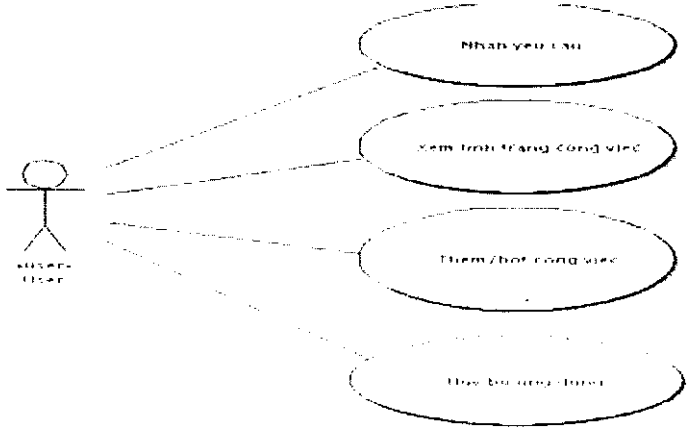
7.2.4.2. Vấn đề tiên đoán tải của tài nguyên và chất lượng thực hiện

Ngoài thông tin về độ dài của công việc, thông tin về tải của tài nguyên cũng là một yếu tố quan trọng để ước lượng thời gian thực hiện công việc. Việc tiên đoán tải của tài nguyên chính xác giúp cho bộ lập lịch đánh giá được tài nguyên nào có khả năng xử lý lượng công việc là bao nhiêu trong khoảng thời gian đã đặt ra. Điều này là cực kỳ quan trọng khi bộ lập lịch phải chịu áp lực về thời gian thực hiện. Tuy nhiên tải của tài nguyên lại luôn luôn biến động do vậy các thông tin này chỉ có ý nghĩa trong một khoảng thời gian nhất định và phải được thường xuyên cập nhật. Mức độ cập nhật thông tin sẽ tuỳ thuộc vào tính chất hoạt động của từng tài nguyên và ta mong đợi rằng thông tin là đủ ổn định trong khoảng thời gian lập lịch cho một công việc nào đó. Một số loại tài nguyên nhỏ như máy tính PC của một người dùng, trạm làm việc... có mức độ ổn định tải không cao. Với những tài nguyên này chúng ta có thể xác định một giá trị tải trung bình cho một khoảng thời gian đủ nhỏ. Một số loại tài nguyên tính toán lớn như các máy tính lớn, các cluster... lại có độ ổn định tải tương đối cao. Lý do đơn giản là các ứng dụng chạy trên chúng thường là các ứng dụng tính toán khoa học hoặc các ứng dụng tương tự. Đặc điểm của các ứng dụng này là thời gian chạy lớn, từ vài giờ đến vài ngày, và yêu cầu sức mạnh tính toán tương đối ổn định, do vậy mà có thể xác định được tải trên chúng trong một khoảng thời gian dài hơn. Từ thông tin này ta có thể ước lượng được số phần trăm tải CPU mà ứng dụng của ta có thể được cấp và thời gian mà ứng dụng của ta sẽ chạy.

7.3. Xây dựng bộ lập lịch cho ứng dụng lưới trong BK Grid 2005

7.3.1. Các yêu cầu đối với bộ lập lịch và môi giới tài nguyên

Trong phần này ta sẽ nêu nên các đặc tả chức năng đối với bộ lập lịch và môi giới tài nguyên dùng ngôn ngữ mô hình hóa UML. Nhìn từ phía người sử dụng, bộ lập lịch và môi giới tài nguyên phải có các chức năng chính sau:



Hình 7-3: Use case của bộ lập lịch

- Tiếp nhận các yêu cầu từ người sử dụng: tập công việc, các đặc tả về chi phí và thời gian thực hiện, các yêu cầu về tối ưu hóa quá trình thực hiện như giảm thiểu thời gian, chi phí thực hiện...
- Xem chi tiết các thông tin về quá trình thực hiện công việc: tình trạng hiện thời của các công việc và các tài nguyên đang được sử dụng, thông tin về thời gian thực hiện và chi phí.
- Ghi lại chi tiết các thông tin về quá trình thực hiện ứng dụng vào cơ sở dữ liệu để làm cơ sở thanh toán cho người dùng sau này.
- Thêm bớt công việc, hủy bỏ việc thực hiện ứng dụng...

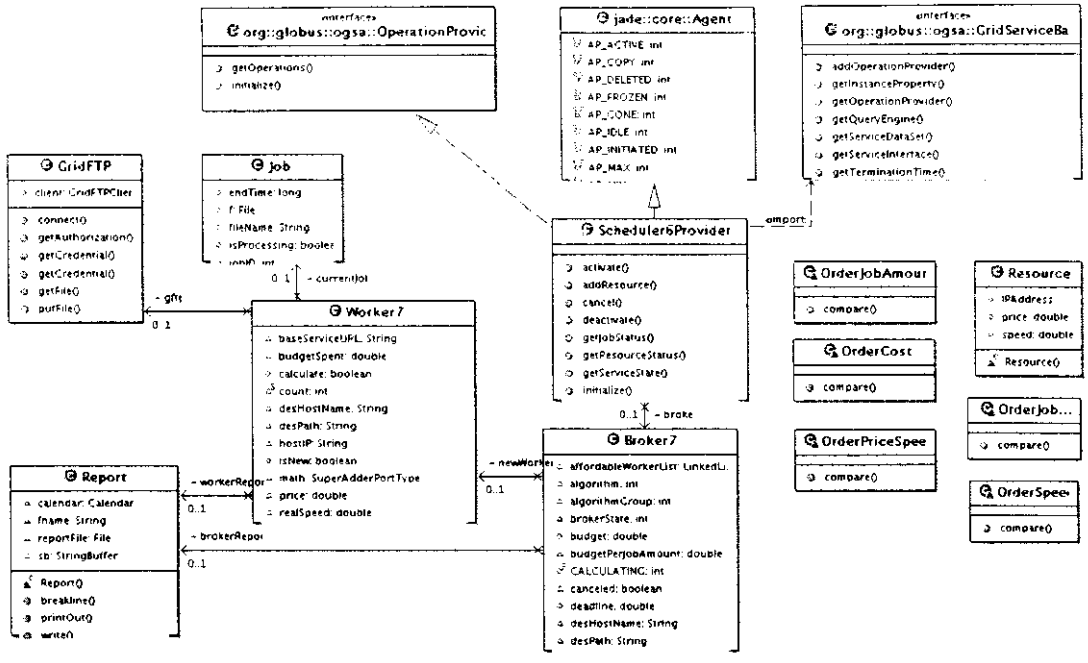
Bộ lập lịch phải được thiết kế để đảm bảo hoạt động tin cậy và hiệu quả trong môi trường lưới, trong đó yếu tố tin cậy phải được đặt lên hàng đầu. Do đó trong bộ lập lịch phải có tích hợp sẵn các chức năng chống chịu lỗi và lập lịch lại khi cần thiết.

7.3.2. Thiết kế bộ lập lịch

Bộ lập lịch cho BK Grid 2005 được thiết kế phù hợp với toàn bộ tổng thể của kiến trúc lưới tức là cũng được thiết kế dưới dạng một dịch vụ lưới. Dịch vụ này được triển khai theo mô hình xưởng chế tác (Factory). Mỗi người dùng khi sử dụng ứng dụng sẽ được khởi tạo một bộ lập lịch mới thay mặt người dùng điều khiển toàn bộ quá trình thực hiện ứng dụng. Các bộ lập lịch này hoạt động tương đối độc lập với nhau và tự do trong việc cạnh tranh sử dụng tài nguyên.

Để cho phép người dùng theo dõi được theo thời gian thực toàn bộ quá trình thực hiện ứng dụng bộ lập lịch sẽ được thiết kế theo mô hình lập trình dịch vụ đa luồng trong đó các luồng tính toán sẽ được tách riêng ra khỏi luồng chính của dịch vụ. Khi đó dịch vụ sẽ luôn có khả năng đáp ứng tức thì các yêu cầu cung cấp thông tin của người dùng.

Cuối cùng để tương tác với hệ thống đa Agent một cách hiệu quả và thuận tiện nhất bản thân dịch vụ lập lịch cũng được thiết kế để trở thành một dịch vụ lai Agent.



Hình 7-4: Các lớp chính của bộ lập lịch

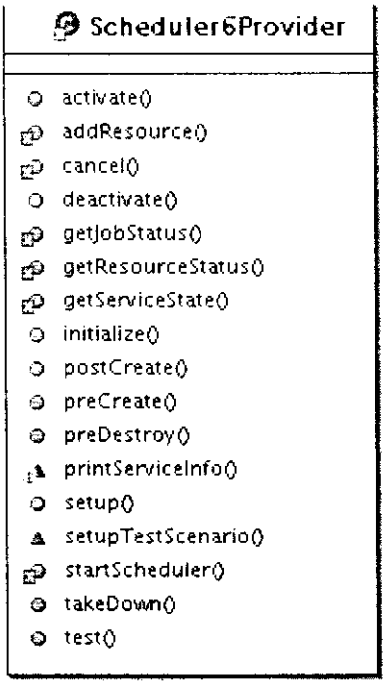
Các lớp chính của bộ lập lịch:

- *Lớp SchedulerProvider*: lớp này chứa luồng chính của dịch vụ làm nhiệm vụ tương tác với người dùng. Lớp này nhận các thông tin đầu vào về tập công việc cần xử lý, các thông tin về giới hạn ngân sách, thời gian của người dùng cũng như các chiến lược tối ưu sau đó chuyển cho lớp Broker để xử lý. Sau khi khởi tạo lớp Broker luồng này của dịch vụ ở trạng thái chờ và sẵn sàng cung cấp các thông tin chi tiết về quá trình thực hiện khi người dùng yêu cầu.
- *Lớp Broker:* lớp này chứa luồng tính toán chính của dịch vụ chịu trách nhiệm phối hợp hoạt động của các dịch vụ tính toán đầu cuối Weka và lập lịch theo các tiêu chí tối ưu mà người dùng đặt ra. Trong lớp này chứa các thuật toán lập lịch là phần trung tâm của bộ lập lịch. Chất lượng của bộ lập lịch phụ thuộc chủ yếu vào các thuật toán này. Lớp Broker không trực tiếp thực hiện công việc tính toán mà khởi tạo một tập các luồng mới từ lớp Worker sau đó theo dõi quá trình thực hiện công việc của các luồng này. Lớp Broker chịu trách nhiệm chính trong việc đưa ra các xử lý kịp thời ứng với sự thay đổi các nguồn tài nguyên trong lưới.
- *Lớp Worker:* chịu trách nhiệm điều khiển từng dịch vụ tính toán đầu cuối được sử dụng để thực hiện từng công việc cụ thể, phát hiện các loại lỗi phát sinh và thông báo kịp thời cho luồng chính Broker để xử lý.
- *Lớp Job:* chứa đặc tả các công việc đang được xử lý.
- *Lớp Resource:* chứa đặc tả các tài nguyên đang được dùng để xử lý.
- *Lớp Report:* chịu trách nhiệm ghi lại chi tiết toàn bộ quá trình thực hiện ứng dụng vào cơ sở dữ liệu làm cơ sở cho quá trình thanh toán sau này. Ngoài ra các thông tin do lớp Report ghi lại cũng được sử dụng làm nguồn tư liệu để đánh giá hiệu năng hoạt động của bộ lập lịch.

7.3.3. Chi tiết xây dựng và triển khai bộ lập lịch

Sau đây là thông tin chi tiết về việc cài đặt và triển khai bộ lập lịch trên ngôn ngữ lập trình Java.

7.3.3.1. Lớp SchedulerProvider



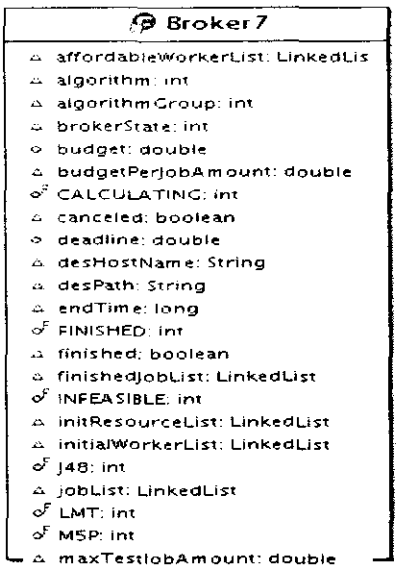
Hình 7-5: Lớp SchedulerProvider

Một số phương thức đáng chú ý:

- *setup():* đây là một phương thức đặc biệt không phải của dịch vụ cũng không phải do ta thêm vào mà là phương thức kế thừa từ lớp Agent. Đây là phương thức khởi tạo của Agent được gọi tự động từ platform của Agent. Do bộ lập lịch được thiết kế dưới dạng dịch vụ lai Agent nên nó vừa mang những đặc tính của Agent, vừa mang những đặc tính của dịch vụ lưới. Là một dịch vụ lưới, bộ lập lịch phải là một tiến trình tồn tại trong trình chứa dịch vụ lưới (Grid Service Container) và phải có các phương thức khởi tạo chuẩn của dịch vụ. Đồng thời, là một Agent, bộ lập lịch phải tồn tại trong platform của agent do vậy nó phải có các phương thức khởi tạo của Agent để gia nhập được vào hệ thống Agent.
- *takeDown():* tương tự như setup(), phương thức này chứa các hàm dùng để hủy bỏ đăng ký của bộ lập lịch khỏi hệ thống Agent và một số tác vụ dọn dẹp khác khi bộ lập lịch kết thúc hoạt động.
- *startScheduler*(int algorithmGroup,int algorithm,int numOfJob,double maxTestJobAmount,double deadline,double budget,int optStrategy) : phương thức khởi tạo bộ lập lịch với một bộ các tham số đầu vào như chi tiết dịch vụ cần sử dụng, các thông tin về giá cả và thời gian thực hiện ứng dụng, chiến lược tối ưu...

- *getResourceStatus()*: lấy thông tin về các tài nguyên hiện đang được bộ lập lịch sử dụng để chạy ứng dụng.
- *getJobStatus()*: lấy các thông tin về tình trạng các công việc đang được thực hiện trên lưới.
- *cancel()*: hủy bỏ toàn bộ việc chạy ứng dụng.

7.3.3.2. Lớp Broker



Hình 7-6: Lớp Broker

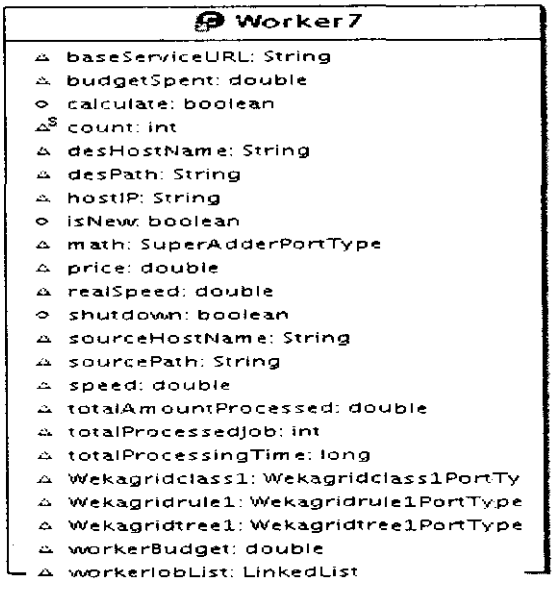
- *initialResourceExplore()*: Phương thức này sẽ lấy thông tin sơ bộ về tình hình tài nguyên lưới. Để có thể bước đầu lập lịch được thì bộ lập lịch phải có một số thông tin ban đầu về tình trạng tài nguyên lưới. Thông thường việc lập lịch sẽ được bắt đầu khi có một số lượng tài nguyên nhất định nào đó. Việc chờ để có được đầy đủ thông tin về tình trạng tài nguyên của toàn bộ lưới có vẻ không khả thi lắm trong một môi trường rộng lớn và phức tạp như vậy. Trong khi ta chờ đợi thông tin về tài nguyên này thì có thể thông tin ta đã có cũng trở nên lỗi thời. Do vậy cái mà ta cần ở đây là một tập nhỏ các tài nguyên vừa để để lập lịch.
- *initJobList()*: phương thức này đọc vào và khởi tạo hàng đợi các công việc. Tập công việc được đệ trình lên bộ lập lịch sẽ là tập các file đầu vào của Weka và thông tin quan trọng cần ghi lại là kích thước của file đầu vào, tức kích thước dữ liệu được xử lý để tiên đoán thời gian thực hiện công việc.
- *run()*: phương thức chính của luồng lập lịch. Như đã nói ở trên bộ lập lịch được thiết kế theo kiểu đa luồng nên bộ lập lịch sẽ chạy trong một luồng riêng.
- *makeInitialSchedule()*: lập bản kế hoạch khả thi ban đầu. Phương thức này sẽ khảo sát sơ bộ dựa trên tình hình tài nguyên hiện thời của lưới và các yêu cầu người dùng đặt ra xem có khả thi không và nếu có thì thực hiện như thế nào.
- *reschedule()*: do tài nguyên lưới liên tục biến động nên nói chung ta sẽ liên tục phải điều chỉnh kế hoạch thực hiện tức là lập lịch lại. Việc lập lịch lại được tiến hành định kỳ theo thời

gian hoặc ngay khi có sự kiện gì đó xảy ra. Ví dụ: có một công việc mới hoàn thành. Sẽ có tình huống xảy ra là lúc đầu kế hoạch thực hiện là khả thi nhưng trong thời gian thực hiện do tài nguyên lưới biến động quá lớn làm cho việc thực hiện ứng dụng ban đầu trở nên bất khả thi, Ví dụ: thời hạn thực hiện hoặc giá thành vượt quá giới hạn đặt ra ban đầu. Lúc đó bộ lập lịch sẽ phải tham khảo ý kiến người dùng về việc có tiếp tục thực hiện công việc hay không.

- *workerErr():* Phương thức xử lý lỗi của tài nguyên, phương thức này được gọi bởi các đối tượng Worker sẽ báo cho bộ lập lịch biết để có những xử lý kịp thời . Vd : lập lịch lại cho công việc bị lỗi.

- *workDone():* Phương thức báo có một công việc mới hoàn thành, được gọi bởi các đối tượng Worker. Phương thức này sẽ báo cho bộ lập lịch biết để lập lịch lại mỗi khi có công việc mới hoàn thành.

7.3.3.3. Lớp Worker



Hình 7-7: Lớp Worker

- *init():* Phương thức khởi tạo các đối tượng Worker. Lớp Broker không trực tiếp gọi các dịch vụ tính toán Weka mà thông qua các lớp Worker. Các lớp Worker được khởi tạo và chạy trong các luồng riêng của chúng cho đến khi nhận được tín hiệu kết thúc từ lớp Broker.

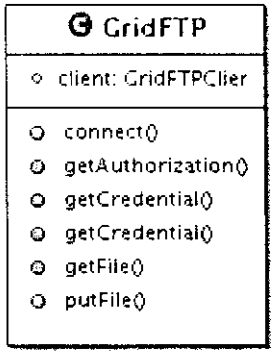
- *calculate():* Phương thức tính toán. Phương thức này sẽ được sử dụng để gọi các dịch vụ tính toán Weka.

- *jobReport():* Thông báo về trạng thái công việc cho lớp Broker để xử lý.

- *updateRealSpeed():* Cập nhật thông tin thực về tốc độ thực hiện công việc của tài nguyên. Khi chưa có công việc nào được thực hiện bộ lập lịch dựa vào các thông tin do tài nguyên cung cấp để lập lịch. Ví dụ tốc độ thực hiện. Khi đã có thông tin thực về các công việc được thực hiện trên tài nguyên đó thì tốc độ thực tế này sẽ được cập nhật lại liên tục cho phù hợp với tình trạng hiện thời của tài nguyên để có những xử lý thích hợp.

- *addJob(Job j):* gán thêm công việc cho tài nguyên hiện tại.
- *removeLastJob():* loại bỏ bớt 1 công việc đã được gán cho tài nguyên hiện tại. Phương thức này được sử dụng khi tài nguyên bị quá tải, không thể thực hiện hết số lượng công việc được giao trong thời hạn đã định.

7.3.3.4. Lớp GridFTP



Hình 7-8: Lớp GridFTP

Lớp này được sử dụng bởi bộ lập lịch để chuyển các file dữ liệu của Weka đến các tài nguyên đã được lựa chọn. Truyền file tin cậy trong môi trường lưới là một trong những dịch vụ được cung cấp bởi Globus Toolkit. Để truyền được file bộ lập lịch phải có được các giấy chứng nhận cần thiết (để chứng thực) và thỏa thuận về việc sử dụng dịch vụ truyền file (được ấn định trong grid-mapfile trên tài nguyên). Các phương thức chính:

- *putFile():* Chuyển 1 file từ máy hiện thời đến 1máy khác trong lưới.
- *getFile():* Lấy 1 file từ một máy khác về máy hiện thời.
- *getCredential():* Đọc file giấy chứng nhận của người dùng.

7.3.4. Cài đặt các thuật toán lập lịch

Các thuật toán lập lịch được cài đặt trong bộ lập lịch cho ứng dụng trên BK Grid 2005 được dựa trên tinh thần của một số thuật toán tối ưu đã trình bày ở phần trên. Các thuật toán lập lịch được chia làm 2 phần: lập lịch ban đầu và lập lịch lại.

- Lập lịch ban đầu: mục đích của lần lập lịch này là đưa ra một bản lập lịch sơ khởi cho yêu cầu của người dùng. Trong lần lập lịch này bộ lập lịch sẽ quyết định xem liệu yêu cầu của người dùng có thể được đáp ứng không với tình trạng tài nguyên lưới ở thời điểm hiện tại. Bộ lập lịch lấy các thông tin về tài nguyên lưới được cung cấp bởi hệ thống Agent sau đó dựa vào các thông tin tài nguyên này để lập lịch theo các chiến lược tối ưu mà người dùng đã chọn. Nếu việc lập lịch cho yêu cầu của người dùng dẫn đến một chi phí vượt quá chi phí ban đầu mà người dùng đã ấn định hoặc thời gian vượt quá thời gian cho phép thì bộ lập lịch kết luận yêu cầu của người dùng không thể thực hiện được và người dùng phải tăng thêm thời gian/chi phí để yêu cầu có thể khả thi ở thời điểm hiện tại. Tất nhiên dự đoán của bộ lập lịch không thể hoàn toàn chính xác được vì tài nguyên lưới có sự biến động liên tục

nên một yêu cầu khả thi ở thời điểm hiện tại vẫn có thể trở thành bất khả thi trong tương lai và ngược lại. Tuy nhiên đây cũng là một gợi ý cho người dùng xem có nên thực hiện yêu cầu ngay hay không.

- Lập lịch lại: do tài nguyên lưới sẽ biến động liên tục nên bộ lập lịch chắc chắn sẽ phải thực hiện lập lịch lại. Khi thực hiện lập lịch lại, bộ lập lịch sẽ thực hiện điều chỉnh kế hoạch thực hiện sao cho phù hợp với tình hình hiện tại của tài nguyên lưới. Để thực hiện điều này bộ lập lịch phải theo dõi lịch sử thực hiện công việc của tài nguyên và tiên đoán khả năng thực hiện công việc trong tương lai. Khác với lần lập lịch ban đầu sử dụng các thông tin về tài nguyên như tốc độ CPU để ấn định lượng công việc phù hợp với khả năng của từng máy, trong lần lập lịch này bộ lập lịch trực tiếp đo tốc độ này (ví dụ Mb/s) thông qua việc ngoại suy từ các lần thực hiện trước đó. Có một số phương pháp như đo vận tốc thực hiện tức thời hoặc dùng phương trình ngoại suy đa thức hay hàm mũ. Tuy vậy việc dự đoán này là một vấn đề khá phức tạp và vẫn đang được tiếp tục nghiên cứu trên thế giới.

7.3.5. Xử lý và chống lỗi

Một trong những yêu cầu quan trọng đối với ứng dụng lưới là khả năng xử lý và chống lỗi. Trong BK Grid 2005 nhiệm vụ xử lý và chống lỗi được phân chia đều cho cả phía bộ lập lịch lẫn phía dịch vụ lưới chịu trách nhiệm xử lý thông tin (Weka). Tùy vào mức độ nghiêm trọng của lỗi mà cách thức xử lý sẽ khác nhau. Sau đây là một số loại lỗi và cách thức xử lý:

- Lỗi tài nguyên: Lỗi này là lỗi nghiêm trọng xảy ra khi vì một lý do nào đó tài nguyên không thể tiếp tục hoạt động được nữa. Ví dụ mất đường truyền mạng, trung tâm máy tính bị mất điện... Khi gặp loại lỗi bất khả kháng này bản thân phía dịch vụ xử lý thông tin (các dịch vụ Weka) không có cách nào báo được cho bộ lập lịch và bộ lập lịch phải chủ động xác định loại lỗi này. Như đã nói trong phần thiết kế bộ lập lịch, các đối tượng thuộc lớp Worker là các đối tượng trực tiếp làm việc với từng dịch vụ lưới trên từng tài nguyên do vậy chính các đối tượng này sẽ trực tiếp xác định các loại lỗi kể trên. Cơ chế xác nhận chủ yếu dựa vào cơ chế time-out của phía gọi dịch vụ lưới: trong thời gian dịch vụ lưới được triệu gọi phía client của dịch vụ lưới sẽ thiết lập một đồng hồ thời gian đếm ngược. Hết thời gian đã định thì dịch vụ lưới trên máy đó coi như không thể liên lạc được và bị loại ra khỏi danh sách tài nguyên sử dụng. Thông tin này cũng được hệ thống Agent theo dõi tài nguyên cập nhật liên tục để trong các lần hoạt động sau bộ lập lịch sẽ không sử dụng tài nguyên này. Trong trường hợp tài nguyên vừa bị lỗi nghiêm trọng được phục hồi lại và hoạt động bình thường (ví dụ sau khi khởi động lại), tài nguyên được coi như mới gia nhập lưới và hệ thống Agent sẽ báo cho bộ lập lịch về sự hiện diện của tài nguyên lưới mới này để sử dụng nếu cần thiết trong các lần lập lịch tiếp theo.
- Lỗi do dịch vụ: Lỗi này xảy ra do dịch vụ chạy bị lỗi và không thể xử lý tiếp. Ví dụ: lỗi out of memory xảy ra khi dịch vụ Weka xử lý file dữ liệu có dung lượng quá lớn vượt quá khả năng

của tài nguyên đó. Khi gặp lỗi này dịch vụ phía xử lý cũng không thể tự xác định và thông báo cho bộ lập lịch được mà bộ lập lịch phải tự xác định. Cơ chế xác định tương tự như trên. Trong trường hợp này các thức xử lý là bộ lập lịch sẽ cố gắng khởi tạo một thể hiện (instance) khác của dịch vụ xử lý.

- Lỗi do dữ liệu xử lý: loại lỗi này thường xảy ra do dữ liệu được xử lý không phù hợp với thuật toán đang được sử dụng hoặc do một số nguyên nhân khác. Đặc điểm chung của các loại lỗi này là được phát hiện và thông báo chủ động cho bộ lập lịch từ phía dịch vụ xử lý dữ liệu. Các giải quyết đơn giản là bộ lập lịch sẽ thông báo lỗi cho người dùng và chuyển bộ dữ liệu khác đến cho dịch vụ để tiếp tục xử lý.

- Lỗi do bảo mật: lỗi này xảy ra do bộ lập lịch không có một giấy uỷ quyền (proxy) hợp lệ để sử dụng các tài nguyên, dịch vụ. Khi gặp lỗi này người dùng phải chủ động sử dụng dịch vụ bảo mật do BK Grid 2005 cung cấp để lấy một giấy uỷ quyền hợp lệ và có đủ thời gian sử dụng, nhất là với những bộ dữ liệu lớn có thời gian xử lý lâu dài.

7.3.6. Một số vấn đề khác trong lập lịch

7.3.6.1. Vấn đề xử lý tương tranh tài nguyên

Vấn đề xử lý tương tranh tài nguyên là một trong những vấn đề nổi bật của các hệ thống quản lý tài nguyên tập trung cổ điển. Tương tranh tài nguyên liên quan đến vấn đề nhiều người dùng có cùng yêu cầu sử dụng một loại tài nguyên nào đó trong khi sức đáp ứng của tài nguyên này có hạn. Trong trường hợp này hệ thống quản lý tài nguyên phải có phương án sắp xếp để tài nguyên được sử dụng cho những mục đích ưu tiên hơn, ví dụ cho người dùng có đặc quyền cao hơn trong hệ thống. Ta hãy xem vấn đề này được giải quyết như thế nào trong môi trường lưới hướng kinh tế nói chung và trong môi trường lưới BK Grid 2005.

Như đã nói trong chương 2, mô hình lưới hướng kinh tế gồm có 2 thành phần cơ bản, đó là nhà cung cấp dịch vụ và người sử dụng dịch vụ. Hai thành phần này có tương đối độc lập với nhau và chỉ bị ràng buộc với nhau bởi các lợi ích kinh tế: người sử dụng được sử dụng tài nguyên dịch vụ để hoàn thành công việc của mình còn người cung cấp thu được lợi nhuận từ việc cho thuê tài nguyên. Thông thường cùng một loại dịch vụ sẽ có nhiều nhà cung cấp khác nhau và người sử dụng sẽ lựa chọn một nhà cung cấp phù hợp để hoàn thành yêu cầu của mình dựa trên những tham số chính như giá cả, chất lượng phục vụ... Về phía người chủ sở hữu tài nguyên tất nhiên chính sách của họ là thu hút được càng nhiều khách hàng càng tốt để gia tăng lợi nhuận đến mức tối đa. Thế nhưng việc gia tăng số người sử dụng tại cùng một thời điểm đồng nghĩa với việc chất lượng phục vụ bị giảm xuống vì khả năng xử lý tài nguyên bị chia sẻ cho nhiều người dùng. Khi chất lượng dịch vụ giảm xuống thì một số người sử dụng sẽ bỏ đi tìm nhà cung cấp dịch vụ khác để đảm bảo công việc của mình được thực hiện theo đúng kế hoạch. Do vậy để đảm bảo chất lượng dịch vụ nhà cung cấp sẽ phải khống chế số lượng người dùng đồng thời đến tài nguyên hay dịch vụ nào đó và

như vậy sẽ có những tình huống xảy ra tranh chấp là có nhiều người cùng đến sử dụng dịch vụ vượt quá khả năng đáp ứng của tài nguyên. Khi này chỉ một số người sử dụng được quyền sử dụng tài nguyên và điều này do nhà cung cấp dịch vụ quyết định.

Trong mô hình lưới hướng kinh tế người ta đã đề xuất một số cách giải quyết vấn đề này:

- Đến trước phục vụ trước (First Come First Serve)
- Mô hình đấu giá: tài nguyên không phục vụ ngay mà gom một số yêu cầu lại sau đó yêu cầu nào của người dùng trả giá cao nhất sẽ được thực hiện.

Trong BKGrid2005, tài nguyên được quản lý hoàn toàn bởi hệ thống Agent đại diện. Mặc dù các dịch vụ lưới được thiết kế theo mô hình xưởng chế tác (Factory) có khả năng phục vụ không hạn chế - tức là có thể sinh ra bao nhiêu thể hiện của dịch vụ cũng được, nhưng để đảm bảo chất lượng dịch vụ Agent đại diện sẽ khống chế số lượng thể hiện được sinh ra đồng thời này, ví dụ 5 thể hiện, nghĩa là dịch vụ có thể thực hiện đồng thời 5 yêu cầu của người dùng. Trước khi được phục vụ các bộ lập lịch và môi giới tài nguyên của người dùng phải liên hệ trước với các Agent đại diện của tài nguyên để giành được quyền sử dụng. Cơ chế phục vụ theo cơ chế đến trước phục vụ trước. Trong trường hợp tài nguyên đã phục vụ hết khả năng những người dùng đến sau sẽ phải đi tìm tài nguyên mới để thực hiện công việc của mình.

7.3.6.2. Vấn đề đặt trước tài nguyên

Để đảm bảo các yêu cầu của mình được thực hiện đúng hạn có nhiều khi người dùng sẽ phải đặt chỗ trước tài nguyên. Ví dụ có một người dùng có một tập công việc khá lớn mà anh ta chưa thể thực hiện ngay ở thời điểm hiện tại mà phải đợi đến một thời điểm nào đó trong tương lai. Tập công việc này có những yêu cầu ngặt nghèo về thời gian thực hiện. Do tình trạng tài nguyên lưới trong tương lai là khó có thể tiên đoán được chính xác nên tốt nhất là người dùng này sẽ đặt trước một tập các tài nguyên anh ta dự định sẽ dùng để hoàn thành tập công việc đó. Như vậy đến khi thực hiện công việc anh ta được đảm bảo là sẽ luôn luôn được quyền sử dụng tập tài nguyên đã đặt trước bất kể tình trạng cung cầu về tài nguyên lưới biến đổi ra sao.

Việc đặt trước tài nguyên hoàn toàn do phía chủ sở hữu tài nguyên hỗ trợ. Trong BK Grid 2005 hệ thống Agent quản lý toàn bộ tài nguyên lưới sẽ cung cấp chức năng này cho người sử dụng. Mỗi Agent quản lý tài nguyên sẽ có một bảng khe thời gian để theo dõi số thể hiện của dịch vụ ở từng thời điểm. Số thể hiện tối đa hay số người dùng được phục vụ đồng thời sẽ tuỳ thuộc vào khả năng của từng tài nguyên và sẽ do Agent khống chế. Để đặt trước tài nguyên người sử dụng cần phải cung cấp định danh của mình trong môi trường lưới và đăng ký với Agent quản lý về khoảng thời gian mà anh ta muốn sử dụng dịch vụ. Agent quản lý sẽ kiểm tra xem trong khoảng thời gian mà anh ta muốn đặt có những người dùng nào đã đặt chỗ rồi và còn có thể phục vụ được thêm người dùng mới không. Nếu được chấp nhận

định danh của người dùng sẽ được ghi vào bảng và anh ta đã thành công trong việc đặt trước quyền sử dụng tài nguyên đó.

Khi các bộ lập lịch và môi giới tài nguyên có yêu cầu sử dụng dịch vụ, chúng sẽ phải xuất trình các giấy uỷ nhiệm hợp lệ. Agent quản lý tài nguyên sẽ kiểm tra định danh trong giấy uỷ nhiệm của người dùng xem có trùng với một yêu cầu đặt trước tài nguyên của một người dùng nào đó trong khoảng thời gian hiện tại không. Nếu có yêu cầu của người dùng sẽ được chấp nhận ngay. Ngược lại nếu khách hàng này không có tên trong danh sách đặt trước Agent quản lý tài nguyên sẽ kiểm tra xem ở thời điểm hiện tại xem tổng số các khách hàng được phục vụ với số khách hàng đã đặt trước xem có vượt qua số lượng cho phép chưa. Nếu chưa thì yêu cầu của khách hàng cũng được đáp ứng ngay, ngược lại thì nó sẽ từ chối phục vụ và người sử dụng phải đi tìm các tài nguyên khác.

Chương 8: Các dịch vụ khai phá dữ liệu

Hệ thống BKGrid 2005 đã tiến hành lưới hóa các mô đun của ứng dụng Weka thành các dịch vụ lưới. Weka là một chương trình nguồn mở do trường đại học Wakailo, New Zealand phát triển với mục đích phục vụ cho bài toán khai phá dữ liệu và đặc biệt là khâu phân tích dữ liệu. Weka là một ứng dụng tương đối tốt trong lĩnh vực khai phá dữ liệu và đặc biệt là phù hợp khi triển khai trên lưới. Bởi lẽ, chương trình Weka được thực thi dưới dạng lô, tức là mỗi khi có yêu cầu khai phá dữ liệu, chương trình sẽ nhận đầu vào và thực hiện việc xử lý mà không có sự tương tác trực tiếp với người dùng, sau đó nếu thành công thì kết quả được trả về hoặc nếu thất bại thì thông báo lỗi cho người sử dụng. Thứ hai, do nhu cầu xử lý khối lượng rất lớn dữ liệu đầu vào nên chương trình Weka được thực thi trên máy đơn sẽ không có hiệu năng như mong muốn. Trên thực tế, chương trình này đang được phát triển tiếp theo hướng song song hóa. Chính vì vậy, hệ thống BKGrid 2005 sẽ lưới hoá chương trình weka để thể hiện các nghiên cứu về phát triển ứng dụng lưới.

8.1. Xây dựng ứng dụng trên lưới

8.1.1. Phương pháp luận phát triển ứng dụng lưới

Bất kỳ một ứng dụng nào muốn được triển khai cũng phải đáp ứng được các yêu cầu của người dùng và phải phù hợp với các điều kiện về tài nguyên của tổ chức. Trong phần này chúng ta sẽ xem xét các tiêu chuẩn, các hướng tiếp cận và các phương pháp để triển khai ứng dụng lưới cũng như các yêu cầu đối với ứng dụng được lưới hóa.

8.1.1.1. Các tiêu chuẩn để triển khai một ứng dụng lưới

Cũng như trong tất cả các công việc khác khi muốn triển khai một ứng dụng thì cần phải đưa ra được những đánh giá về hiệu quả mà công việc mang lại có tương xứng với sự đầu tư. Sau đây là một số các điểm mạnh của lưới mang lại: thứ nhất, lưới cung cấp một sức mạnh có thể đáp ứng được các yêu cầu đặc biệt của các ứng dụng tính toán hiệu năng cao. Thứ hai, tính toán lưới có thể cung cấp một môi trường cho các ứng dụng phần mềm đặc biệt thực hiện được nhanh hơn. Cuối cùng, các lưới có thể mang lại hiệu quả sử dụng tài nguyên cao hơn ví dụ như đối với các máy tính cá nhân để bàn hiệu quả sử dụng tài nguyên thông thường chỉ là 10% và đối với các máy phục vụ thì tỷ lệ này là 30% tuy nhiên với môi trường lưới thì tỷ lệ này là từ 80-90%.

Trong các mô hình xí nghiệp doanh nghiệp, các yêu cầu về tính toán đòi hỏi phải có một chiến lược cao, các lưới được sử dụng để cung cấp khả năng sức mạnh tính toán thay thế cho các siêu máy tính đắt tiền. Ví dụ như trong khoa học nghiên cứu Gen, protein, phát minh

thuốc, đưa ra các dự báo hay mô hình hóa các dự đoán trong công nghiệp dầu mỏ hoặc khí ga ... là những ví dụ điển hình về tính toán hiệu năng cao.

Khả năng tính toán tuyệt vời mà lưới đem lại là một trong những đặc tính quan trọng của nó. Một ứng dụng được thực hiện nhanh hơn sẽ mang lại nhiều lợi thế trong công việc. Lưới có thể cung cấp khả năng tăng tốc độ thực hiện của ứng dụng theo mô hình tuyến tính hoặc gần tuyến tính. Điều này có nghĩa là với một ứng dụng khi thực hiện bởi 2 máy trên lưới thì thời gian thực hiện có thể giảm đi một nửa, và khi là 3 máy thì thời gian thực hiện có thể giảm xuống bằng một phần ba so với thời gian khi thực hiện ứng dụng đó trên máy đơn.

Một trong những lợi ích thuyết phục nhất của tính toán lưới đó là khả năng sử dụng tài nguyên một cách hiệu hơn. Với hiệu suất sử dụng tài nguyên lên tới 80% 90% có nghĩa là tăng hiệu suất sử dụng của máy phục vụ lên 3 lần và tăng hiệu suất sử dụng với máy tính cá nhân để bàn lên 10 lần. Việc tăng hiệu suất sử dụng các tài nguyên trực tiếp ảnh hưởng tới việc giảm giá thành sản phẩm và giá thành bảo trì bảo dưỡng.

8.1.1.2. Các phương pháp triển khai ứng dụng lưới

Điểm mấu chốt trong việc có hay không triển khai ứng dụng lưới đó là nó có tận dụng được tất cả các lợi thế mà lưới hay tổ chức tài nguyên tính toán ảo mang lại hay không. Các ứng dụng phù hợp nhất để triển khai dưới dạng ứng dụng lưới đó là các ứng dụng mà đòi hỏi phải xử lý một số lượng dữ liệu lớn và cũng đòi hỏi phải sử dụng một số lượng tài nguyên cũng rất lớn..

Có hai phương pháp để triển khai các ứng dụng lưới đó là: phương pháp xử lý phân tán các lô theo kiểu hàng đợi (scripted batch queue distribution method), và phương pháp xử lý các tiến trình phân tán một cách song song ngay từ trong mã của chương trình (programmatic method of coding for parallel) .

Với hướng tiếp cận hàng đợi lô thì ít phải chỉnh sửa mã nguồn, thực hiện bằng cách đồng thời chạy các công việc trên các máy khác nhau theo một kịch bản nào đó. Các công việc người dùng đệ trình được sắp xếp trong một hàng đợi, và bộ lập lịch sẽ cấp phát các tài nguyên cần thiết được lấy từ bộ quản lý và môi giới tài nguyên để thực hiện công việc. Ví dụ có 10 công việc cần được thực hiện, trong đó mỗi công việc thực hiện sẽ mất 1 giờ đồng hồ trên mỗi máy đơn, khi thực hiện 10 công việc này theo hướng tiếp cận hàng đợi lô thì chỉ mất 1 giờ để thực hiện khi mỗi công việc được thực hiện đồng thời trên một máy trong khi phải mất đến 10 giờ để thực hiện song 10 công việc khi thực hiện cả 10 công việc đó trên 1 máy. Như vậy là các yêu cầu để thực hiện theo hướng tiếp cận này có hiệu quả đó là phải có một số lượng lớn các công việc đơn, đồng thời phải có một số lượng lớn tài nguyên sẵn sàng để xử lý các công việc. Chỉ áp dụng được cho các ứng dụng có kịch bản (scriptable).

Với hướng tiếp cận thực hiện phân tán song song thực hiện từ mã chương trình thì nó đòi hỏi chương trình nguồn phải được chỉnh sửa sao cho khi triển khai thực hiện chương trình, chương trình nguồn thực hiện một công việc lớn sẽ được tự động phân chia thành các công

việc con độc lập, các công việc con độc lập này có thể lại được phân chia thành các công việc con nhỏ hơn độc lập với nhau,...Quá trình phân chia này có thể tiếp tục cho đến khi nào các công việc được phân chia ra có thể được thực hiện riêng rẽ trên các máy đơn. Các kết quả đạt được sẽ được tổng hợp ngược lại để đưa ra kết quả cuối cùng. Phương pháp này có thể áp dụng cho cả hai loại ứng dụng có kịch bản(scriptable) và phi kịch bản (non-scriptable).

8.1.1.3. Các hướng tiếp cận khi triển khai ứng dụng lưới

Một ứng dụng có thể là có kịch bản hay phi kịch bản. Từ việc xác định loại ứng dụng có thể đưa ra được hướng tiếp cận phù hợp nhất đối với việc triển khai lưới hóa ứng dụng đó.

Khi một ứng dụng có dạng là phi kịch bản thì nó chỉ có thể được lưới hóa theo hướng tiếp cận phân tán song song từ trong mã lệnh. Với ứng dụng là có kịch bản thì nó có thể lưới hóa theo cả hai hướng tiếp cận kể trên.

Sau đây là 3 yếu tố quan trọng để xác định hướng tiếp cận lưới hóa thích hợp nhất cho các ứng dụng có kịch bản, đó là:

Số lượng công việc cần được xử lý

Số lượng tài nguyên sẵn dùng

Kích thước của các công việc và khả năng của tài nguyên.

Với giải pháp hàng đợi lô thích hợp với lưới hóa các công việc mà lượng tài nguyên của công việc đó yêu cầu tương đương với lượng tài nguyên sẵn dùng có thể đáp ứng của lưới, khi lượng tài nguyên yêu cầu và lượng tài nguyên sẵn dùng là tương đương thì công việc sẽ được thực hiện một cách tối ưu khi sử dụng phương pháp này. Giải pháp hàng đợi lô hiệu quả nhất khi quan hệ giữa công việc và tài nguyên là quan hệ nhiều-nhiều, và sự quản lý tài nguyên, điều khiển người dùng được yêu cầu ở mức cao.

Thêm vào đó số lượng, kích thước của các công việc và khả năng của các tài nguyên cũng có mối quan hệ rất lớn. Chỉ có các tài nguyên có khả năng cao mới có thể thực hiện được các công việc lớn. Khi quan hệ giữa khả năng của tài nguyên và kích thước của công việc là không phù hợp thì hiệu quả của hướng tiếp cận hàng đợi lô sẽ không cao

Sau đây là một số các yếu tố dẫn tới việc lựa chọn phương pháp thực hiện phân tán song song từ trong mã lệnh:

Có ít công việc.

Số lượng tài nguyên ít.

Đặc biệt kích thước của các công việc là rất lớn.

Khả năng của các nguồn tài nguyên là nhỏ.

Tần suất đệ trình công việc là không liên tục.

Sự triển khai theo hướng "hàng đợi lô" bị ảnh hưởng bởi các yếu tố như giá cả hoặc các yêu cầu hệ thống không được đáp ứng.

Phương pháp hàng đợi lô phù hợp với các ứng dụng phân tán dạng có kịch bản thực hiện trong môi trường xí nghiệp với nhiều người sử dụng, nguồn tài nguyên lớn, có đầy đủ các nguồn tài nguyên về công nghệ thông tin, và có các chính sách và luật trong công việc phù hợp với quyền ưu tiên và sử dụng với các tài nguyên tính toán. Trong khi đó với các ứng dụng tiếp cận theo hướng phân tán song song từ trong dòng lệnh thì nó phù hợp đối với các mô hình xí nghiệp nhỏ hơn, những ứng dụng có cường độ xử lý cao hoặc đối với các lưới có khả năng thấp. Ngoài ra nó còn được sử dụng riêng để triển khai việc phân tán các ứng dụng phi kịch bản

Mặc dù sự khác biệt giữa 2 phương pháp tiếp cận triển khai ứng dụng lưới là tương đối lớn, tuy nhiên chúng không phủ nhận nhau mà chúng bổ sung cho nhau một cách tích cực. Bằng sử dụng các API của Distributed Resource Management Application API (DRMAA) và OGSA, phương pháp tiếp cận theo hướng phân tán song song từ dòng lệnh có thể thiết lập một giao diện với phần mềm DRM để có thể tạo ra được một môi trường tích hợp mà ở đó cả các ứng dụng dạng có kịch bản lẫn các ứng dụng phi kịch bản cùng được lưới hóa trong cùng một cơ sở hạ tầng lưới và được quản trị bởi một môi trường thống nhất.

8.1.1.4. Các yêu cầu đối với ứng dung được lưới hóa

Có hai yêu cầu đối với quá trình chỉnh sửa và triển khai một ứng dụng thông thường thành một ứng dụng lưới đó là:

có thể truy cập được vào mã nguồn.

chỉnh sửa được nó.

Có ba thành phần trong hệ thống liên quan đến các yêu cầu trên:

+ Nhóm thứ nhất là các nhà cung cấp phần mềm là những người phát triển (theo hướng ứng dụng lưới) lưới và thương mại các ứng dụng phần mềm. Các nhà cung cấp phần mềm có trong tay mã nguồn của sản phẩm và đội ngũ phát triển các phần mềm đó.

+ Nhóm thứ hai đó là các trung tâm nghiên cứu và các xí nghiệp nghiên cứu các lĩnh vực khoa học trong cuộc sống có sử dụng các ứng dụng phần mềm nguồn mở. Phần mềm nguồn mở cho phép được sửa mã chương trình.

+ Nhóm thứ ba đó là các xí nghiệp triển khai các ứng dụng phần mềm sở hữu của riêng mình, nhằm bảo mật sự triển khai thực hiện công việc của họ thông qua sự thực thi cao cấp của công nghệ thông tin.

Cả các ứng dụng phần mềm nguồn mở và các phần mềm độc quyền đều có thể được chỉnh sửa để triển khai dưới dạng các ứng dụng phần mềm lưới thông qua đội ngũ phát triển phần mềm của chính công ty đó hoặc thông qua các nhà cung cấp phần mềm.

8.1.2. Tiến trình lưới hóa ứng dụng

Quá trình lưới hóa một ứng dụng người phát triển luôn luôn phải quan tâm đến hiệu năng của ứng dụng sau khi ứng được lưới hóa, bởi vì không phải tất cả các ứng dụng đều mang

lại hiệu quả cao khi được thực hiện trên lưới. Một ứng dụng được lưới hóa một cách hiệu quả nó phải có các tính chất sau:

Các công việc đó phải có thể được phân chia thành các công việc nhỏ hơn.

Mỗi một công việc nhỏ có thể được thực hiện một cách độc lập trên một máy tính.

Các kết quả được trả lại từ thực hiện từ các công việc nhỏ có thể gắn kết được với nhau và cho ra kết quả cuối cùng.

Một công việc hay một thuật toán có thể được thực hiện lại dưới dạng một ứng dụng phân tán sẽ được triển khai lại với một số các bước sau:

8.1.2.1. Quá trình phân tích

Để lưới hóa một ứng dụng thì trước hết nó phải được phân tích để chỉnh sửa các chức năng cơ bản trong mã lệnh của chương trình.

Bước đầu tiên trong quá trình phân tích mã lệnh đó là phải xác định ứng dụng đó sẽ được phân chia làm bao nhiêu các đơn vị công việc; mỗi một công việc là một đơn vị chương trình có thể thực hiện độc lập trên một máy đơn. Các thuật toán được phân chia thành hai loại: các thuật toán có thể dễ dàng phân chia thành các tiến trình công việc nhỏ hơn mà các tiến trình đó có thể thực hiện độc lập với nhau. Những thuật toán với tính chất như vậy gọi là *embarrassingly parallel*. Các ứng dụng có kịch bản mà có thể triển khai theo hướng hàng đợi lô luôn luôn chứa các thuật toán *embarrassingly parallel*; tuy nhiên, cũng có rất nhiều các ứng dụng phi kịch bản có chứa các thuật toán *embarrassingly parallel* và có thể dễ dàng chỉnh sửa để triển khai chúng trên lưới theo hướng tiếp cận phân tán song song từ trong mã lệnh.

Hoàn toàn ngược lại, các thuật toán phụ thuộc yếu (*tightly coupled)* có những phần cần phải thực hiện trao đổi kết quả với nhau trong quá trình thực hiện. Với những thuật toán dạng này thì thông thường sẽ khó khăn hơn trong quá trình chỉnh sửa mã lệnh vì nó bắt buộc phải được tiếp cận phân tán song song từ dòng lệnh.

Bước thứ hai trong quá trình phân tích là để xác định những thành phần cần thiết cho mỗi công việc khi chúng được thực hiện riêng rẽ trên một máy đơn. Các yêu cầu chi tiết đối với một ứng dụng đó là các file thành phần, thư viện cần thiết, cơ sở dữ liệu và các yêu cầu đặc biệt hoặc các yêu cầu về phần cứng. Các yêu cầu cơ bản về môi trường thực hiện đối với một ứng dụng này sẽ được tự động gửi đến lưới như một phần của ứng dụng mỗi khi ứng dụng được thực hiện.

Bước cuối cùng trong quá trình phân tích là định nghĩa kiểu của các kết quả sẽ trả về và xác định quá trình xử lý tiếp theo đối với các kết quả đó là chúng sẽ được lưu trữ, được xử lý tiếp hay vừa được xử lý tiếp vừa được lưu trữ. Cần lưu ý là các kết quả thu được khi thực hiện trong môi trường phân tán sẽ có thể không được thực hiện đồng bộ như trong định nghĩa.

8.1.2.2. Quá trình chỉnh sửa lại ứng dụng

Đối với một ứng dụng đã được triển khai thì rõ ràng các chức năng của nó đã được xác định thông qua quá trình phân tích thiết kế. Tuy nhiên, khi lưới hóa ứng dụng đó thì toàn bộ các chức năng đó lại phải định nghĩa lại một lần nữa.

Quá trình này thực hiện việc phân chia các chức năng sẵn có của ứng dụng ban đầu thành các chức năng nhỏ hơn có thể thực hiện được trên lưới. Đồng thời nó xác định tất cả các dữ liệu cần thiết cho quá trình thực hiện của mỗi một chức năng vừa được phân chia.

Chú ý rằng kích thước của các chức năng có thể biến đổi dựa theo các ràng buộc của môi trường triển khai, ví dụ như tốc độ kết nối, mạng để triển khai lưới là động hay tĩnh, độ tin cậy cũng như sự sẵn dùng của các máy tính trong mạng,... Dựa vào các dàng buộc đó, kích thước mặc định của một chức năng được xác định là giá trị cực đại của tỷ lệ giữa thời gian thực hiện tính toán của một chức năng với thời gian cần thiết để thiết lập kết quả và thời gian giao tiếp.

Bước tiếp theo đó là xây dựng các chức năng vừa được phân chia ở bước trước thành các chức năng của lưới, mà các chức năng lưới mà chúng ta xây dựng ở đây là các chức năng được xây dựng theo hướng dịch vụ.

Bước cuối cùng trong quá trình lưới hóa một ứng dụng là phải xây dựng một cơ chế để tổng hợp các kết quả đạt được từ các chức năng nhỏ đó thành kết quả tương ứng với kết quả thực hiện của chức năng của ứng dụng ban đầu. Các kết quả thu được có thể sẽ không theo thứ tự như trong quá trình xây dựng các chức năng, cho nên cần phải có một cơ chế lưu trữ các kết quả trả về. Sau khi tất cả các kết quả mong đợi đã được trả về thì các kết quả đó sẽ được tích hợp với nhau để cho kết quả đầu ra hay có thể nó lại là đầu vào cho một quá trình thực hiện tiếp theo.

8.1.3. Sáu bước lưới hoá ứng dụng

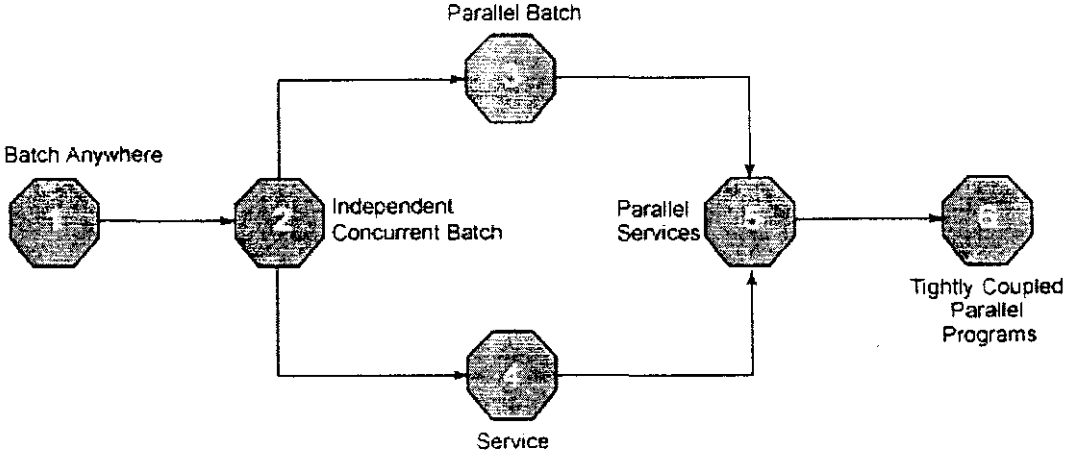
Một ứng dụng chỉ được coi là ứng dụng lưới khi nó tận dụng được các lợi thế do cơ sở hạ tầng tổ chức ảo đem lại để tăng tốc độ xử lý hay khả năng kết nối của lưới. Không chỉ có vậy, theo đặc tả OGSA thì ứng dụng lưới phải được chạy như là một dịch vụ Web trong môi trường lưới trong khi vẫn sử dụng được các dịch vụ nền tảng được cung cấp bởi cơ sở hạ tầng lưới. Ví dụ như nền tảng J2EE (Java 2 Enterprise Edition) chạy như là một dịch vụ Web trên nền phần đệm hỗ trợ lưới (grid-enable middleware) WebSphere Application Server (WSAS) và sử dụng các dịch vụ được cung cấp bởi WSAS và cơ sở hạ tầng lưới.

Bước đầu tiên của quá trình lưới hoá là khảo sát quá trình thực hiện đơn giản của một ứng dụng trong một lưới. Sau đó là tìm hiểu cách các ứng dụng cần lưới hoá sử dụng hai bước đầu tiên để có thể thực thi như là các công việc theo lô đơn lẻ hoặc song song. Tiếp theo, các bước ba và bốn chỉ ra cách chia nhỏ một ứng dụng theo lô và dịch vụ hoá ứng dụng đó để phục vụ người sử dụng thông qua phần đệm. Hai bước cuối cùng sẽ đưa ra một số dạng

thức (paradigm) khả thi để triển khai một ứng dụng sử dụng dịch vụ lưới theo cách song song. Mặc dù các bước được đề cập dưới đây là cần thiết trong quá trình lưới hoá ứng dụng nhưng không bắt buộc phải thực hiện đủ cả năm bước. Rất nhiều ứng dụng chỉ đạt được bước thứ năm và rất ít có thể đạt được bước sáu. Bước thứ sáu được chuyên biệt hoá cho một số ứng dụng và được thiết kế từ ban đầu để hỗ trợ cho mô hình xử lý song song kết nối chặt (tightly coupled) cho các chương trình mà các chương trình này tạo nên ứng dụng hoàn chỉnh.

Ngoài ra, ứng dụng không cần thiết phải thực hiện các bước một cách tuần tự, từng bước một.

Hình vẽ sau đây sẽ mô tả sáu bước lưới hóa ứng dụng:



Hình 8-1: Sáu bước lưới hóa ứng dụng

+ Bước một: Xử lý công việc theo lô
+ Bước hai: Xử lý theo lô đồng thời
+ Bước ba: Xử lý theo lô song song
+ Bước bốn: Dịch vụ
+ Bước năm: Dịch vụ song song
+ Bước sáu: Chương trình song song phụ thuộc chặt (Tightly Coupled)

Các bước này có thể được phân thành các giai đoạn như sau:

+ Giai đoạn thực hiện: bước một, bước hai và dạng thức đơn giản nhất của bước ba với mục đích tập trung vào khả năng của một ứng dụng có thể được chạy trên lưới.

+ Giai đoạn thích ứng: dạng thức phức tạp hơn của bước ba và bước bốn, nằm nhằm đáp ứng các chức năng và giá trị của một ứng dụng bằng cách lưới hoá ứng dụng với điều kiện không có nhiều thay đổi phần đệm của lưới. Ứng dụng tương tự có thể được cấu trúc để chạy trên môi trường phi lưới. Ứng dụng sau khi đã đạt đến giai đoạn này đã hội đủ các điều kiện để có thể trở thành ứng dụng lưới.

+ Giai đoạn khai thác: ứng dụng đạt được bước thứ sáu sẽ khai thác lưới hay cơ sở hạ tầng các cụm máy tính cho mục đích thực hiện ứng dụng bởi vì ứng dụng được viết từ ban đầu với mục đích là thực hiện trên lưới. Các ứng dụng qua sáu bước không thể hoàn thành một cách chính xác và thành công nếu như không được chạy trên lưới.

Các bước này có liên quan đến nhau theo nghĩa là bước sau được phát triển trên cơ sở của bước trước nó. Ví dụ, một ứng dụng hiện thời có thể được lưới hoá theo cách mà một đơn vị tính toán hướng tài nguyên của nó được chạy như một công việc theo lô trên bất cứ tài nguyên tính toán nào trong lưới (bước một). Lưới sẽ quyết định việc thực thi công việc vào thời gian chạy ứng dụng. Sau đó, bằng việc loại bỏ các tác nhân cản trở việc thực thi đồng thời, ứng dụng đó có thể được coi như là một tập bao gồm các thể hiện công việc theo lô độc lập nhau được chạy đồng thời trên các nút lưới (Bước hai). Cách này làm cho toàn bộ ứng dụng chạy nhanh hơn và hiệu quả hơn, xử lý được nhiều đầu vào hơn là chỉ chạy duy nhất một thể hiện vào một thời điểm. Tiếp theo, ứng dụng có thể thực hiện tiếp thông qua một trong hai cách sau:

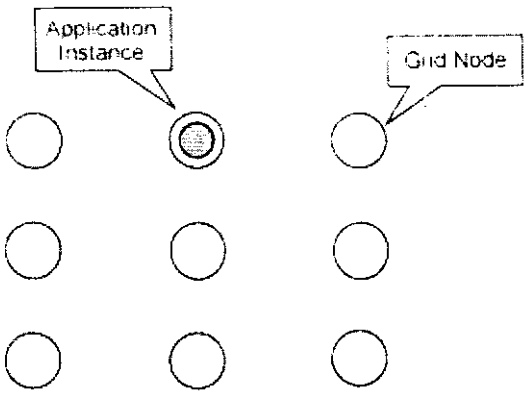
+ Tiếp tuc loại bỏ các tác nhân cản trở việc thực thi các công việc đồng thời và cho phép các đơn vị chương trình đã được lưới hoá chạy chạy song song (bước ba). Từ bước này, khái niệm ứng dụng chỉ mang tính chất đại diện mà thay vào đó là đơn vị công việc. Một đơn vị công việc có thể là một mảng các đơn vị công việc thuộc hai bước đầu tiên. Ngoài ra, trong bước này cần phải thêm vào hai thành phần: thành phần chia công việc, quản lý các công việc trên lưới và một bộ tập hợp các kết quả.

+ Ứng dụng có thể được thiết kế lại theo cách là thành phần công việc đã được lưới hoá không chạy như là tập các công việc song song mà được thực hiện như là một dịch vụ (bước bốn).

Cuối cùng, các phần của ứng dụng có thể được thiết kế lại ở mức cao hơn như là các dịch vụ có trạng thái, được gọi từ xa và thực hiện một cách song song (bước năm). Đối với một ứng dụng bình thường, tức là ứng dụng được thiết kế để thực hiện trên các máy đơn, bước thứ năm là đủ cho quá trình lưới hóa. bước thứ sáu thông thường là không đạt được cho các ứng dụng sẵn có bởi vì chi phí để thiết kế lại ứng dụng có thể cao hơn những lợi ích mà ta có thể thu lại. Do đó, tốt hơn là để có một ứng dụng lưới hoàn chỉnh, ta thiết kế các chương trình song song phụ thuộc chặt (tightly coupled) ngay từ lúc xây dựng ứng dụng.

8.1.4. Xử lý công việc theo lô (Bước 1)

Hình vẽ sau đây sẽ minh hoạ phần đệm lưới sẽ quyết định nút lưới nào thực hiện công việc. Một ví dụ cho ứng dụng là một truy vấn xem một số nguyên x đưa vào có phải là số nguyên tố hay không. Nhiều nút trong lưới có thể cùng một lúc đưa ra các truy vấn. Lưới sẽ trả lại kết quả chính xác cho máy đã gửi yêu cầu



Hình 8-2: Xử lý công việc theo lô

Trong bước này, mục đích là để có thể chạy một thể hiện của ứng dụng trên hầu hết bất cứ nút nào trên lưới và việc thực hiện này là do phần đệm lưới quyết định vào thời gian chạy công việc. Phần đệm có thể chọn một số nút khác vào thời điểm chạy tiếp theo. Ngoài ra, đối với bước đầu tiên, khía cạnh thực hiện công việc đồng thời là ở chỗ, các nút được thực hiện các công việc của các ứng dụng khác nhau tại cùng một thời điểm. Các đặc tính chủ yếu của bước xử lý công việc theo lô bao gồm các điểm sau:

Độc lập về nơi thực hiện công việc tức là một công việc bất kì được thực hiện tại một nút xác định và không ảnh hưởng tới các nút khác.

Việc thực thi công việc không quá phức tạp và khó quản lý.

Việc xác định bản quyền tức là nút nào chạy công việc nào là hoàn toàn thực hiện được.

Thời gian khởi động, kích hoạt ứng dụng là chấp nhận được.

Mức bảo mật tương xứng, thoả đáng, đầy đủ.

Sự lập lịch được thực hiện bởi lịch trình hay yêu cầu của công việc.

Đối với bước đầu tiên, chức năng nền tảng là một ứng dụng nhận các tham số đầu vào và sử dụng tệp hay cơ sở dữ liệu đã xác định trong các tham số dòng lệnh. Mục đích của bước này là chạy một thể hiện của ứng dụng trên một tập bất kì các nút lưới và các nút thực hiện công việc được quyết định bởi phần đệm của lưới vào thời gian chạy. Trong lần chạy tiếp theo, các nút thực hiện có thể không giống với các nút đã thực hiện ở lần chạy trước, điều này phụ thuộc vào phần đệm. Khía cạnh chạy đồng thời trong trường hợp này là nói đến các nút được chọn để thực hiện công việc và các nút khác có thể đồng thời phục vụ cho các ứng dụng khác nhau.

Trên quan điểm người xây dựng ứng dụng lưới, một số vấn đề quan trọng cần được xử lý trong bước thứ nhất:

- Chứng thực thực hiện trên các nút lưới: Ứng dụng phải được chứng thực để có thể được thực hiện trên một nút lưới xác định. Nếu như ứng dụng đã qua chứng thực thì các yêu cầu ràng buộc tài nguyên và các điều kiện thực hiện phải được phù hợp với nhau. Trừ khi có yêu cầu người sử dụng phải chứng thực phần mềm của họ trên tất cả các nút lưới, còn lại ứng

dụng phải được áp dụng cơ chế quản lý khoá nút. Với bước này, ta phải thực hiện ở mức tối thiểu là mô hình chứng thực "bất cứ một nút nào tại một thời điểm" (any one node at a time) tức là phải đảm bảo là tại bất cứ thời điểm nào trong thời gian ứng dụng đang được thực thi trên lưới phải tồn tại một nút sẵn sàng và cho phép ứng dụng được chạy trên nó. Năm mô hình của chuẩn quản lý sử dụng phần mềm (XSLM – X-Open Software License Use Management) có thể được trình bày vắn tắt bao gồm:

+ Cơ bản: Chứng thực theo tính chất của ứng dụng hay theo vị trí thực hiện ứng dụng.

+ Theo tên: Chứng thực dựa trên người đã đăng kí dùng hay các máy có tên xác định. Đây chính là mô hình chứng thực khoá nút.

+ Đồng thời: Chứng thực được giới hạn sử dụng bởi một số lượng các người dùng đồng thời.

+ Tích luỹ hay hao tổn: Chứng thực dựa trên thực tế sử dụng (khối lượng hay thời gian thực hiện công việc).

+ Khả năng: Chứng thực được giới hạn trong việc sử dụng tài nguyên ít hơn khả năng tính toán cho phép nhất định. Về vấn đề này, người phát triển ứng dụng có thể quản lý chứng thực theo nhiều cách:

Mang tính chất thương mại hơn là mang tính chất kỹ thuật, với sự thoả thuận của người sử dụng cho các điều kiện chứng thực.

Có các vết kiểm tra: có cơ chế bản ghi sử dụng để theo dõi và quản lý quá trình thực hiện công việc hay ứng dụng trên các nút theo yêu cầu của người sử dụng.

Có thể tận dụng dịch vụ quản lý chứng thực của phần đệm lưới, như là Global License Broker của Platform Computing.

Sử dụng các phần mềm quản lý chứng thực như License Power của Isogon, License Usage Manager của IBM.

Tự thực hiện quá trình quản lý. Đây là phương pháp khó thực hiện nhất và thường phục vụ cho các yêu cầu đặc biệt.

+ Bảo mật: Ngay từ bước đầu tiên này, vấn đề bảo mật đã được quan tâm. Một cách lý tưởng là ta cung cấp cơ chế bảo mật ở ngoài phạm vi ứng dụng. Việc quyết định người dùng nào được phép thực hiện ứng dụng nên là một phần của môi trường lưới. Một người sử dụng không có mã số chứng thực hợp lệ không có quyền thực hiện ứng dụng.

+ Sự tương đồng giữa các nút: Giả thiết rằng cùng một một công việc sẽ được thực hiện trên các nút khác nhau tại các lần chạy khác nhau. Do đó, cần phải xác định các tệp cần thiết phải luôn trong trạng thái khả dụng, tệp nào sẽ được tạo ra và tệp nào có thể bị thay đổi. Việc xác định này giúp cho người sử dụng hiểu rõ hơn về quá trình thực hiện công việc cũng như có sự cung cấp cần thiết cho lưới.

+ Quản lý công việc: Ứng dụng có thể được thực hiện dưới sự quản lý của bất kì hệ thống quản lý công việc nào. Quá trình thực hiện công việc không thể do người sử dụng đảm

nhận. Nhiệm vụ của người sử dụng là cung cấp các tham số đầu vào cần thiết cho quá trình thực hiện ứng dụng. Sau đó chờ đợi để nhận kết quả trả về.

+ Truyền thông trong mạng cục bộ (LAN – Local Area Network): Nếu ứng dụng được thực hiện trên các nút mà các nút này được kết nối hoàn chỉnh trong cùng một mạng con và chỉ sử dụng các tài nguyên nội bộ thì vấn đề truyền thông trong mạng cũng phải được quan tâm. Trong đó, phải chú ý các hỗ trợ đầu vào cho ứng dụng và các tệp yêu cầu cho các nút trước khi thực thi công việc cũng như việc truyền các tệp kết quả tới nơi yêu cầu sau khi thực hiện xong công việc trên một nút nào đó. Ngoài ra còn một số điểm cần lưu ý như sử dụng cở sở dữ liệu trên các nút đang bận, truyền thông điệp giữa các nút, hệ thống giao dịch trong lưới... Tất nhiên, phải luôn nhớ rằng các ứng dụng khác trên cùng nút hay trên các nút khác cũng đang sử dụng hệ thống mạng con.

+ Truyền thông trong mạng diện rộng (WAN – Wide Area Network): Trong mạng diện rộng, ứng dụng có thể được thực hiện trên nhiều nút thuộc các mạng con khác nhau và vấn đề truyền thông trở nên phức tạp hơn nhiều. Lúc đó, các tệp làm đầu vào cho ứng dụng thực hiện sẽ được lưu trữ ở các địa điểm khác nhau, thuộc các mạng con khác nhau. Ngoài ra, sau khi thực hiện xong công việc, các kết quả phải được trả về một địa điểm xác định nào đó. Lúc này, các kỹ thuật truy cập từ xa như NFS (Network File System), truyền tệp... sẽ được triển khai nếu như tần suất truy cập và số lượng dữ liệu có thể được truyền tải không vượt quá thời gian thực hiện công việc của ứng dụng. Ví dụ như khi tất cả các công việc cần một phần nhỏ của một tệp dung lượng rất lớn mà không cần sao chép hay truyền tải cả tệp đó tới nơi thực hiện công việc.

+ Sử dụng cơ sở dữ liệu tại các địa điểm khác nhau: Cùng với vấn đề truyền thông giữa các nút trên các mạng con khác nhau, còn một vấn đề lớn cần quan tâm là truy cập cơ sở dữ liệu và các tài nguyên giao dịch khác tại một hay nhiều địa điểm. Khi một hệ quản trị cơ sở dữ liệu (DBMS – Database Management System) có nhiều máy khách cùng truy vấn, vấn đề là làm sao giảm thiểu thời gian trễ giữa các lần truy cập. Để cho thời gian trễ giảm bớt đi và quản lý bảo mật tốt hơn, hệ quản trị cơ sở dữ liệu có thể cho phép ứng dụng thực hiện các thao tác cập nhật theo các thủ tục có sẵn chứ không cho phép cập nhật trực tiếp vào tài nguyên dữ liệu. Hơn thế nữa, khi dữ liệu cần được tích hợp từ các địa điểm khác nhau, các công nghệ kết tập dữ liệu hiện nay như DB2 Infomation Integrator có thể giảm bớt lưu lượng truyền thông trên mạng diện rộng cũng như mã lệnh của máy khách.

+ Một số vấn đề khác: Các điểm sau đây mang tính chất lựa chọn và không bắt buộc đối với bước thứ nhất. Tuy nhiên, các ý sau góp phần thực hiện tốt hơn cho bước đầu tiên:

Ứng dụng nên mở rộng khả năng được điều khiển thông qua các tham số do người sử dụng đưa vào. Ví dụ như số lượng bộ nhớ, số lượng bộ vi xử lý tối đa cần thiết để thực hiện công việc...

Có thể cho phép ứng dụng được dừng và trả về kết quả tạm thời mà không gây ra vấn đề nghiêm trọng đối với hệ thống hay các nút lưới.

Nếu có thể, ứng dụng nên thực hiện các điểm kiểm tra (checkpoint) theo chu kì nhất định. Nếu được như vậy, khi ứng dụng được thực hiện trở lại sau một sự cố dừng chương trình thì nó có thể bắt đầu ngay các công việc tiếp theo mà không phải thực hiện lại từ đầu. Các tham số về chu kì thực hiện tác vụ kiểm tra có thể do người sử dụng cung cấp như sau mỗi n bản ghi đầu vào hay sau m phút...

Nên cung cấp các thông tin tiêu thụ tài nguyên cần thiết cho ứng dụng trong kế hoạch triển khai như thông tin yêu cầu tài nguyên bộ nhớ, dạng thức vào ra, yêu cầu về CPU... Các yêu cầu này phải được đĩên tả chính xác, dễ hiểu và theo một chuẩn công nghiệp cho các chuyên việc triển khai ứng dụng. Các đặc tả về yêu cầu này sau đó sẽ được dịch và thực thi bởi phần đệm lưới.

Giữ tính độc lập đối với các phần đệm lưới: các phần mềm thương mại hiện này đều không thể giải quyết hết tất cả các vấn đề trong môi trường tính toán lưới. Thay vào đó, phần đệm lưới chỉ được kiểm chứng thông qua các ứng dụng được thực thi trên đó. Chính vì vậy, ứng dụng phải được thiết kế theo các chuẩn công nghiệp được sử dụng rông rãi, sử dụng các công nghệ được hỗ trợ mạnh và nếu có thể sử dụng các nền tảng mã nguồn mở và không nên phụ thuộc chặt chẽ vào một phần đệm lưới nhất định.

8.1.5. Xử lý theo lô đồng thời (Bước hai)

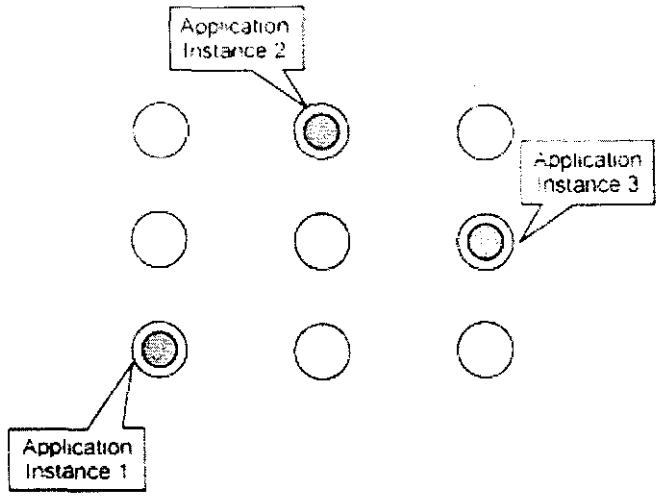
Bước thứ hai trong quá trình lưới hoá một ứng dụng hỗ trợ quá trình thực hiện đồng thời các thể hiện độc lập của một ứng dụng. Ngoài các đặc tính của bước đầu tiên, bước này còn bao gồm các điểm sau:

Nhiều thể hiện chạy một cách độc lập mà không có liên hệ với nhau.

Các công việc độc lập là tương đồng với nhau. Ví dụ như Công việc X cho kế toán A có thể chạy đồng thời với công việc X cho kế toán B.

Cơ sở dữ liệu và các tài nguyên khác không tồn tại các thời điểm truy cập tức thời (hot spot) hay xảy ra tình trạng nghẽn (deadlock).

Hình vẽ mô tả thể hiện của ứng dụng đang chạy trên lưới.



Hình 8-3: Xử lý theo lô đồng thời

Các vấn đề cần phải xem xét khi chia công việc theo lô đồng thời bao gồm:

- Chứng thực: nếu như ứng dụng đã được chứng thực, các mô hình chứng thực và công nghệ chứng thực phải cho phép và đảm bảo sự đồng nhất các yêu cầu về chứng thực. Bên cạnh các công nghệ như đã trình bày ở bước đầu tiên còn có mô hình chứng thực dựa trên sự thoả thuận. Các tài nguyên có thể thoả thuận bao gồm máy tính, bộ vi xử lý hay thể hiện của dịch vụ.
- Không có sự can thiệp: trong bước thứ hai, nhiều thể hiện của dịch vụ được thực hiện đồng thời, do đó điều quan trọng là các thể hiện này không can thiệp nhau. Hệ thống lưới phải đảm bảo thể hiện đầu tiên được kích hoạt có thể không phải là thể hiện đầu tiên thực hiện xong. Bên cạnh đó, đầu ra hay các cập nhật của thể hiện này không được xung đột với đầu ra hay các cập nhật của thể hiện khác.
- Các điểm truy cập tức thời (hotspot): một cơ sở dữ liệu được thiết kế phục vụ cho nhiều thể hiện cùng cập nhật một hàng có thể gây ra các lỗi không mong muốn. Do đó, ứng dụng phải thoả mãn:

+ Mỗi thể hiện chỉ được cập nhật một số hàng xác định và khác nhau đối với cùng một tài nguyên lưu trữ trong một thời điểm.

+ Các thể hiện cập nhật các hàng giống nhau tại các thời điểm khác nhau và thời gian trễ phải đủ nhỏ để không gây ra hiện tượng "tắc nghẽn".

8.1.6. Xử lý theo lô song song (Bước 3)

Bước này nhận mỗi công việc theo lô của người sử dụng, chia nhỏ công việc ra, phân các công việc nhỏ hơn cho các nút tính toán sau đó thu thập lại kết quả và tổng hợp kết quả để trả lại cho người sử dụng.

Với bước này, các khái niệm nút khách và nút chủ bắt đầu được sử dụng. Mô hình nút khách/nút chủ không được dùng để áp dụng cho các công việc chạy theo lô trong hai bước đầu tiên. Trong hai bước đầu tiên, với các thể hiện đơn và độc lập với nhau, một nút sẽ gửi

yêu cầu thực hiện công việc. Công việc này được gửi tới bộ lập lịch và bộ lập lịch sẽ đệ trình công việc tới phần đệm của lưới để thực hiện công việc. Trong khi đó, trong bước thứ ba, nút khách là máy gửi yêu cầu đồng thời chia công việc, đệ trình công việc và cuối cùng là tập hợp kết quả trả về. Một trong những đặc tính khác biệt trong bước này là nút đệ trình công việc có khả năng kết tập các kết quả trả về. Tất nhiên, về mặt vật lý thì máy khách chia công việc và máy kết tập kết quả trả về có thể là các máy tính khác nhau. Còn về mặt logic thì chúng cùng đảm nhận vai trò của nút khách, không trực tiếp thực thi công việc tính toán nghiệp vụ đơn thuần.

Các đặc tính chính của bước thứ ba bao gồm:

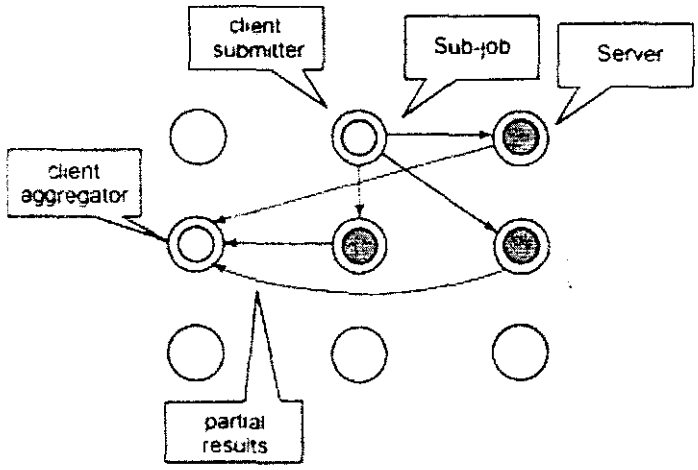
Nút khách làm nhiệm vụ chia nhỏ công việc chính thành các công việc song song và gửi các công việc thực hiện tới các nút lưới.

Chức năng của ứng dụng có thể được phân ra thành một nút khách và nhiều nút chủ thực hiện công việc.

Các công việc thực hiện với tư cách là các thể hiện độc lập.

Sau khi thu thập các kết quả trả về từ máy chủ, máy khách có thể nhóm lại để cho ra kết quả cuối cùng.

Hình vẽ mô tả quá trình chi nhỏ các công việc con giữa các nút lưới :



Hình 8-4: Xử lý theo lô song song

Như vậy, sau khi áp dụng bước thứ hai là xử lý theo lô đồng thời, nhiều thể hiện của một chương trình cùng được chạy và trạng thái của các thể hiện này phụ thuộc vào từng người sử dụng khác nhau. Với bước thứ ba là chạy xử lý theo lô song song, một công việc theo lô được chia nhỏ hơn nữa. Các công việc con này được thực hiện đồng thời và có trạng thái tuỳ thuộc vào người sử dụng đệ trình công việc con. Bước thứ ba hỗ trợ các tính năng sau:

+ Một chức năng của chương trình được chia nhỏ và do một nút khách đại diện, đi theo nó là nhiều nút chủ chủ phục vụ.

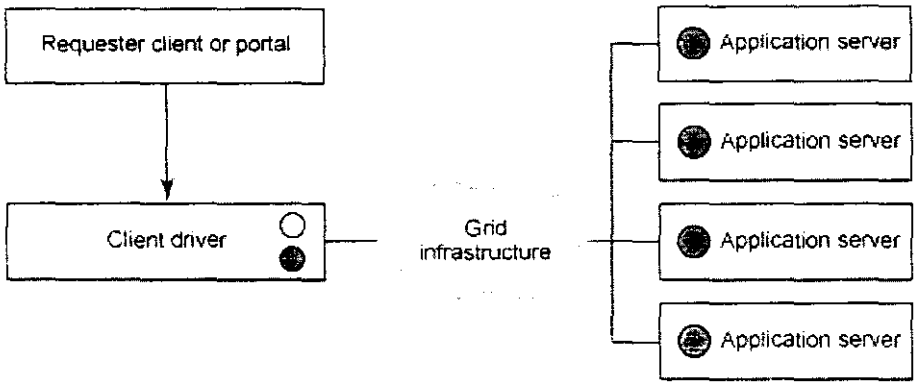
+ Nút chủ thực hiện công việc có thể chia một công việc thành các công việc con và gửi tới các nút khác để thực hiện.

+ Lưới tính toán sẽ phân bố các công việc giữa các nút.

+ Các nút chủ đang thực hiện trên các nút lưới đóng vai trò là các thể hiện độc lập nhau.

+ Sau khi thu thập kết quả trả về từ các nút chủ, nút khách có thể kết tập các kết quả này lại và trả về cho người sử dụng.

Như vậy, trong bước thứ ba này đã có sự phân hoá thành các nút với các chức năng khác nhau. Nút khách đóng vai trò là nút đệ trình công việc đến các nút thực hiện hay còn gọi là nút chủ. Sau khi các nút chủ thực hiện xong sẽ gửi trả lần lượt kết quả cho nút kết tập. Nút kết tập công việc làm nhiệm vụ đợi các kết quả từng phần công việc trả về sau đó tập hợp lại và trả lại kết quả cuối cùng, hoàn chỉnh cho người sử dụng. Mặt khác, trong bước này, chức năng chính là việc một chương trình nhận các tham số dòng lệnh và sử dụng các tệp hay cơ sở dữ liệu đã xác định từ tham số đưa vào. Nếu như sử dụng thuật ngữ nút khách và nút chủ thì chương trình đóng vai trò là ứng dụng được thực hiện trên các nút chủ và do các nút khách đệ trình. Hình vẽ dưới đây minh hoạ quan điểm trên:



Hình 8-5: Máy khách và máy chủ trong môi trường lưới

Phần đệm lưới trong bước thứ ba sẽ thực hiện các công việc sau:

Chuyển các đơn vị chương trình tới các nút để thực hiện.

Đảm bảo tất cả các đơn vị chương trình này được thực hiện thành công.

Đơn giản hoá việc kết tập các kết quả từ các nút thực hiện. Phần đệm sẽ chuyển các kết quả đến một nút trung tâm làm nhiệm vụ tập hợp kết quả hay đơn giản hơn là đảm bảo sao cho các kết quả trả về có thể dễ dàng được truy cập.

Xét theo phương diện người phát triển ứng dụng lưới thì các công việc cần tập trung bao gồm:

+ Sự chia nhỏ công việc: Nút khách phải chia nhỏ các công việc và chuyển các phần công việc này tới nút chủ thông qua phần đệm lưới để xử lý.

+ Xử lý từng phần: mỗi một thể hiện của ứng dụng trên nút chủ phải có khả năng xử lý một phần công việc khác nhau.

+ Tính độc lập và sự ảnh hưởng lẫn nhau: các công việc con phải độc lập với nhau và không xung đột với nhau trong quá trình thực hiện.

+ Kết tập kết quả: nút khách phải có khả năng kết tập các kết quả từ các phần kết quả của các công việc con.

8.1.7. Dịch vụ (Bước bốn)

Từ bước này trở đi, khái niệm về công việc theo lô không còn, thay vào đó là dịch vụ trên lưới để thực hiện các công việc. Bước này tập trung vào quá trình chuyển hoá từ một công việc chạy theo lô sang kiến trúc hướng dịch vụ. Bước này có thể được thực hiện ngay sau khi đã hoàn thành bước thứ hai mà không cần phải qua bước thứ ba. Tức là trước khi thực hiện bước này, mỗi máy khách phải chia nhỏ công việc và đệ trình tới các nút tính toán trên lưới.

Các đặc tính chủ yếu của bước này là:

Máy khách sử dụng phần đệm của lưới để kích hoạt dịch vụ.

Trên quan điểm máy khách, phần đệm của lưới sẽ gọi dịch vụ. Dịch vụ được cấu trúc để phục vụ cho các lời gọi từ xa và được tổ chức như là một lớp, thư viện tiến trình con hay thư viện liên kết động (DLL-Dynamic Link Lybrary).

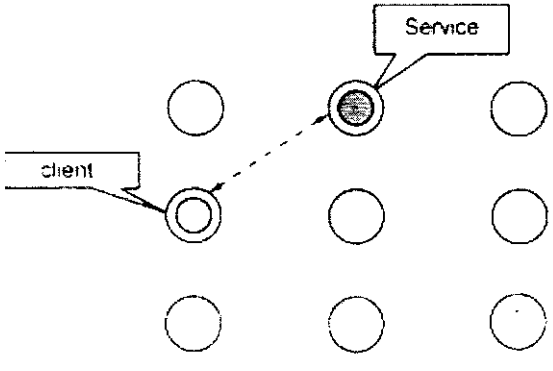
Máy khách và máy chủ được thiết là phụ thuộc lỏng (loosely coupled). Tức là việc phát triển các chức năng của dịch vụ, thay đổi dịch vụ không ảnh hưởng nhiều tới quá trình triệu gọi dịch vụ từ phía máy khách. Bởi lẽ, đối với máy khách thì phần đệm lưới mới trực tiếp gọi dịch vụ. Thông thường thì phần đệm lưới đảm nhận hết các thao tác phức tạp trong quá trình tương tác giữa máy khách và máy chủ.

Dịch vụ có thể lưu giữ trạng thái của nó giữa các lần gọi.

Dịch vụ có khả năng chia sẻ giữa các máy khách độc lập.

Sự đánh giá hiệu năng của lưới không bao gồm thời gian khởi động dịch vụ. Bởi vì dịch vụ được thiết kế phục vụ mục đích phục vụ liên tục và trong một lần chạy chương trình, dịch vụ có thể được gọi nhiều lần và thường được khởi động trước khi có yêu cầu thực hiện.

Hình vẽ mô tả một máy khách kích hoạt dịch vụ :



Hình 8-6: Dịch vụ

Bước thứ tư này mở rộng từ bước thứ hai là chạy công việc theo lô độc lập đồng thời, bên cạnh tính song song hoá. Ngoài ra, bước này cũng có một số tính chất tương tự như bước

thứ ba là chạy song song các công việc theo lô. Như vậy, trong quá trình lưới hoá một ứng dụng, có thể chọn một trong hai bước ba hoặc bốn mà không phải thực hiện lần lượt cả hai bước này. Mục đích của bước này là làm sao để chương trình chủ có thể phục vụ cho một số dạng thức gọi từ xa. Máy khách sẽ thông qua phần đệm lưới để gọi dịch vụ. Khi đó phần đệm sẽ làm nhiệm vụ kích hoạt dịch vụ sau khi nạp dịch vụ vào bộ nhớ. Từ chiến thuật này trở đi, khái niệm lưới càng được cụ thể hoá bởi vì hệ thống có thể thực hiện cùng một lúc các dịch vụ khác nhau trên các nút lưới được lựa chọn bởi phần đệm lưới. Khái niệm dịch vụ cũng được thể hiện rõ nét bởi vì dịch vụ được kích hoạt với số lần không hạn chế mà không phải thông qua cơ chế lập lịch theo lô, quá trình khởi động chương trình... Ví dụ, trong trường hợp một chương trình chủ phải mất vài giây để kiểm tra tính toàn vẹn và sau đó xây dựng vài gigabytes cấu trúc dữ liệu trong bộ nhớ từ các dữ liệu đầu vào trước khi thực hiện chức năng chính. Quy trình này phải được thực hiện tuần tự mỗi khi có yêu cầu gửi đến chương trình. Tuy nhiên, vì ứng dụng đã được dịch vụ hoá nên mỗi một thể hiện của dịch vụ có thể đáp ứng nhiều đòi hỏi một lúc. Như vậy thì thời gian chờ đợi của máy khách sẽ giảm xuống đáng kể.

Để có thể dịch vụ hoá ứng dụng, chương trình phải có một hay nhiều khả năng đáp ứng các lời gọi hàm từ xa. Đồng thời đầu vào, đầu ra phải được xác định từ tham số dòng lệnh. Hiện nay, một dịch vụ được định nghĩa theo chuẩn GWSDL với phần mở rộng cho lưới. Trong tệp này sẽ mô tả các thao tác có thể được gọi từ xa hay nói cách khác là dịch vụ sẽ hỗ trợ các hàm nào cho máy khách. Cơ chế gọi hàm cũng như ánh xạ từ tệp GWSDL đến dịch vụ thực hiện phải được phần đệm lưới đảm nhận. Sau khi dịch vụ được nạp vào bộ nhớ, các hàm mang tính chất công cộng (public) phải thoả mãn tính chất được sử dụng lại. Nếu không có tính chất này thì phần đệm lưới phải thực hiện quy trình giải phóng và nạp lại dịch vụ giữa mỗi lần sử dụng và dịch vụ phải thực hiện các thao tác khởi động cũng như huỷ các kết quả trung gian sau mỗi lần được kích hoạt. Trong trường hợp tối thiểu, dịch vụ phải có tính sử dụng lại tuần tự. Kiểu sử dụng lại này có nghĩa là phần đệm lưới có thể cho thực hiện lần lượt các yêu cầu của máy khách trên các chức năng của dịch vụ. Như vậy, mỗi lần được kích hoạt thì dịch vụ có thể đồng thời phục vụ nhiều chức năng khác nhau cho các máy khách.

Một số vấn đề cần quan tâm trong bước thứ tư:

- Lưu trạng thái dịch vụ: cũng giống như trong bước thứ hai, có thể xảy ra trường hợp dịch vụ được nhiều truy vấn đòi hỏi cùng một thời điểm. Trong tình huống này, phần đệm phải hỗ trợ khả năng đồng thời xử lý nhiều yêu cầu từ phía máy khách tới một dịch vụ nào đó. Tất cả các yêu cầu phải thoả mãn trong bước này cũng giống như trong bước thứ hai. Bên cạnh đó, nếu dịch vụ có lưu giữ lại thông tin người dùng hay thông tin trạng thái giữa các lần gọi thì phải đáp ứng các yêu cầu sau:

+ Phải thực hiện trong một môi trường hỗ trợ tính năng lưu giữ trạng thái, ví dụ như OGSA. Trong môi trường này, mỗi yêu cầu tới một dịch vụ nhất định sẽ được chuyển tới các thể hiện của dịch vụ.

+ Lưu giữ các thông tin trạng thái vào một vị trí nào đó như tệp, cơ sở dữ liệu... sao cho các thể hiện khác của cùng một dịch vụ có thể dễ dàng truy cập. Ngoài ra, mỗi một thể hiện của dịch vụ phải xác định được thông tin của chính thể hiện đó trong một lần gọi nhất định chứ không phải thông tin của thể hiện khác của cùng một dịch vụ.

- Đa người dùng: bước thứ tư cho phép dịch vụ có tính chất đa người dùng. Phụ thuộc vào phần đệm lưới có hỗ trợ tính năng đa người dùng có thể cùng truy chia sẻ một thể hiện của dịch vụ. Nếu như phần đệm lưới không hỗ trợ tính năng này thì một thể hiện chỉ có thể phục vụ cho một người dùng và người dùng chỉ có thể sử dụng các thể hiện khác nhau của các dịch vụ khác nhau tại cùng một thời điểm. Một vấn đề nữa cũng cần được quan tâm là tính đồng bộ hoá khi một thể hiện phục vụ cho nhiều người sử dụng. Nếu như tại cùng một thời điểm chỉ có một yêu cầu của người sử dụng đối với một thể hiện được đáp ứng thì dịch vụ hỗ trợ tính năng sử dụng lại tuần tự là đủ. Ngoài ra, dịch vụ phải được thực thi theo kiểu đa luồng, tức là cùng một thời điểm, một thể hiện có thể đáp ứng nhiều lời yêu cầu khác nhau và mỗi một yêu cầu sẽ tương ứng với một luồng công việc.

- Chứng thực: trong bước này có thể áp dụng mô hình chứng thực theo lần sử dụng. Dịch vụ sẽ quyết định đơn vị cho lần sử dụng như một lần kích hoạt dịch vụ hay một tập số lần gọi dịch vụ có liên quan từ người sử dụng.

- Liên kết lỏng giữa chương trình máy khách và dịch vụ: máy khách thông qua phần đệm lưới sẽ tìm, kích hoạt và kết nối với dịch vụ máy chủ theo các kênh bảo mật. Trong trường hợp này, phần đệm lưới đóng vai trò là nhân tố tạo nên sự liên kết lỏng giữa máy khách và dịch vụ. Tức là việc liên kết sẽ không phụ thuộc hoàn toàn vào một phần đệm xác định nào mà chỉ tuỳ vào phương thức kết nối và cách thức giao tiếp giữa máy khách và dịch vụ. Phần đệm đóng vai trò hỗ trợ các giao tiếp và thủ tục gọi dịch vụ từ máy khách. Nếu như máy khách không thông qua phần đệm để liên kết tới dịch vụ thì chi phí và thời gian sẽ tăng lên đáng kể và băng thông truyền tải cũng bị chiếm dụng nhiều hơn. Nếu thông qua phần đệm thì nhiều yêu cầu có thể được nhóm lại và gửi tới một dịch vụ dãy để xử lý các yêu cầu này. Chính vì vậy mà thời gian kết nối, giá cả và dung lượng truyền trên mạng được giảm đi nhiều.

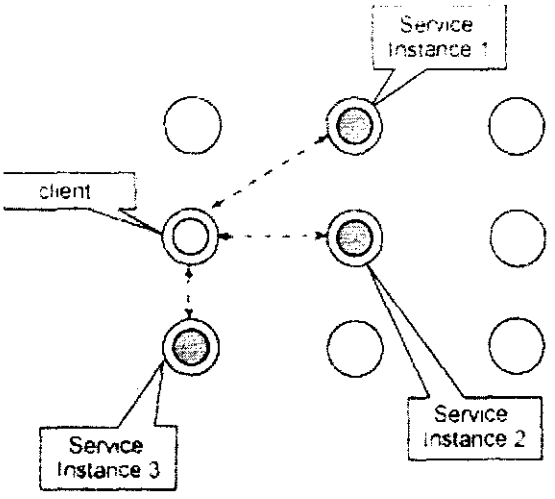
8.1.8. Dịch vụ song song (Bước năm)

Bước này kết hợp kiến trúc hướng dịch vụ của bước thứ tư và bước thứ ba là mô hình chia nhỏ công việc thành các công việc theo lô chạy song song. Đặc tính chủ yếu của bước này còn có thêm:

Cung cấp nhiều các thể hiện dịch vụ một lúc

Cho phép máy khách kích hoạt các thể hiện này song song

Hình vẽ minh họa việc gọi đồng thời các thể hiện dịch vụ song song.



Hình 8-7: Dịch vụ song song

Như vậy, so với bước trước thì bước thứ năm được nâng cấp hơn và thể hiện rõ nét tính song song của dịch vụ. Một nút khách có thể đồng thời đưa ra nhiều lời gọi tới cùng một dịch vụ. Các lời gọi này được các thể hiện của dịch vụ xử lý. Mô hình này còn có một ưu điểm nữa là nếu như dịch vụ cần phải nâng cấp thì chỉ cần chỉnh duy nhất dịch vụ ban đầu, còn lại các thể hiện của dịch vụ sẽ được cập nhật đồng thời.

Khi dịch vụ đạt được đến mức bước này, điểm khác biệt lớn nhất so với bước trước là nhiều thể hiện của một dịch vụ có thể cùng được tạo ra cùng một lúc và có thể phục vụ một cách song song các người dùng khác nhau. Bước này tương tự như bước thứ ba là xử lý theo lô song song. Trong bước này, nút khách sẽ nhận các yêu cầu sử dụng dịch vụ, chia nhỏ các yêu cầu này và chuyển tới phần đệm lưới, sau đó thì các thể hiện của dịch vụ sẽ được kích hoạt để trực tiếp thực thi. Như vậy, với bước thứ năm:

Mỗi thể hiện của nút khách sẽ sử dụng nhiều thể hiện của dịch vụ một cách song song.

Các thể hiện của dịch vụ không giao tiếp với nhau. Dạng thức này còn có tên là "song song có biên giới" (embarrasingly parallel) tương tự như việc chia các công việc thành công việc con như trong bước thứ ba.

Mức độ truy vấn dịch vụ là có giới hạn. Chương trình nút khách là có sự phụ thuộc với nút chủ.

Chiến thuật thứ năm thực chất là hợp của hai bước thứ ba và thứ tư. Như đối với bước thứ ba, bước thứ năm kiểm soát song song các công việc đã được chia nhỏ tuy nhiên các công việc con này chính là các thể hiện dịch vụ như trong bước thứ tư. Do bước này kết hợp cả hai bước thứ tư và năm nên nó có tất cả các ưu điểm của hai bước trên. Thời gian phản hồi từ dịch vụ được cải thiện đáng kể bởi vì dịch vụ được thực thi theo dạng thức song song và quá trình khởi động dịch vụ coi như là mặc định trước lúc nút khách yêu cầu dịch vụ. Một lợi

thế khác nữa là nút khách có thể tận dụng được dịch vụ theo nhiều cách, ví dụ như: dạng thức song song dựa trên trạng thái. Kỹ thuật "song song có biên giới" trong bước thứ ba dựa trên quá trình chuyển các tham số khác nhau tới các thể hiện khác nhau đối với cùng một công việc đã được chia nhỏ. Cách làm này còn được gọi là dạng thức song song hướng tham số "parametric parallelism". Bước thứ năm còn hỗ trợ một dạng thức song song khác là song song theo trạng thái chứ không chỉ theo tham số. Trong dạng thức này, mỗi thể hiện có các thông tin trạng thái khác nhau. Dạng thức bao gồm các kỹ thuật sau:

1. Multicast: với kỹ thuật "multicast", nút khách kích hoạt tất cả các thể hiện chương trình chủ với cùng một tham số đầu vào. Tuy nhiên, các nút khách sẽ xử lý các tham số này dựa trên trạng thái nội tại và các thông tin mà nó đã lưu giữ. Do đó, trạng thái của mỗi thể hiện là khác nhau và kết quả trả về cũng không giống nhau.

2. Ta xem xét một ví dụ về lý thuyết số: tìm các số nguyên tố. Giả sử đã có 100 thể hiện dịch vụ chủ, mỗi thể hiện dịch vụ sẽ kiểm tra xem các số đầu vào có tính chất bị chia bởi một số các số nguyên tố kiểm tra trong danh sách của nó hay không. Thể hiện dịch vụ đầu tiên chứa danh sách các số nguyên tố kiểm tra từ 1 đến 1,000,000. Tương tự, thể hiện thứ hai chứa danh sách các số nguyên tố kiểm tra từ 1,000,001 đến 2,000,000... Nếu như một số rất lớn được gửi tới toàn bộ các thể hiện cùng một lúc (multicast), nó có thể được kiểm tra xem là số nguyên đó có chia được cho một số nguyên tố nhỏ hơn 100,000,000 hay không trong khoảng thời gian bằng một phần trăm lần nếu như chỉ có một thể hiện của dịch vụ tương đương thực hiện công việc. Trong trường hợp này, cũng cần chú ý tới yếu tố trạng thái của dịch vụ. Do các thể hiện của dịch vụ đều có trạng thái, tức là một dãy các số nguyên tố kiểm tra nên không mất thời gian nạp và tính toán lại dãy số này mỗi khi có yêu cầu xử lý gửi tới.

3. Eachcast: chương trình khách có thể gửi một tập các tham số xác định, có thể khác nhau tới mỗi thể hiện của dịch vụ thông qua phần đệm lưới. Trong ví dụ trên, đối với người sử dụng có thể yêu cầu kiểm tra cùng một lúc 100 số nguyên thuộc khoảng xử lý của 100 thể hiện dịch vụ.

4. Anycast: chương trình khách có thể qua phần đệm lưới có thể gửi một số lần các truy vấn sử dụng dịch vụ lớn hơn số thể hiện của dịch vụ. Kỹ thuật này đặc biệt thích hợp khi mà tốc độ thực hiện của các thể hiện dịch vụ trên các nút là khác nhau. Ví dụ như trong trường hợp người sử dụng muốn biết một số nguyên lớn hơn 1,00,000,000 có phải là số nguyên tố hay không. Chương trình khách sẽ chia phần số nguyên lớn hơn 100,000,000 thành 100 phần bằng nhau và áp dụng kỹ thuật "eachcast" cho mỗi thể hiện dịch vụ chủ. Tuy nhiên, nếu như tốc độ thực hiện trên các nút không như nhau thì không nhất thiết phải chia thành các phần công việc bằng nhau. Chương trình khách có thể chia thành 300 phần công việc, sau đó nhờ phần đệm lưới áp dụng kỹ thuật "eachcast" tới các nút sao cho tận dụng tối đa năng lực xử lý của các nút.

5. Unicast: trong các kỹ thuật trên, mỗi khi có yêu cầu sử dụng dịch vụ thì tất cả các thể hiện dịch vụ đều được kích hoạt. Còn đối với mô hình này, chương trình khách có thể chỉ sử dụng một thể hiện dịch vụ trong nhiều thể hiện. Trong ví dụ đã trình bày, nếu như số nguyên cần kiểm tra nhỏ hơn 100,000,000 thì chỉ cần một thể hiện của dịch vụ có dãy số nguyên tố kiểm tra có chứa số đầu vào xử lý.

8.1.9. Chương trình song song phụ thuộc chặt (Bước sáu)

Các chương trình song song phụ thuộc chặt thường được dùng để giải quyết các bài toán chuyên biệt trong các ngành như thiết kế, vật lý, mô phỏng sinh học.

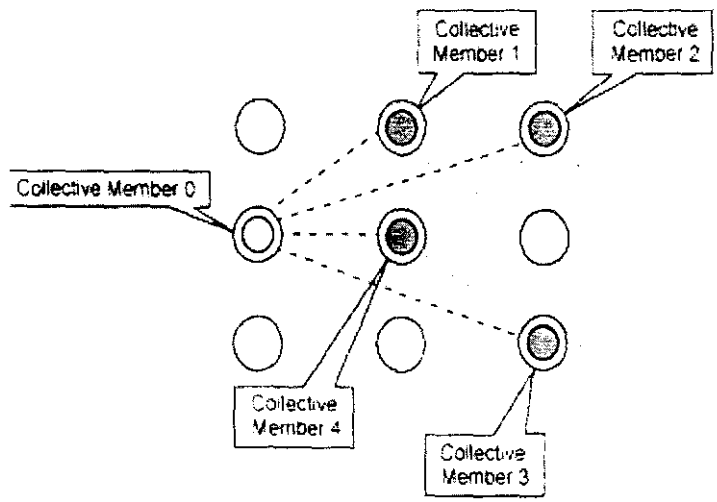
Đặc tính chủ yếu của bước này là nó cung cấp cơ chế truyền thông và đồng bộ hoá việc thực hiện giữa máy khách, máy chủ và giữa các dịch vụ. Nghĩa là cả máy khách, máy chủ, phần đệm lưới là phụ thuộc chặt chẽ với nhau. Bất cứ sự thay đổi nào ở một trong ba phía đều kéo theo sự thay đổi các phần còn lại. Chính vì vậy, các ứng dụng thuộc loại này thường được thiết kế theo hướng tính toán lưới ngay từ đầu.

Đặc tính nổi trội của bước này là:

Cung cấp hệ thống tính toán cực mạnh.

Hỗ trợ các ứng dụng mà thường không phù hợp khi thực hiện trên một nút hay là đòi hỏi thời gian tính toán khổng lồ.

Hình vẽ mô tả cách thức các chương trình tương tác và phụ thuộc chặt trở thành bước cuối cùng :



Hình 8-8: Các chương trình song song phụ thuộc chặt

Tóm lại, sáu bước trong quá trình lưới hoá một ứng dụng đã có có thể được tóm tắt qua bảng sau:

Bước 1	xử lý theo lô	Ứng dụng thực hiện như một công việc đơn trên một hay một số nút lưới

Bước 2	xử lý theo lô đồng thời	Các thể hiện độc lập của ứng dụng được thực hiện một cách đồng thời. Trong quá trình thực hiện, các thể hiện này không có sự giao tiếp với nhau và các thể hiện này là tương đồng nhau
Bước 3	xử lý theo lô song song	Công việc của một chương trình chạy theo lô được chia nhỏ thành các công việc con để cho các thể hiện chương trình độc lập nhau có thể xử lý song song
Bước 4	Dịch vụ	Chương trình trở thành một tiến trình dịch vụ được gọi bởi nút khách thông qua phần đệm lưới
Bước 5	Dịch vụ song song	Chương trình trở thành đa thể hiện của một dịch vụ được gọi bởi nút khách một cách song song thông qua phần đệm lưới
Bước 6	Các chương trình song song phụ thuộc chặt	Đây là các chương trình chạy song song chuyên biệt, được thiết kế ngay từ đầu để có thể thực hiện trên lưới. Do được chuyên biệt hoá nên các ứng dụng này phải được xử lý song song theo mô hình phụ thuộc chặt

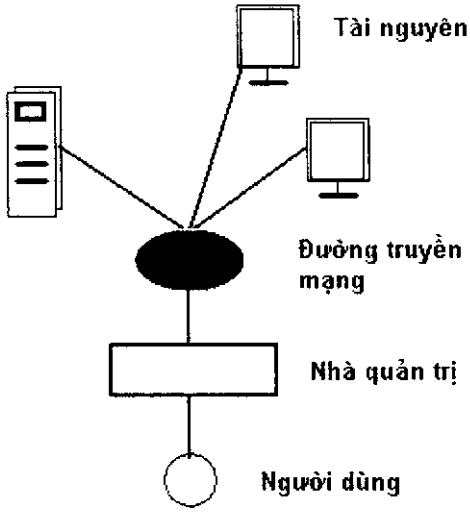
Mục đích chính của việc lưới hoá ứng dụng qua sáu bước kể trên là nâng cao hiệu năng tính toán của ứng dụng, tận dùng tối đa khả năng tính toán của lưới. Một yếu tố khác là để tách biệt giữa quá trình phát triển ứng dụng và quá trình triển khai lên lưới tức là việc phát triển ứng dụng không bị lệ thuộc vào phần đệm lưới cũng như cơ sở hạ tầng lưới.

8.2. Xây dựng ứng dụng trên lưới theo mô hình hướng dịch vụ

Trong phần trên chúng ta đã đã đề cập các giải pháp tổng thể để xây dựng một ứng dụng lưới. Giải pháp đó là từ một hệ thống tài nguyên đã có trong đó có cả các ứng dụng chuyên biệt, người phát triển ứng dụng sẽ xây dựng một lưới tính toán sau đó sẽ lưới hoá ứng dụng đã có theo trình tự sáu bước đã nêu. Phần tiếp theo đi sâu vào chi tiết ở mức triển khai và cài đặt ứng dụng lưới. Đây là các kỹ thuật thuần tuý mà người xây dựng ứng dụng lưới phải nắm bắt được trong quá trình xây dựng. Trước hết xin giới thiệu sự phát triển về các quan điểm xây dựng ứng dụng lưới.

8.2.1. Quan điểm xây dựng lưới theo mô hình hướng tài nguyên

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của công nghệ tính toán, tính toán lưới đã đạt được một số kết quả nhất định. Một số lưới đã được xây dựng bao gồm lưới tính toán, lưới dữ liệu, lưới khoa học, lưới tri thức hay các cộng đồng lưới. Mỗi loại lưới cùng chia sẻ một dạng thức tài nguyên, do đó có kiến trúc khác với loại lưới khác.



Hình 8-9: Mô hình lưới tính toán đơn giản

Một tiêu chí quan trọng để đánh giá về lưới bất kì đó là khả năng thoả mãn yêu cầu người dùng. Sự thoả mãn yêu cầu người dùng dựa trên chất lượng dịch vụ (QoS) được lưới cung cấp như tính hữu dụng, hiệu năng, tính đơn giản khi sử dụng, chi phí... Tính toán lưới trong giai đoạn phát triển đầu tiên gắn liền với các chức năng về dữ liệu, tính toán và cơ chế truy cập lưới. Các chức năng này của lưới phải thoả mãn các yêu cầu:

1. Dữ liệu: dữ liệu là một phần quan trọng trong bất kì môi trường lưới nào. Do đó, dữ liệu phải được quản lý có hiệu quả bao gồm các kho chứa dữ liệu, trao đổi dữ liệu, truy cập dữ liệu và tính an toàn bảo mật cho dữ liệu. Những đòi hỏi cơ bản về dữ liệu cho một ứng dụng lưới bao gồm:

Khả năng tích hợp dữ liệu các nguồn tài nguyên phân tán, độc lập và không đồng nhất.

Khả năng cung cấp cơ chế trao đổi dữ liệu có hiệu quả và hỗ trợ kịp thời về mặt dữ liệu mỗi khi phục vụ cho mục đích tính toán.

Khả năng cung cấp cơ chế bộ đệm, cơ chế sao lưu dữ liệu để giảm thiểu yêu cầu về băng thông khi lan truyền trên mạng.

Khả năng hỗ trợ cơ chế khai phá dữ liệu, cho phép người sử dụng tìm kiếm theo các đặc trưng của dữ liệu.

Khả năng mã hoá và bảo vệ dữ liệu sao cho dữ liệu được truyền thông trên mạng luôn được an toàn.

2. Tính toán: các yêu cầu về mặt tính toán của một ứng dụng lưới bao gồm:
Khả năng cho phép quản lý độc lập các nguồn tài nguyên tính toán. Điều này có nghĩa là các thành phần tính toán khi tham gia lưới đều có tính tự trị nhất định.
Cung cấp cơ chế thông minh và trong suốt đối với người sử dụng khi lựa chọn các tài nguyên tính toán để thực hiện công việc nhất định.
Điều khiển được trạng thái làm việc của các tài nguyên như nắm rõ tình trạng làm việc các tài nguyên, cấu hình động các tài nguyên và đưa ra lịch trình làm việc cho các tài nguyên.
Có cơ chế phát hiện lỗi và sửa lỗi khi tài nguyên gặp sự cố.
Đảm bảo cơ chế an ninh phục vụ cho quá trình chia sẻ và truy cập tài nguyên.
Như vậy, môi trường tính toán lưới (lưới tính toán, lưới dữ liệu...) bao gồm một hay nhiều môi trường hỗ trợ phần cứng hay phần mềm mà có thể cung cấp khả năng tính toán độc lập, mạnh mẽ và chi phí thấp cho người sử dụng (định nghĩa của Foster&Kesselman, 1998). Tuỳ vào mục đích cụ thể mà lưới xây dựng có kiến trúc cũng như những đòi hỏi khác nhau. Nếu đó là lưới tính toán thì sức mạnh tính toán được quan tâm nhiều, như khả năng tải, độ tin cậy, khả năng chịu lỗi... của tài nguyên lưới. Hay nói cách khác đi, các lưới đã và đang được xây dựng theo mô hình "hướng tài nguyên". Mô hình này quan tâm hàng đầu đến các loại hình tài nguyên tham gia lưới. Từ đó mới có thể hỗ trợ người sử dụng một số nhu cầu nhất định. Tuy nhiên mô hình lưới "hướng tài nguyên" ngày càng bộc lộ các điểm yếu như:
Tính chất "hướng tài nguyên" có thể tạo điều kiện cho nhà quản trị lưới quản lý tốt hơn các thành phần tham gia lưới. Tuy nhiên, tính chất này lại giới hạn khả năng đáp ứng của lưới. Chẳng hạn, khi người sử dụng có nhu cầu lưu trữ một lượng lớn dữ liệu trên lưới tính toán thì có khả năng lưới tính toán không đáp ứng được.
Vấn đề thứ hai là tính liên tác giữa các nguồn tài nguyên không đồng nhất. Một khi lượng tài nguyên trong lưới nhiều lên, cơ chế trao đổi đa dạng thì việc phối hợp giữa các nguồn tài nguyên này là rất khó khăn.
Vấn đề mở rộng lưới cũng được đặt ra. Do lưới được xây dựng trên cơ sở một mô hình tài nguyên chung nên việc mở rộng hay nâng cấp sẽ bị hạn chế. Nhà quản trị không thể thay đổi kiến trúc lưới đã xây dựng trong khi đó nhu cầu của người sử dụng ngày càng phức tạp.
Chính vì vậy, xu thế phát triển lưới trong vài năm gần đây là theo mô hình hướng dịch vụ (SOA). Đây là mô hình được xây dựng trên nền tảng các công nghệ và các chuẩn giao thức đã có. Phần tiếp theo sẽ trình bày chi tiết về mô hình lưới hướng dịch vụ.

8.2.2. Mô hình hướng dịch vụ đối với việc phát triển ứng dụng lưới

Đây là mô hình hiện đại và đang được hoàn thiện. Phần này trình bày chi tiết về mô hình ứng dụng lưới theo hướng dịch vụ (SOA). Phần này được trình bày theo quan điểm cơ sở mô hình. Như đã trình bày ở phần trước, tính toán lưới ở giai đoạn phát triển đầu tiên tập trung chủ yếu vào các vấn đề mang tính chất tài nguyên thuần túy. Trước khi bắt đầu xây

dựng một lưới, người thiết kế phải hoạch định rõ ràng các nguồn tài nguyên tham gia lưới như sức mạnh tính toán của tài nguyên, các khi dữ liệu và các nguồn lưu trữ...

Ngày nay, các ý tưởng về lưới đã có sự thay đổi nhất định. Thay vì tập trung vào chi tiết các tài nguyên tham gia lưới, các nhà thiết kế đặt trọng tâm vào quá trình phối hợp hoạt động giữa các nguồn tài nguyên không đồng nhất. Công việc này được thực hiện thông qua một tổ chức được gọi là tổ chức ảo (VO-Virtual Organization). Như vậy, khái niệm về các nguồn tài nguyên cơ sở tham gia vào lưới không có sự thay đổi theo như ý tưởng ban đầu và ý tưởng hiện nay. Các nguồn tài nguyên này là sức mạnh tính toán, dữ liệu, phần cứng, các ứng dụng, thiết bị mạng hay bất cứ hình thức tài nguyên tính toán nào. Vấn đề khác nhau chính là cách thức tổ chức tài nguyên hay nói cách khác đó là mô hình tổ chức các tài nguyên tham gia lưới. Việc đưa ra mô hình tổ chức mới, thông qua tổ chức ảo đã làm đơn giản hóa nhiều việc chia sẻ tài nguyên cũng như quá trình hoạt động của lưới.

Trong hệ thống lưới, các tài nguyên cơ sở là các hệ thống tính toán (Computer System), các máy tính phân cụm (Cluster), các hệ cơ sở dữ liệu (Database System)... Tuy nhiên, thay vì tổ chức các tài nguyên cơ sở này trong các đơn vị nguyên tử (như các ý tưởng ban đầu về lưới), người sử dụng hợp lệ có thể đưa ra các yêu cầu sử dụng tài nguyên bất cứ lúc nào (On-Demand). Như vậy, thông qua môi trường tính toán lưới như trên, các tài nguyên luôn sẵn sàng phục vụ các tổ chức hay người sử dụng hợp lệ. Các yêu cầu sử dụng mà lưới có thể đáp ứng rất đa dạng, có thể là yêu cầu về khả năng lưu trữ, yêu cầu giải pháp phần mềm cho một bài toán xác định hay đòi hỏi về băng thông truyền tải dữ liệu...

Dựa trên các đòi hỏi của người sử dụng, mỗi yêu cầu của họ thường có thể có nhiều giải pháp khác nhau. Mỗi loại yêu cầu như vậy sẽ tương ứng với một tổ chức ảo. Mỗi loại tổ chức ảo được xây dựng, bảo trì và phục vụ người sử dụng trên cơ sở các nguồn tài nguyên cần thiết để giải quyết bài toán đặt ra. Tổ chức ảo để giải quyết bài toán dự báo thời tiết có nhiệm vụ:

+ Thiết lập các tài nguyên cần thiết để xử lý bài toán như ứng dụng phần mềm dự đoán liên quan đến thời tiết, yêu cầu về phần cứng để chạy ứng dụng trên và dung lượng lưu trữ cần thiết để lưu trữ dữ liệu trong quá trình dự báo.

+ Quản lý các nguồn tài nguyên này, đảm bảo quá trình vận hành trơn tru. Trong khi thực hiện, tổ chức ảo có thể đòi hỏi thêm các tài nguyên nếu cần thiết.

Như vậy, mô hình lưới như trên có tính linh hoạt và động hơn nhiều so với mô hình lưới với sự tham gia của các đơn vị nguyên tử. Mô hình này còn được gọi là mô hình tính toán theo yêu cầu (On-Demand Computing Environment). Muốn xây dựng hệ thống lưới như vậy thì phải tạo được một cơ sở hạ tầng các nguồn tài nguyên dồi dào. Các tài nguyên khi tham gia vào lưới cho dù ở bất cứ đâu hay thuộc tổ chức nào đều trong trạng thái sẵn sàng đáp ứng yêu cầu. Hay nói cách khác, khi xây dựng lưới theo mô hình thông qua các tổ chức ảo, về mặt tài nguyên cần phải đáp ứng các yêu cầu như sau:

Khả năng tìm kiếm tài nguyên trong lưới theo khả năng hay theo chức năng
Khả năng cấp phát tài nguyên theo yêu cầu của người sử dụng.
Có khả năng quản lý và phối hợp các nguồn tài nguyên tương ứng với các mức độ đòi hỏi dịch vụ (SLAs-Service Level Agreements) khác nhau.
Đồng thời, các tài nguyên tham gia lưới cũng phải có một số đặc tính tự trị ở một mức độ nhất định như tính tự chẩn đoán, tự sửa chữa, tự cấu hình và tự quản lý.
Cuối cùng, bất cứ sự yêu cầu hay sử dụng tài nguyên lưới cũng phải thông qua cơ chế bảo mật chặt chẽ. Hơn thế nữa, các cơ chế bảo mật này phải trong suốt đối với người sử dụng.
Hệ thống lưới nếu đáp ứng được các yêu cầu như trên sẽ là một hệ thống có một cơ sở hạ tầng về mặt tài nguyên tương đối hoàn chỉnh và mạnh mẽ. Vấn đề tiếp theo cần quan tâm là việc xây dựng các tổ chức ảo. Nếu cơ sở hạ tầng được xây dựng tốt nhưng không có các tổ chức ảo hay các tổ chức ảo không tận dụng hết các lợi thế của lưới thì chưa thể gọi là lưới hoàn thiện được. Bộ mặt của lưới được thể hiện qua các tổ chức ảo. Chính vì vậy, để tạo dựng một lưới hoàn chỉnh thì các tổ chức ảo của lưới phải thỏa mãn được các đòi hỏi:
+ Trước hết, mỗi tổ chức ảo phải cung cấp đầy đủ thông tin cho người sử dụng về các tác vụ mà nó thực thi, các bài toán mà nó giải quyết.
+ Các tổ chức ảo phải có khả năng thu thập động các tài nguyên mang tính chất không đồng nhất trên cơ sở các yêu cầu của người sử dụng và tính phức tạp của bài toán cần giải quyết.
+ Khả năng quản lý ở mức cao các nguồn tài nguyên, phối hợp các nguồn tài nguyên tương ứng với các mức yêu cầu của dịch vụ.
+ Hỗ trợ cơ chế bảo mật tiện lợi, an toàn, dễ sử dụng cho người dùng. Chẳng hạn như người sử dụng chỉ cần đăng kí một lần duy nhất khi tham gia vào lưới và từ đó có thể sử dụng các tài nguyên lưới.
+ Vấn đề cuối cùng khi hoàn thiện lưới là các ứng dụng mà lưới có thể cung cấp cho người sử dụng. Đây là mục tiêu cuối cùng của lưới. Vì thế, lưới phải thỏa mãn một số đòi hỏi như:
Khả năng xác định chính xác, rõ ràng vấn đề mà người sử dụng đưa ra.
Đưa ra các đánh giá ban đầu và yêu cầu về tài nguyên để thực hiện được ứng dụng.
Có khả năng cung cấp các dịch vụ với các mức độ chất lượng (QoS-Quality of Service) khác nhau.
Như vậy, mỗi khi người sử dụng đưa ra yêu cầu của họ, yêu cầu này được xử lý bởi các tổ chức ảo. Có thể hiểu các tổ chức ảo này là các đơn vị logic có đảm nhận hết các phần phức tạp của lưới như xác định yêu cầu, tìm kiếm các nguồn tài nguyên phù hợp, đưa ra yêu cầu về tài nguyên, phối hợp hoạt động giữa các tài nguyên để trả lại kết quả cuối cùng cho người sử dụng. Như vậy, các tổ chức ảo chỉ quản lý tài nguyên ở mức cao. Việc điều hành, bảo trì cơ sở hạ tầng của lưới không phải công việc trực tiếp của các tổ chức ảo mà do các đơn vị khác đảm nhận. Các đơn vị này chính là các nhà cung cấp dịch vụ. Chức năng của các tổ chức ảo sẽ là cầu nối giữa các nhà cung cấp dịch vụ và người sử dụng. Mô hình hướng lưới

hướng dịch vụ như trên có ưu điểm hơn hẳn so với các mô hình về lưới trước đây. Thể hiện ở các điểm sau:

+ Mô hình có tính chất động, tức là mỗi khi người sử dụng có nhu cầu thì các tổ chức ảo sẽ được kích hoạt và liên hệ với các nhà cung cấp dịch vụ để giải quyết bài toán đặt ra.

+ Chi tiết cụ thể về tài nguyên bao gồm quá trình quản lý, bảo trì, vận hành đều thông qua các nhà cung cấp dịch vụ. Do đó, các tài nguyên này được sử dụng triệt để hơn, hiệu năng cao hơn vì các nhà cung cấp luôn có những giải pháp để nâng cao hiệu quả kinh tế khi khai thác các tài nguyên này.

+ Việc bảo trì và mở rộng lưới sẽ đơn giản hơn nhiều. Việc bảo trì lưới là do các nhà cung cấp dịch vụ đảm nhiệm. Tất nhiên là giữa các nhà cung cấp dịch vụ luôn có các chuẩn hoạt động nhất định bởi vì họ vẫn hoạt động trong cùng một lưới thống nhất. Mặt khác, vấn đề mở rộng lưới đơn giản là sự thỏa hiệp của một nhà cung cấp dịch vụ mới khi muốn tham gia hoạt động trong lưới. Bất kì nhà cung cấp nào muốn vào lưới phải tuân theo các chuẩn cũng như các quy tắc hoạt động của lưới.

Phần tiếp theo của sẽ trình bay chi tiết hơn về mô hình hướng dịch vụ và quá trình tích hợp giữa công nghệ dịch vụ Web và dịch vụ lưới.

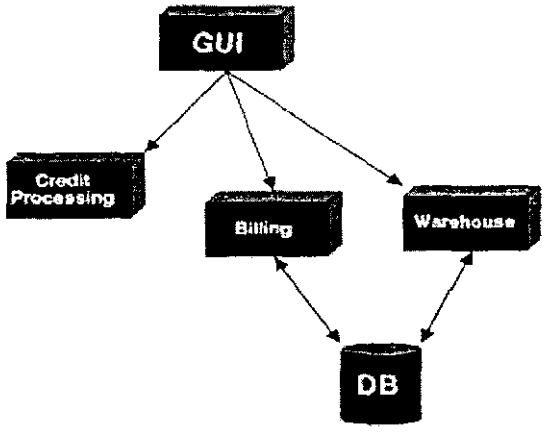
8.2.3. Xây dựng ứng dụng lưới theo kiến trúc SOA

8.2.3.1. Mô hình SOA

Trong phần này, ứng dụng lưới được trình bày trên quan điểm công nghệ. Một trong các công nghệ được ứng dụng để phát triển ứng dụng lưới là công nghệ dịch vụ Web. Ngoài ra còn có các công nghệ liên quan như XML, Security...

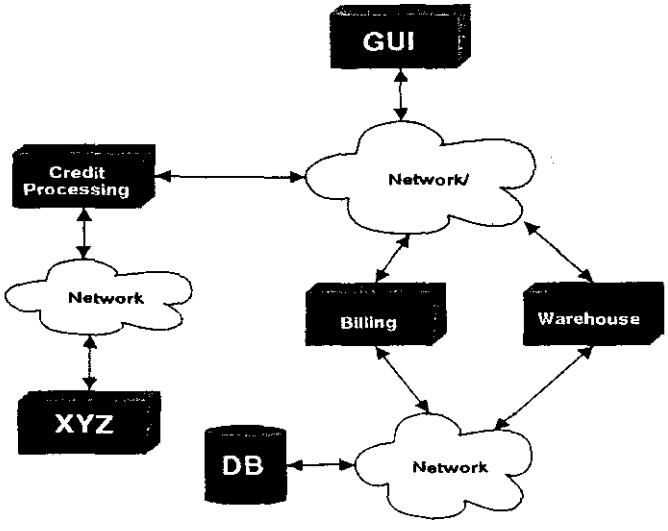
Kiến trúc hướng dịch vụ được phát triển nhằm mục đích tăng khả năng kết nối động, không bị lệ thuộc chặt chẽ vào nhau (loosely-couple) của các thành phần trong cùng một hệ thống. Các thành phần của hệ thống có thể là các ứng dụng hay các dịch vụ cơ sở. Hơn thế nữa, kiến trúc này còn định nghĩa các chuẩn để tích hợp các thành phần này với nhau. Trong kiến trúc SOA, hệ thống là một tập các thành phần được kết nối với nhau qua mạng. Một ứng dụng hay một dịch vụ có thể được xây dựng trên cơ sở kết hợp các thành phần này lại với nhau. Chính vì tính chất động và không bị lệ thuộc chặt chẽ giữa các thành phần nên hệ thống dễ dàng được thay đổi hay nâng cấp để đáp ứng nhu cầu ngày càng cao của người sử dụng. Các thành phần sẽ được phát triển độc lập mà không ảnh hưởng tới toàn bộ hệ thống. Như vậy, mỗi khi hệ thống cần nâng cấp thì chỉ những thay đổi chỉ diễn ra ở các môđun cần thiết. Các thay đổi này không ảnh hưởng tới hoạt động của hệ thống và chỉ khi nào các thành phần cần sửa chữa được hoàn thiện thì lúc đó hệ thống mới ra đời sẽ thay thế hệ thống cũ. Những ưu điểm của kiến trúc SOA được minh họa qua ví dụ sau:

Một ứng dụng xử lý giao dịch được thiết kế có các chức năng :xử lý thẻ thanh toán, xử lý dữ liệu thô, tính toán hóa đơn và giao diện người dùng. Theo phương pháp thiết kế truyền thống tức là thiết kế hướng chức năng mô tả hệ thống như hình vẽ:



Hình 8-10: Mô hình thiết kế ứng dụng truyền thống

Mô hình này có ưu điểm là rõ ràng, mạch lạc, người phát triển quản lý toàn bộ các môđun, điều khiển tương tác giữa các thành phần. Tuy nhiên, giả sử yêu cầu về chức năng tính toán hóa đơn có sự thay đổi thì toàn bộ hệ thống phải ngừng hoạt động để nâng cấp. Sở dĩ có điều này bởi vì các môđun gắn kết với nhau chặt chẽ và hệ thống chỉ hoạt động khi các môđun con cùng hoạt động và phối hợp với nhau. Chính vì vậy, mô hình SOA ra đời đã khắc phục nhiều điểm yếu của mô hình hướng chức năng. Mô hình này được minh họa như sau:



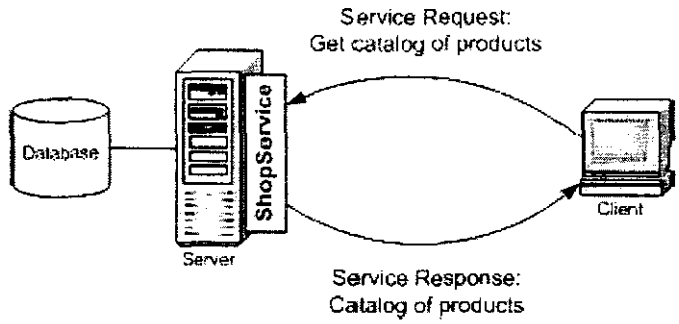
Hình 8-11: Mô hình thiết kế ứng dụng theo hướng SOA

Như vậy, ứng dụng cần xây dựng đã được chia thành các thành phần và được kết nối với nhau qua mạng máy tính. Các thành phần không cần thiết phải gắn kết chặt chẽ với nhau mà chỉ khi nào có yêu cầu thì chúng mới giao tiếp với nhau. Mặt khác, mỗi thành phần có thể được nâng cấp, thay đổi độc lập với các thành phần khác. Vì vậy, hệ thống luôn trong trạng thái sẵn sàng đáp ứng yêu cầu luôn thay đổi của người sử dụng.

Trong phần tiếp theo, chúng ta sẽ tìm hiểu một công nghệ đang phát triển mạnh và gắn liền với kiến trúc SOA, đó là công nghệ WebService.

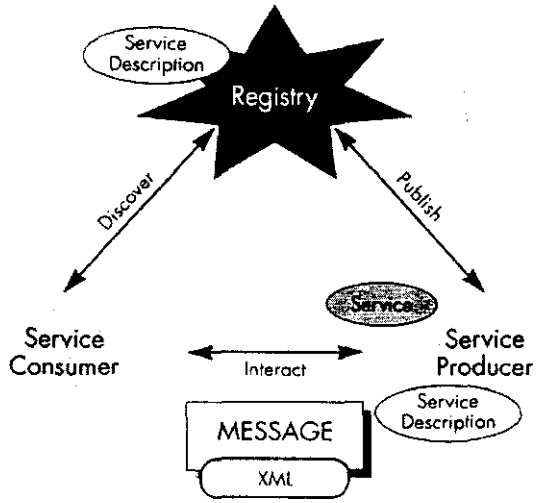
8.2.3.2. Công nghệ dịch vụ Web (Web Service)

Một trong những ứng dụng thể hiện rõ nét nhất của kiến trúc SOA là công nghệ WebService. Mô hình dịch vụ Web ra đời trước khi kiến trúc SOA được xây dựng. Tuy nhiên, chỉ khi cơ sở lý luận về mô hình lập trình hướng dịch vụ hoàn thiện thì mô hình dịch vụ Web mới thực sự phát triển và trở thành một chuẩn công nghiệp như hiện nay. Trong công nghệ WebService, người sử dụng đóng vai trò là máy khách (Client) sẽ sử dụng các dịch vụ do nhà cung cấp dịch vụ (hay còn gọi là Server) hỗ trợ. Người sử dụng và nhà cung cấp dịch vụ trao đổi với nhau thông qua cơ chế truyền thông điệp



Hình 8-12: Mô hình Web Service

Mô hình tương tác giữa các thành phần trong công nghệ Web Service diễn ra như sau:

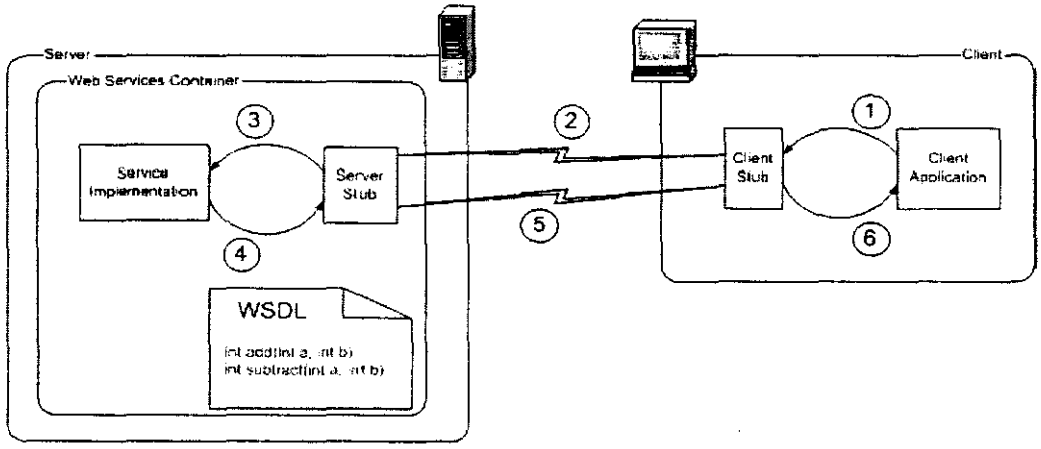


Hình 8-13: Tương tác giữa các thành phần trong Web Service

Nhà cung cấp sau khi xây dựng các dịch vụ sẽ mô tả dịch vụ của họ với người sử dụng theo định dạng chuẩn (ví dụ WSDL...). Đồng thời các dịch vụ này sẽ được đăng kí với một thanh ghi đóng vai trò như là phương tiện để quảng bá dịch vụ cho nhà cung cấp. Thanh ghi cũng

là nơi mà người sử dụng có thể tìm kiếm dịch vụ phù hợp nhất cho nhu cầu của họ. Quy trình thực hiện một giao dịch có thể được tóm gọn như sau:

1. Khi máy khách muốn khởi động một dịch vụ, nó gọi một gói chương trình, gọi là stub khách. Stub này có nhiệm vụ chuyển yêu cầu theo chuẩn SOAP và chuyển tới máy chủ theo giao thức HTTP.
2. Máy chủ nhận yêu cầu, chuyển nó cho stub chủ. Stub này chuyển ngược lại yêu cầu dưới dạng SOAP về dạng mà trình thực thi hiểu được.
3. Trình thực thi dịch vụ nhận yêu cầu và thực hiện.
4. Kết quả thực thi được chuyển qua stub chủ. Stub chủ lại chuyển kết quả này dưới dạng SOAP.
5. Stub khách nhận phản hồi dưới dạng SOAP và chuyển nó về dạng sao cho trình ứng dụng khách hiểu được.
6. Trình ứng dụng nhận kết quả của dịch vụ và sử dụng kết quả.



Hình 8-14: Mô tả một giao dịch dịch vụ Web

Dịch vụ Web có ưu thế hơn các công nghệ tính toán phân tán khác (như CORBA, RMI...) như:

Không phụ thuộc vào nền tảng và ngôn ngữ vì nó dựa vào chuẩn XML. Giữa máy khách và máy chủ không cần có cùng nền tảng và không phụ thuộc vào nhau (loosely coupled).

Dịch vụ Web sử dụng giao thức HTTP để truyền thông điệp giữa máy khách và máy chủ. Do đó khả năng ứng dụng trên Internet là rộng rãi.

Tuy nhiên công nghệ này cũng có một số điểm yếu như:

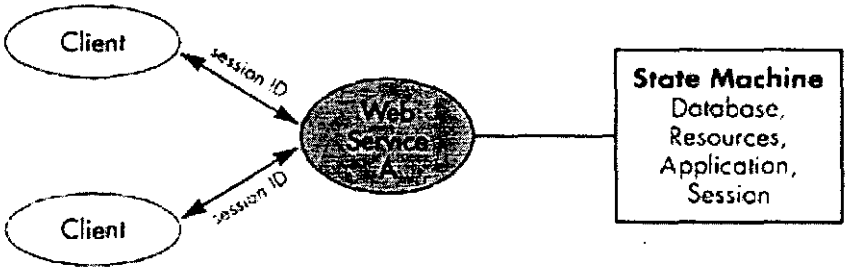
Do sử dụng chuẩn XML là ngôn ngữ chung nên hiệu năng không cao bằng việc sử dụng mã nhị phân (XML có dạng text).

Không có tính đa dạng tức là không hỗ trợ nhiều dịch vụ hỗ trợ khác (giao dịch, quản lý vòng đời, tính bảo tồn...). Đây cũng chính là các dịch vụ mà Grid Service được bổ sung thêm vào.

Công nghệ WebService chính là cơ sở để xây dựng các GridService. Như đã trình bày ở phần trước, tính toán lưới là quá trình kết hợp thống nhất các nguồn tài nguyên không đồng nhất. Tính chất chia sẻ này phần nào đã được thể hiện qua mô hình SOA và công nghệ WebService. Chính vì vậy, GridService là một bước phát triển cao hơn của WebService, sẽ tận dụng hết ưu điểm của công nghệ cũ đồng thời phải được bổ sung các đặc tính mang đặc trưng của lưới. Để hiểu rõ hơn sự bổ sung này, chúng ta tìm hiểu một số sự khác biệt giữa WebService và GridService. Grid Service là công nghệ tính toán lưới dựa trên Dịch vụ Web, trong đó có bổ sung một số đặc tính mới để phù hợp với môi trường lưới..

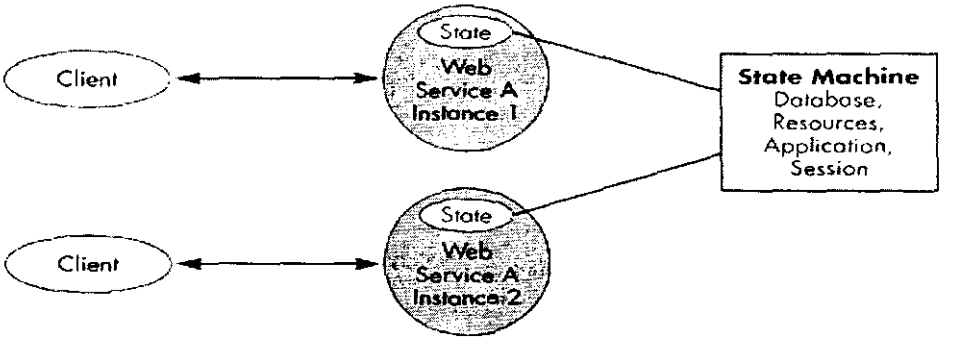
8.2.3.3. Web Service và GridService

Một ứng dụng hay một dịch vụ lưới có thể quản lý trạng thái bên trong của nó và các thông tin về trạng thái của dịch vụ sẽ được người sử dụng lưu trữ. Thông thường, các thông tin về một phiên làm việc của người sử dụng sẽ được lưu lại để phục vụ cho các lần truy cập tiếp theo. Ví dụ như trong một hệ thống bán hàng qua mạng, các thông tin về người mua hàng như tên người mua, ngày mua, thông tin về giao dịch... sẽ được lưu lại. Lợi thế của mô hình này là phía nhà cung cấp dịch vụ sẽ không phải lưu trữ bất kì thông tin nào về khách hàng đồng thời cũng không phải tạo ra các thể hiện của dịch vụ khác nhau cho mỗi người dùng khác nhau.



Hình 8-15: Mô hình Web Service phi trạng thái

Mô hình WebService phi trạng thái này được sử dụng tương đối phổ biến. Bên cạnh đó, còn một mô hình WebService có trạng thái. Trong mô hình này, các thông tin về trạng thái sẽ được chính dịch vụ lưu trữ, người sử dụng chỉ cần kết nối với dịch vụ và sử dụng dịch vụ mà không phải lưu bất kì thông tin nào. Muốn vậy, mỗi khi có yêu cầu dịch vụ thì một thể hiện của dịch vụ phải được tạo ra và thể hiện này là duy nhất, chỉ phục vụ cho người sử dụng cho đến khi hết yêu cầu sử dụng dịch vụ. Mô hình này được minh họa như sau:



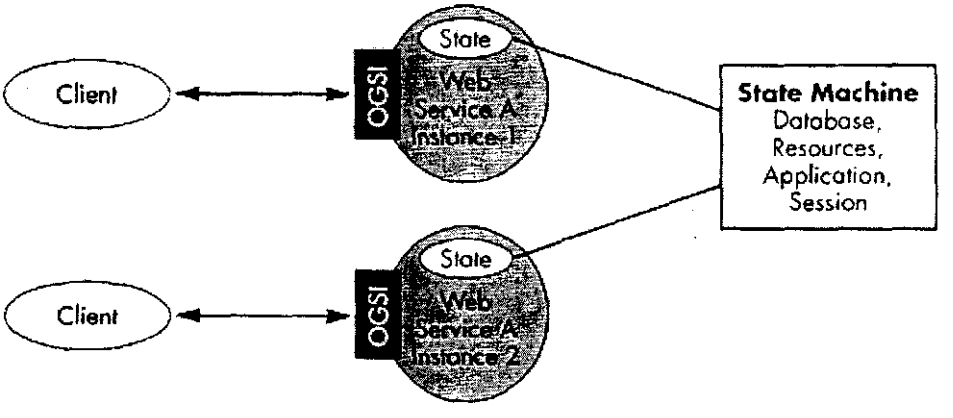
Hình 8-16: Mô hình Web có trạng thái

Mô hình Web Service có trạng thái này là cơ sở cho Grid Service. Grid Service chính là Web Service lưu trạng thái của chính dịch vụ lưới đồng thời nó cũng qui định một tập các giao diện chuẩn để giao tiếp với người sử dụng. Thông qua các giao diện chuẩn này, máy khách có thể thiết lập hay lấy về các thông tin trạng thái ngoài các giao tiếp dịch vụ thông thường. Tập các giao diện chuẩn phải tuân theo đặc tả OGSI (Open Grid Service Infrastructure). OGSI là đặc tả cho các dịch vụ lưới, cung cấp bộ máy phục vụ cho các công việc như:

Quản lý vòng đời dịch vụ.

Tìm kiếm, khai phá trạng thái dịch vụ.

Thông báo các thay đổi về trạng thái của dịch vụ.



Hình 8-17: Mô hình dịch vụ lưới theo chuẩn OGSI

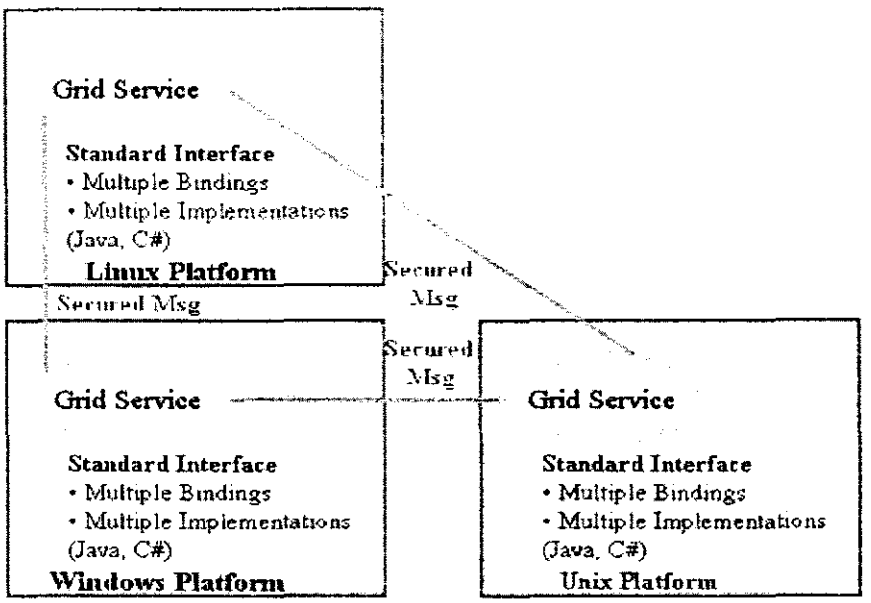
Phần tiếp theo của sẽ trình bày về kiến trúc OGSA và các kỹ thuật triển khai dịch vụ lưới.

8.2.3.4. Dịch vụ lưới và chuẩn OGSA

Như đã trình bày trong chương một. OGSA là chuẩn để xây dựng dịch vụ lưới. Hơn nữa, OGSA hỗ trợ cho mô hình hướng dịch vụ. Chính vì vậy, phần này đi sâu vào kiến trúc OGSA theo quan điểm kỹ thuật để thấy được sự hỗ trợ của chuẩn OGSA trong việc xây dựng một ứng dụng lưới.

Các dịch vụ lưới tuân theo chuẩn OGSA là thống nhất với nhau không phụ thuộc vào nền tảng trên đó dịch vụ hoạt động (Hình 8.19). Các dịch vụ lưới là các dịch vụ Web đặc biệt có cung cấp thêm các giao diện chuẩn theo quy ước. Các giao diện chuẩn của dịch vụ lưới

phục vụ cho việc tìm kiếm dịch vụ, khởi tạo dịch vụ động, quản lý vòng đời dịch vụ, cơ chế thông báo... Ngoài ra, do các giao diện trên là chuẩn nên nhà phát triển dịch vụ có thể thực thi dịch vụ bằng các ngôn ngữ khác nhau như C, C#, Java... Hơn thế nữa, OGSA còn hỗ trợ một cơ chế bảo mật lưới để đảm bảo các dịch vụ giao tiếp với nhau an toàn. Các dịch vụ lưới tự mô tả các thông tin về phương thức, tham số... của nó thông qua các tệp mô tả giao diện WSDL; so với file mô tả dịch vụ Web, file mô tả dịch vụ lưới có thêm thẻ có tên là **<gsdl>**.



Hình 8-18: Các dịch vụ lưới OGSA trên các nền tảng khác nhau

OGSA định nghĩa cú pháp của một thể hiện của dịch vụ lưới bao gồm khởi tạo dịch vụ lưới, đặt tên, quản lý vòng đời và các giao thức liên kết giữa các dịch vụ với nhau. Quá trình khởi tạo một thể hiện của dịch vụ bao gồm việc tạo một tiến trình trong môi trường chứa. Môi trường chứa này sẽ đảm bảo dịch vụ mà nó triển khai là tuân theo đúng cú pháp cũng như chuẩn của một dịch vụ lưới. Ngoài ra, một giao diện dịch vụ lưới có thể tạo ra nhiều thể hiện khác nhau. Chính vì vậy, OGSA đưa ra các đặc tả sao cho môi trường chứa có khả năng quản lý nhiều thể hiện dịch vụ lưới (như chỉnh xửa dịch vụ, thêm dịch vụ...) cùng một thời điểm.

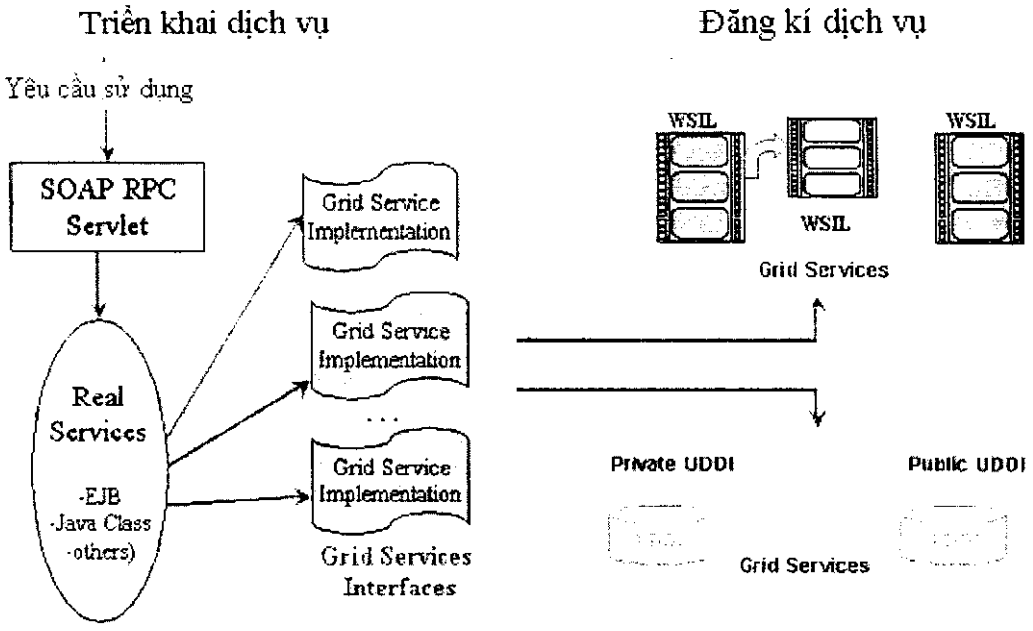
- Khả năng của dịch vụ: đây là khái niệm được sử dụng rộng rãi trong công nghệ dịch vụ Web. Ví dụ như, một dịch vụ chuyên chở có khả năng vận chuyển một tấn hàng trong hai ngày với giá 100 $. Tương tự như vậy, khả năng của dịch vụ lưới có thể là tài nguyên tính toán, tài nguyên lưu trữ, băng thông mạng, chương trình, cơ sở dữ liệu... Các dịch vụ lưới được đặc trưng bởi khả năng mà dịch vụ đó cung cấp. Khả năng của dịch vụ được thể hiện qua giao diện dịch vụ. Giao diện dịch vụ càng nhiều, hỗ trợ đa dạng các phương thức thì khả năng của dịch vụ càng cao.

- Phiên bản dịch vụ và nâng cấp: giao diện dịch vụ và các thông tin phiên bản của dịch vụ được xác định trong thẻ serviceType của dịch vụ theo đặc tả OGSA. Các dịch vụ lưới trong hệ phân tán được nâng cấp dựa trên các thông tin về phiên bản định nghĩa trong tệp mô tả dịch vụ. Tính tương thích giữa các dịch vụ lưới được quản lý dựa trên các thông tin về phiên bản. Người sử dụng dịch vụ dễ dàng tìm và kích hoạt dịch vụ với phiên bản phù hợp nhất.
- Quản lý trạng thái "mềm": dịch vụ lưới có thể tự quản lý trạng thái của nó để phân biệt giữa các dịch vụ có cùng một giao diện. Khái niệm thể hiện dịch vụ lưới nhằm nói tới một thể hiện của dịch vụ lưới trong quá trình thực hiện dịch vụ. Các dịch vụ tương tác với nhau thông qua cơ chế trao đổi thông điệp. Trạng thái bên trong của dịch vụ có thể đảm bảo rằng dịch vụ lưới đã nhận thông điệp hay chưa. Cơ chế trao đổi thông điệp tin cậy được đảm bảo bằng trạng thái bên trong của dịch vụ này có thể được dùng để xây dựng các giao dịch hướng thương mại.

Ví dụ như đối với một dịch vụ không thường trực, OGSA cung cấp một cơ chế lưu lại thông tin trạng thái liên quan đến bất kì một thao tác nào gặp sự cố. Nếu có một thao tác không thực hiện được và khi không còn lời triệu gọi dịch vụ thì thông điệp có tên là keeplive sẽ dừng lại, không chuyển tới dịch vụ nữa. Sau đó, dịch vụ sẽ tự huỷ và giải phóng tài nguyên có liên quan.

Hình 8-19 minh họa quá trình triển khai và đăng ký dịch vụ lưới. Thành phần RPC Servlet theo chuẩn SOAP và chi tiết thực thi dịch vụ được triển khai bên phía máy chủ (Application Server) chạy WebSphere hay Apache Tomcat. Mỗi thông điệp triệu gọi dịch vụ sẽ được Servlet xử lý và chuyển tới dịch vụ tương ứng.

Cũng theo mô tả trên hình 8-19, hiện nay các dịch vụ lưới có thể khai báo với bộ đăng ký UDDI (Universal Description, Discovery and Integration) hay với các tài liệu WSIL (Dịch vụ Web Inspection Language). UDDI cho phép đăng ký và tìm kiếm các hãng kinh doanh cũng như các dịch vụ lưới của các hãng. UDDI là trung tâm chứa các thông tin về các dịch vụ lưới. Có hai loại thanh ghi UDDI: công cộng và cá nhân. Các nhà cung cấp dịch vụ có thể đăng ký dịch vụ của họ thông qua các thanh ghi công cộng của các hãng như IBM, Microsoft, HP... Tuy nhiên, nếu muốn dịch vụ lưới mang tính cá nhân, không bị lệ thuộc vào các hãng khác thì nhà cung cấp dịch vụ phải có một thanh ghi UDDI riêng để tự quản lý quá trình đăng ký cũng như cập nhật các dịch vụ. Ngoài ra, nếu muốn triển khai các dịch vụ lưới trên qui mô nhỏ hay với mục đích thử nghiệm thì có thể sử dụng các tài liệu WSIL. Các tài liệu WSIL hỗ trợ dịch vụ lưới cho phép khai phá dịch vụ, triển khai và kích hoạt dịch vụ mà không cần các thanh ghi UDDI. Điều này có thể thực hiện được là do WSIL là ngôn ngữ đã được chuẩn hoá và cho phép tham chiếu tới các tài liệu mô tả dịch vụ (ví dụ như WSDL). Hơn nữa, các tài liệu WSIL có thể được truy cập thuận tiện trên mạng dưới dạng một trang Web.



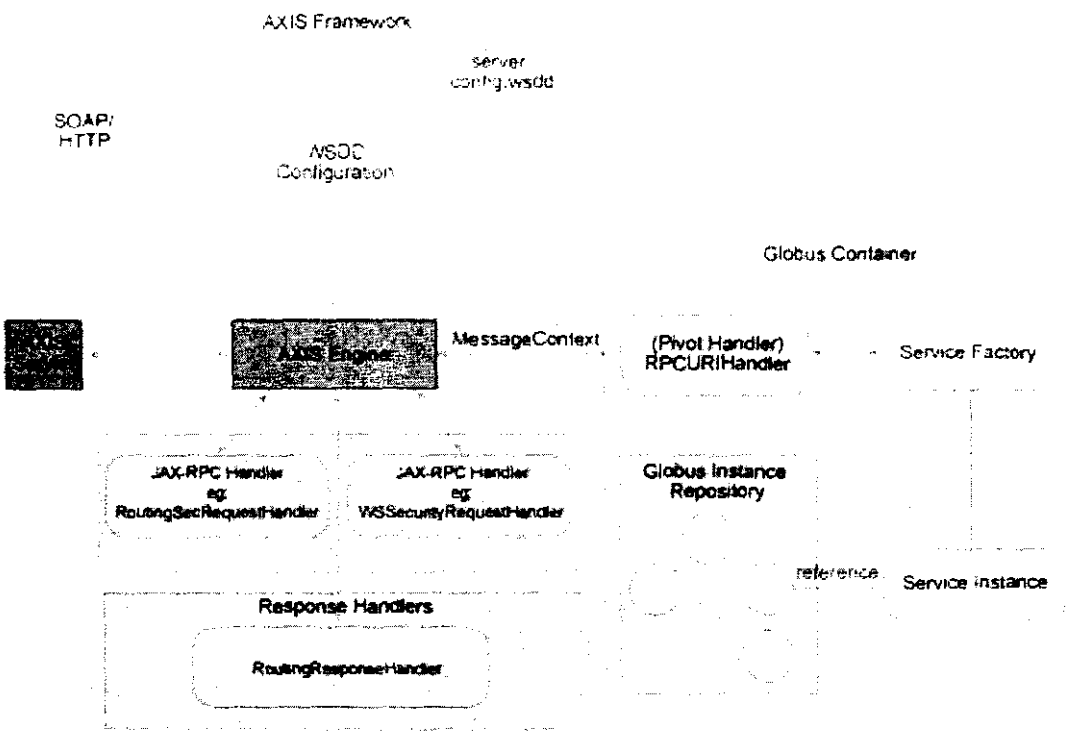
Hình 8-19: Quá trình triển khai và đăng kí dịch vụ lưới trong OGSA

8.2.4. Xây dựng dịch vụ lưới trong Globus Toolkit 3.2

Như vậy, dịch vụ lưới được xây dựng dựa trên nền tảng dịch vụ Web. Một phần là do chuẩn giao diện Web đã ra đời và đang được sử dụng phổ biến trên mạng Internet. Phần quan trọng nữa là kỹ thuật world wide Web là thể hiện rõ nét nhất của mô hình hướng dịch vụ. Cuối cùng là do dịch vụ Web được phát triển dựa trên một loạt các chuẩn công nghiệp khác như HTTP, SOAP, XML... Ngoài các đặc tính kế thừa từ công nghệ dịch vụ Web, dịch vụ lưới còn phải tuân theo chuẩn OGSA- chuẩn dành riêng cho các dịch vụ lưới. Có thể xem OGSA là bản thiết kế, khung chuẩn mà các dịch vụ muốn được xem như dịch vụ lưới phải đáp ứng đầy đủ. Muốn thực thi bản thiết kế này phải sử dụng chuẩn OGSI. OGSI so với OGSA tương tự như người kỹ sư và kiến trúc sư. OGSA đưa ra bản thiết kế còn OGSI trực tiếp thực thi bản thiết kế đó. Chính vì vậy, OGSA là chuẩn đóng vai trò nền tảng, tạo cơ sở cho việc hoàn thành các dịch vụ lưới. Còn OGSI mới thực sự là chuẩn ở mức thực thi, xây dựng dịch vụ. Phần này sẽ trình bày chi tiết mô hình lập trình dịch vụ lưới theo chuẩn OGSI bằng Globus Toolkit 3.2. Như đã giới thiệu ở phần trên, đây là bộ công cụ được sử dụng rộng rãi nhất hiện nay và hỗ trợ hoàn toàn chuẩn OGSI. Các vấn đề sẽ được trình bày bao gồm: kiến trúc phần mềm của Globus Toolkit 3.2 phía máy chủ và máy khách.

8.2.4.1. Mô hình dịch vụ lưới phía máy chủ

Các thành phần trong mô hình dịch vụ lưới phía máy khách được thể hiện chi tiết qua hình vẽ sau:



Hình 8-20: Mô hình dịch vụ lưới phía máy chủ

Các thành phần ở phía máy chủ bao gồm:

1. Bộ máy dịch vụ Web. Bộ máy này do phần mềm Apache AXIS cung cấp làm chức năng xử lý các thao tác thông thường đối với dịch vụ Web như nhận các thông điệp SOAP từ máy khách, xử lý thông điệp, gửi trả kết quả lại máy khách, cấu hình dịch vụ Web... Như trên hình vẽ, bộ máy AXIS bao gồm ba thành phần, trong đó có hai thành phần làm nhiệm vụ xử lý thông điệp nhận từ máy khách và một thành phần hồi đáp thông điệp lại máy khách. Ngoài ra, bộ máy AXIS còn tương tác với bộ quản lý cấu hình dịch vụ lưới. Bộ quản lý cấu hình này nhận các tệp cấu hình dưới dạng WSDD (Web Service Description Deployment). Thông qua bộ quản lý cấu hình dịch vụ lưới mà các tham số cho dịch vụ lưới được thiết lập và dịch vụ Web được hỗ trợ thêm các tính chất để trở thành dịch vụ lưới.

2. Trình chứa Globus. Đây là nơi quản lý các dịch vụ Web có trạng thái thông qua các thể hiện riêng biệt của kênh quản lý (instance handler), kho lưu (repository), cơ chế quản lý vòng đời (kích hoạt/huỷ dịch vụ...). Thông điệp dịch vụ sau khi được bộ máy dịch vụ Web xử lý sẽ chuyển tới kênh quản lý Pivot handler. Pivot handler chịu trách nhiệm tạo ra các thể hiện dịch vụ Web và kích hoạt các thao tác trên dịch vụ. Trong Globus Toolkit 3.2 sử dụng cơ chế thông điệp gói, tức là các tham số được gói lại được xử lý độc lập với nhau. RPCURIHandler là một thực thi của nền tảng AXIS để xử lý các thông điệp gói do bộ máy AXIS truyền tới.

Sau đó, RPCURIHandler liên hệ với kho lưu của thùng chứa thông qua kênh quản lý thể hiện dịch vụ để tìm được dịch vụ thích hợp và kích hoạt các hàm xử lý của dịch vụ.

Sau khi xưởng chế tác dịch vụ (service factory) tạo ra một thể hiện dịch vụ lưới, thùng chứa sẽ đồng thời liên kết nó với một kênh quản lý thể hiện (GSH-Grid Service Handler) riêng biệt và đăng kí kênh này với kho lưu (Container Repository). Kho lưu nắm mọi thông tin về các thể hiện dịch vụ có trạng thái và được liên hệ với các thành phần khác của thùng chứa như Pivot handler, dịch vụ lưới khác để phục vụ mục đích:

Xác định thể hiện dịch vụ và kích hoạt phương thức.

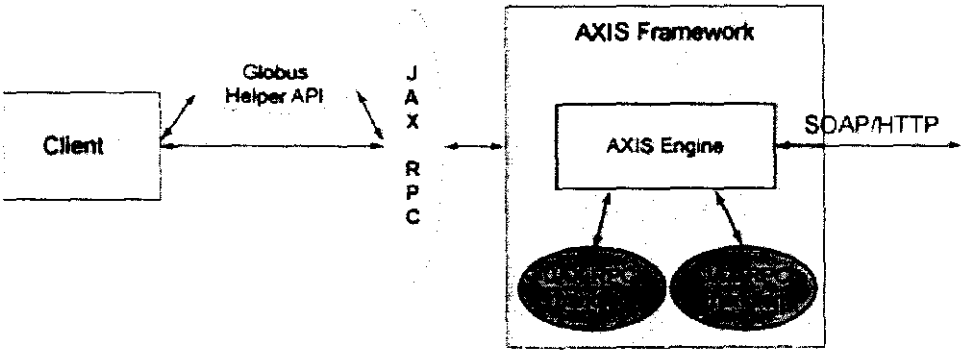
Thiết lập hay lấy thông tin về dịch vụ.

Kích hoạt hay dừng kích hoạt dịch vụ

Liên kết kênh quản lý dịch vụ với dịch vụ thực thi.

8.2.4.2. Mô hình dịch vụ lưới máy khách

Mô hình này thể hiện dưới hình vẽ:



Hình 8-21: Mô hình dịch vụ lưới phía máy khách

Phía máy khách cũng sử dụng mô hình lập trình dựa trên công nghệ gọi thủ tục từ xa thông qua một bộ xử lý JAX-RPC và bộ máy AXIS. Ngoài ra, Globus Toolkit 3.2 còn hỗ trợ rất nhiều các lớp trợ giúp để cho người sử dụng không phải quan tâm chi tiết mô hình lập trình theo chuẩn OGSI.

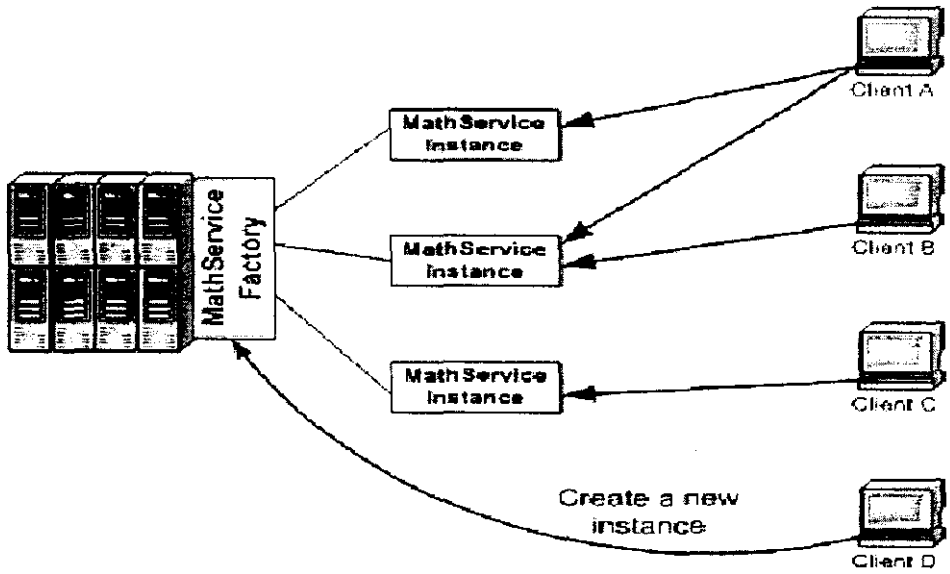
8.2.5. Các kỹ thuật triển khai khi xây dựng dịch vụ lưới

Phần này trình bày các kỹ thuật cài đặt dịch vụ lưới tuân theo chuẩn OGSA. Như đã trình bày ở phần trên, dịch vụ Web có hai nhược điểm là tính không trạng thái và tính không trong suốt. Tính không trạng thái tức là sau mỗi lần khởi động một dịch vụ thì các thông tin về lần hoạt động trước không được lưu giữ lại. Dịch vụ Web không thể áp dụng để cài đặt một dịch vụ hoàn chỉnh bao gồm một tập các dịch vụ con có liên quan hoặc trả về cho người sử dụng trạng thái hoạt động của các phiên làm việc trước của dịch vụ. Mặc dù dịch vụ Web có thể khắc phục phần về khả năng lưu trạng thái như sử dụng cơ sở dữ liệu trạng thái, đặt các cookies ở máy khách... nhưng dịch vụ Web vẫn gặp khó khăn về tính không trong suốt đối với người sử dụng. Tính không trong suốt tức là các thông tin về một phiên làm việc của dịch

vụ được lưu lại có thể được truy cập bởi bất kì người dùng nào hoặc trong khi một dịch vụ đang thực hiện thì người dùng khác có thể truy cập vào và làm hỏng việc thực thi. Dịch vụ lưới đã thể giải quyết các vấn đề còn tồn tại đối với dịch vụ Web thông qua một loạt các kỹ thuật trình bày dưới đây.

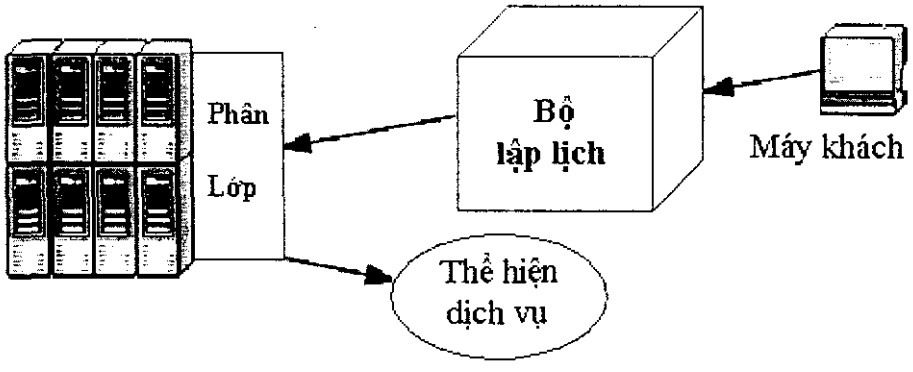
8.2.5.1. Cơ chế xưởng chế tác dịch vụ (Service Factory)

Vai trò của Service factory là sản sinh một thể hiện dịch vụ khi có yêu cầu từ phía máy khách. Điều này giải quyết được tình trạng quá tải yêu cầu và tính trong suốt của dịch vụ lưới. Cơ chế tự huỷ của dịch vụ cho phép một dịch vụ tự huỷ khi thực hiện xong, nhờ đó bảo toàn tính trong suốt cho người sử dụng. Ngoài ra, một thể hiện dịch vụ có thể được chia sẻ cho nhiều máy khách. Khi một máy khách cần thực hiện dịch vụ thì nó sẽ yêu cầu máy chủ (cụ thể là bộ máy Factory ở máy chủ) sinh ra một thể hiện (instance) và có toàn quyền sử dụng thể hiện này cho đến lúc thực hiện xong thì có thể huỷ thể hiện này đi. Điều này tạo nên tính "tạm thời" (transient) giữa các máy khách.



Hình 8-22: Mô hình xưởng chế tác

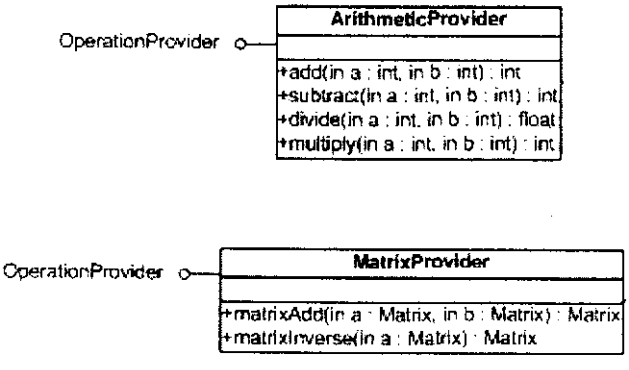
Các dịch vụ khai phá dữ liệu trên các tài nguyên của hệ thống BKGrid 2005 đều được áp dụng cơ chế xưởng chế tác. Cơ chế này đảm bảo tính riêng tư cho mỗi người sử dụng lưới. Mỗi khi có yêu cầu, thông qua bộ lập lịch một thể hiện duy nhất được sinh ra phục vụ riêng cho người sử dụng. Các thể hiện này được sinh ra với số lượng không bị hạn chế tuy nhiên tuỳ vào hiệu năng của từng tài nguyên mà người quản trị có thể cấu hình dịch vụ để giới hạn số lượng thể hiện dịch vụ cùng hoạt động trong một khoảng thời gian nào đó.



Hình 8-23: Cơ chế xưởng chế tác áp dụng cho dịch vụ phân lớp dữ liệu

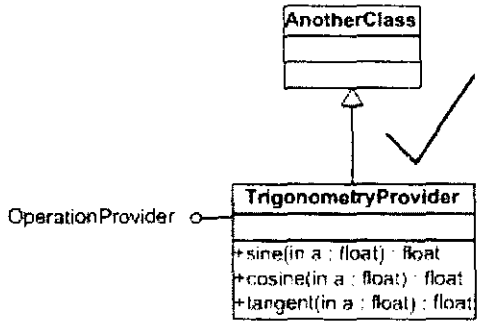
8.2.5.2. Cơ chế nhà cung cấp chức năng (Operation Provider)

Cơ chế xưởng chế tác dịch vụ có ưu điểm là đơn giản, dễ hiểu, tuy nhiên các dịch vụ theo cơ chế này khó có thể được mở rộng hay kế thừa được. Mô hình nhà cung cấp tác vụ giải quyết được vấn đề này bằng cách coi mỗi dịch vụ như một lớp tác vụ. Mỗi lớp tác vụ này gọi là một nhà cung cấp tác vụ (Operation Provider). Ví dụ như, dịch vụ MathFactory sẽ cung cấp hai chức năng. Một chức năng để xử lý số là ArithmeticProvider và một chức năng để xử lý ma trận là MatrixProvider.



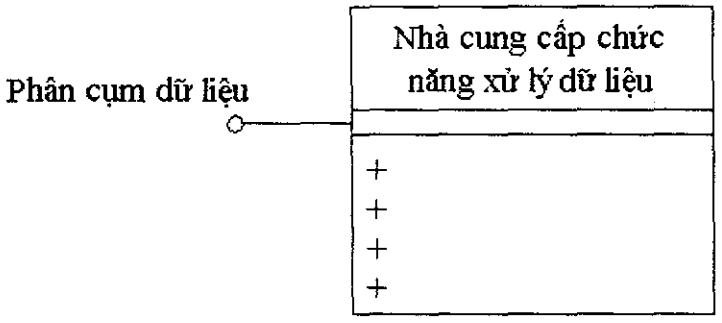
Hình 8-24: Hai nhà cung cấp dịch vụ riêng biệt

Hai chức năng này là tách riêng với nhau và người sử dụng có thể kế thừa được từ các chức năng này.



Hình 8-25: Nhà cung cấp dịch vụ được kế thừa từ lớp khác

Trong hệ thống BKGrid 2005, mỗi dịch vụ khai phá dữ liệu là một nhà cung cấp dịch vụ. Ví dụ như nhà cung cấp dịch vụ phân cụm dữ liệu Wekacluster cung cấp bốn chức năng tương ứng với bốn thuật toán phân cụm dữ liệu đầu vào là:



Hình 8-26: Nhà cung cấp chức năng phân cụm dữ liệu trong BKGrid 2005

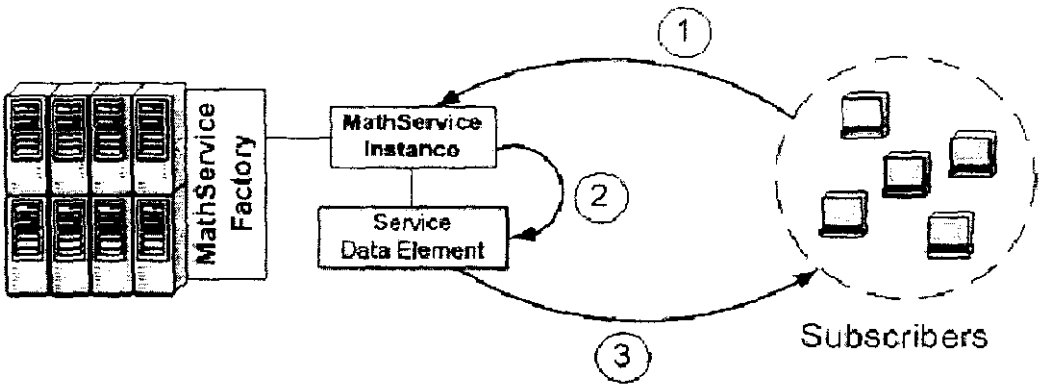
8.2.5.3. Cơ chế thông báo (Notification)

Đây là dịch vụ liên quan mật thiết tới dữ liệu dịch vụ. Dịch vụ khai báo đơn giản là việc thông báo tới một máy khách khi dịch vụ lưới có sự thay đổi. Hiện nay sử dụng nhất là mô hình Observer/Observale. Trong đó máy chủ làm nhiệm vụ theo dõi là Observable và máy chủ làm nhiệm vụ thông báo là Observer. Có hai kỹ thuật khai báo là đẩy (push) và pull (kéo).

+ Kiểu khai báo đẩy: Observer gửi thông điệp tới Observale yêu cầu được thông báo mỗi khi có sự thay đổi. Khi có thay đổi thì Observable thông báo cho Observer biết. Nếu Observer cần thông tin đó thì nó lại gửi yêu cầu nhận thông tin về sự thay đổi đó.

+ Kiểu khai báo kéo: trong kỹ thuật này, mỗi khi có sự thay đổi thì Observable gửi thông điệp đồng thời cả thông tin về sự thay đổi tới Observer.

Trong GT3.2, mô hình thông báo được thực hiện như sau:



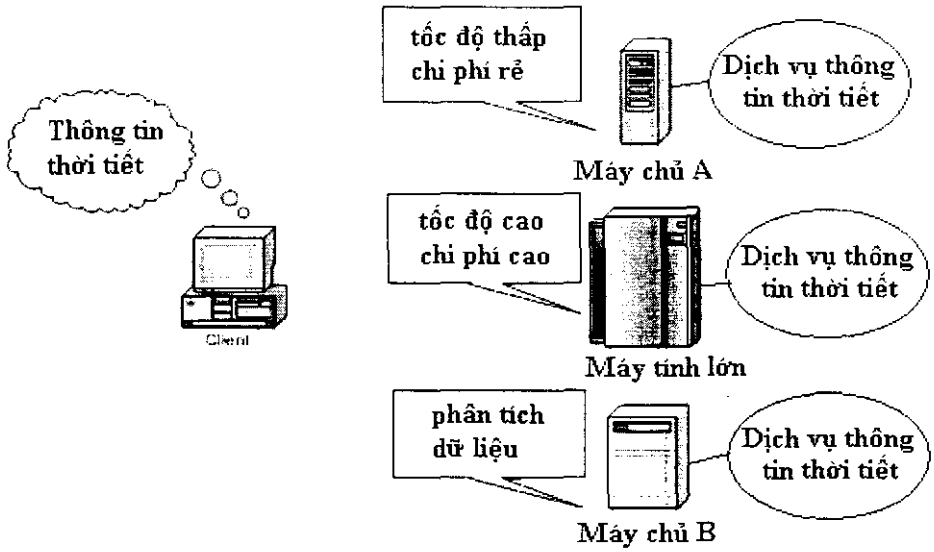
Hình 8-27: Mô hình thông báo trong GT3.2

1. addListener :Máy khách đăng ký được thông báo tới dịch vụ (bao gồm cả SDE xác định).
2. notifyChange :Khi có thay đổi xảy ra, dịch vụ sẽ yêu cầu SDE th ông báo tới máy khách.
3. deliverNotification: SDE thông báo cho máy khách về thay đổi đã xảy ra.

8.2.5.4. Dữ liệu dịch vụ (Service Data)

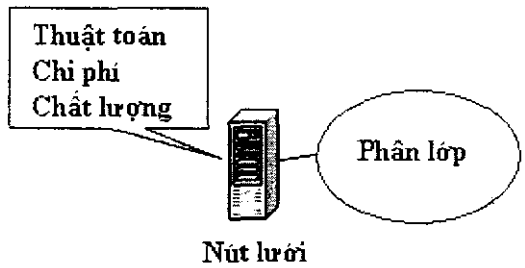
Một trong những đặc điểm quan trọng của dịch vụ Web là khả năng khai phá dịch vụ, cho phép người sử dụng tìm được địa chỉ URI (Uniform Resource Indentifier) của dịch vụ Web phù hợp yêu cầu sử dụng. Ví dụ như một người sử dụng muốn tìm một dịch vụ Web cung cấp thông tin về thời tiết của thành phố Hà Nội một ngày bất kì trong quá khứ. Như đã trình bày, người sử dụng có thể tìm kiếm trong thanh ghi UDDI. Cách thức để dịch vụ Web tự quảng bá là thông qua tệp mô tả dịch vụ Web (WSDL). Tuy nhiện, ngôn ngữ này lại thiên về tính kỹ thuật, tức là mô tả chi tiết

về phương thức kích hoạt, giao thức sử dụng giữa máy khách và máy chủ dịch vụ... Do đó, cần một ngôn ngữ khác đơn giản hơn trong việc xác định và phân loại dịch vụ. Giải pháp là dữ liệu dịch vụ (Service Data). Dữ liệu dịch vụ là tập các thông tin có cấu trúc gắn liền với dịch vụ lưới. Các thông tin này có thể dễ dàng được truy vấn. Chính vì vậy, các dịch vụ lưới được phân loại và đánh chỉ số theo các đặc tính của chúng. Cũng trong ví dụ nêu trên,có thể tồn tại ba dịch vụ cung cấp tính năng truy vấn thông tin thời tiết tại Hà Nội cho người sử dụng. Nhờ có dữ liệu dịch vụ mà người sử dụng chọn ra dịch vụ phù hợp nhất. Chẳng hạn, người sử dụng muốn có dịch vụ chất lượng cao, thông tin đa dạng và chính xác mà không cần quan tâm tới chi phí dịch vụ thì có thể chọn dịch vụ thứ nhất trên máy tính lớn. Nếu muốn sử dụng dịch vụ có chi phí vừa phải thì họ có thể liên hệ với máy chủ A. Trong trường hợp người sử dụng muốn có thêm các thông tin phân tích thời tiết thì dịch vụ trên máy chủ B là thích hợp hơn cả.



Hình 8-28: Sử dụng dữ liệu dịch vụ lựa chọn dịch vụ thích hợp

Các dịch vụ khai phá dữ liệu trong hệ thống BKGrid 2005 có sử dụng kỹ thuật dữ liệu dịch vụ để cung cấp các thông tin cần thiết cho bộ lập lịch. Các thông tin này bao gồm thông tin về thuật toán cài đặt để xử lý dữ liệu, chi phí người sử dụng phải trả, chất lượng dịch vụ. Bộ lập lịch dựa vào các thông tin do dịch vụ cung cấp cùng với các ràng buộc của người sử dụng sẽ đưa ra quyết định sử dụng dịch vụ trên các tài nguyên khác nhau.



Hình 8-29: Dữ liệu dịch vụ trong dịch vụ khai phá dữ liệu

Chi tiết cài đặt cũng như tương tác giữa dịch vụ và bộ lập lịch được trình bày chi tiết trong tài liệu kỹ thuật kèm với luận án tốt nghiệp.

Ngoài ra, còn một số kỹ thuật khác liên hỗ trợ cho quá trình quản lý dịch vụ lưới như:

+ Quản lý vòng đời: đảm bảo tính trong suốt của dịch vụ. Trong quá trình khởi tạo, hoạt động và huỷ dịch vụ, người phát triển ứng dụng đều có thể can thiệp và tiến hành các xử lý khác nhau theo mục đích của dịch vụ.

+ Cơ chế bản ghi (logging). Cơ chế này ghi lại mọi hoạt động của dịch vụ như thời gian khởi tạo, người khởi tạo, thời gian thực hiện... Các thông tin này được ghi ra dưới dạng tệp văn bản và người quản trị dịch vụ có thể dựa vào đó để quản lý tốt hơn dịch vụ của mình.

8.3. Triển khai dịch vụ Weka trên BKGrid 2005

8.3.1. Tổng quan về WEKA

Weka là một phần mềm nguồn mở về khai phá dữ liệu được phát triển bởi đại học University of Waikato nước New Zealand. "Weka" là từ viết tắt cho cụm từ Waikato Environment for Knowledge Analysis. Weka có thể được sử dụng ở nhiều cấp độ khác nhau. Cấp độ đơn giản của nó là ta có thể áp dụng các thuật toán của Weka để xử lý dữ liệu từ dòng lệnh. Nó bao gồm nhiều các công cụ dùng cho việc biến đổi dữ liệu, như các thuật toán dùng để rời rạc hóa dữ liệu. Weka có thể cung cấp các chức năng về tiền xử lý dữ liệu dùng trong quá trình tiền xử lý dữ liệu cho các thuật toán học máy (machine learning). Weka cung cấp tất cả các chức năng cơ bản của khai phá dữ liệu bao gồm các thuật toán về phân lớp (classifier), các thuật toán về tiền xử lý dữ liệu (filter), các thuật toán về phân cụm (cluster), các thuật toán về kết tập luật (association rule).

Toàn bộ hệ thống Weka được viết bằng ngôn ngữ Java, tạo sự thuận lợi cho việc lưới hoá Weka. Chi tiết việc thực hiện lưới hoá Weka được trình bày trong các chương sau; chương này tập trung trình bày các đặc điểm chính của ứng dụng này.

8.3.1.1. Khai phá dữ liệu (data mining)

Khai phá dữ liệu là tiến trình khám phá các tri thức đáng quan tâm, ví dụ như các mẫu hình (pattern), các kết tập luật (association rule), các biến đổi _ các cấu trúc đặc biệt và các cấu trúc dị thường từ một số lượng lớn dữ liệu được lưu trữ ở các cơ sở dữ liệu, data warehouses,...

Với tất cả các khả năng đó khai phá dữ liệu đã trở thành một trong những lĩnh vực đang rất được quan tâm trong ngành công nghiệp công nghệ thông tin. Ví dụ như nó có thể được sử dụng vào trong phân tích và nghiên cứu thị trường, hỗ trợ trong việc quản lý công việc hay ra các quyết định,...

Khai phá dữ liệu thực chất là một bước quan trọng trong khám phá tri thức. Thông thường trong tiến trình khám phá tri thức gồm các bước (hình 8-31):

+ Chuẩn hóa dữ liệu (data cleaning) có chức năng giải quyết các vấn đề đối với dữ liệu như nhiễu, lỗi, hoặc dữ liệu ở dạng không hợp lệ.

+ Tích hợp dữ liệu (data integration): thực hiện tích hợp nhiều các nguồn dữ liệu hỗn tạp lại với nhau.

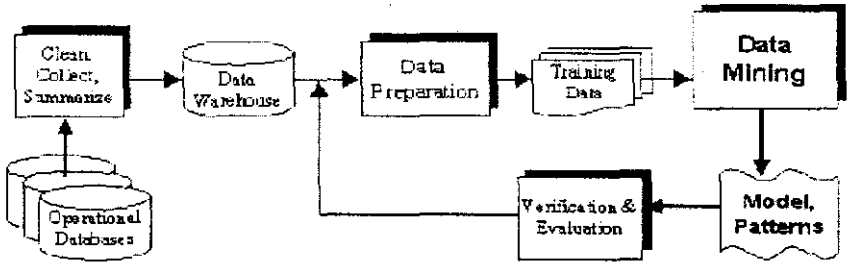
+ Lựa chọn dữ liệu (data selection): thực hiện lấy ra các dữ liệu phù hợp từ cơ sở dữ liệu cung cấp cho quá trình phân tích.

+ Biến đổi dữ liệu (data transformation): thực hiện chuyển đổi dữ liệu thành định dạng phù hợp với yêu cầu của quá trình khai phá thông qua hai thao tác đó là tổng hợp hay kết tập.

+ Khai phá dữ liệu (data mining): là một tiến trình đặc biệt trong đó có sử dụng các phương thức thông minh để trích rút ra các mẫu hình dữ liệu.

+ Sự đánh giá mẫu hình (pattern evaluation): thực hiện đánh giá sự đúng đắn đối với các mẫu hình quan tâm dựa vào các đánh giá đặc biệt.

+ Trình diễn tri thức (knowledge presentation): sử dụng các phương pháp cần thiết để thể hiện các tri thức đã được khai phá cho người dùng.



Hình 8-30: Mô hình tiến trình khám phá tri thức

Với các hệ thống cơ sở dữ liệu quan hệ rộng khắp và các data warehouse 4 tiến trình xử lý đầu của quá trình khám phá tri thức (Chuẩn hóa dữ liệu, tích hợp dữ liệu, lựa chọn dữ liệu, biến đổi dữ liệu) có thể được thực hiện thông qua việc xây dựng các data warehouse và thực hiện các thao tác xử lý OLAP trên các data warehouse đó.

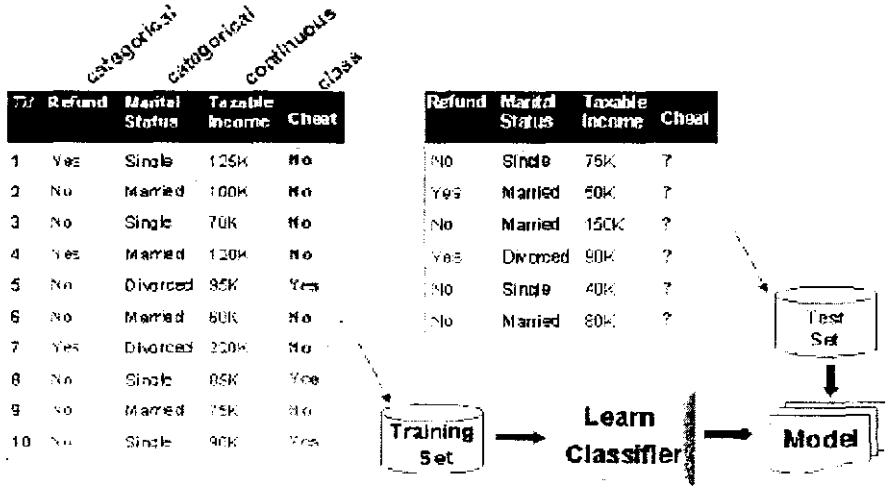
8.3.1.2. Các chức năng của WEKA

Các chức năng của gói phần mềm Weka hay chính là các chức năng trong khai phá dữ liệu.

- Chức năng phân lớp (classifier) của WEKA

+ Định nghĩa phân lớp (classifier)

Ví dụ có một tập dữ liệu về những người vay tiền để đầu tư kinh doanh của một ngân hàng. Trong đó mỗi một người tương ứng với một đối tượng của tập dữ liệu. Mỗi một đối tượng có các thuộc tính như: khả năng tái đầu tư của nhà kinh doanh đó, tình trạng hôn nhân, tình trạng đóng thuế của nhà kinh doanh, và thuộc tính quan trọng nhất mà các ngân hàng quan tâm đó là thuộc tính cho biết có nên cho nhà kinh doanh đó vay tiền hay không và thuộc tính này được gán nhãn là class.



Hình 8-31: Minh họa về quá trình thực hiện của phân lớp.

Hình trên mô tả quá trình học máy và sử dụng kết quả của quá trình học máy để kiểm thử các tập dữ liệu khác. Trong hình trên với một tập dữ liệu về các đối tượng như đã mô tả ở trên với các thuộc tính đã có đầy đủ giá trị được dùng làm dữ liệu huấn luyện cho quá trình học máy. Thông qua quá trình học máy kết quả thu được là một mô hình. Mô hình này sẽ được sử dụng để kiểm thử cho các tập dữ liệu khác khi mà các tập dữ liệu này có thuộc tính được gán nhãn là class có giá trị không xác định.

Quá trình phân lớp phân tích một tập dữ liệu-gọi là tập dữ liệu huấn luyện (ví dụ như một tập các đối tượng mà mỗi một đối tượng giá trị của thuộc tính class đã được xác định) và tạo ra một mô hình cho mỗi một lớp dựa vào các thuộc tính của các đối tượng trong tập dữ liệu đó. Cây quyết định hay chính là các luật phân lớp được tạo ra từ quá trình phân lớp. Ví dụ với phân lớp về các loại bệnh có thể được dùng để hỗ trợ việc dự đoán bệnh cho các bệnh nhân thông qua các triệu chứng của họ.

+ Các thuật toán về phân lớp trong WEKA

Giả sử rằng chúng ta có một dữ liệu là weather.arff mô tả các thông tin về thời tiết một vùng nào đó.

Mô tả về dữ liệu weather.arff.

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
```

```
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

File dữ liệu mô tả các thành phần chủ yếu về thời tiết như nhiệt độ, độ ẩm, tình trạng gió,... Đây là file dữ liệu có định dạng chuẩn dữ liệu đầu vào cho các thuật toán trong ứng dụng Weka. Mô tả về định dạng file *arff* được nêu trong phụ lục.

Weka sử dụng thuật toán J4.8, một biến thể của thuật toán C5.0, được chuyển dữ liệu trong weather.arff thành một cây quyết định. Chúng ta có thể thực hiện thuật toán dưới dạng dòng lệnh như sau:

```
java weka.classifiers.j48.J48 -t weather.arff
```

Tham số "-t" trong dòng lệnh phía trên chỉ ra cho thuật toán biết rằng đó là quá trình thực hiện huấn luyện (training) đối với dữ liệu đầu vào đó là tham số tiếp theo. Ý nghĩa các tham số của thuật toán J48 được mô tả trong bảng 8-1.

Kết quả thực hiện trên console được mô tả trên hình 8-32. Kết quả thực hiện của thuật toán ở dạng đồ hoạ được trình bày trong hình 8-33.

Như kết quả cho thấy, gốc của cây là thuộc tính quang cảnh (outlook), và sau đó mức thứ 2 của cây là các thuộc tính độ ẩm (humidity) và tình trạng gió (windy). Trong cấu trúc cây, dấu ":" thể hiện phân cách giữa lá -tên của lớp- và số các thể hiện thuộc lớp đó. Tiếp theo sau đó là tổng số các nút của cây (kích thước của cây).

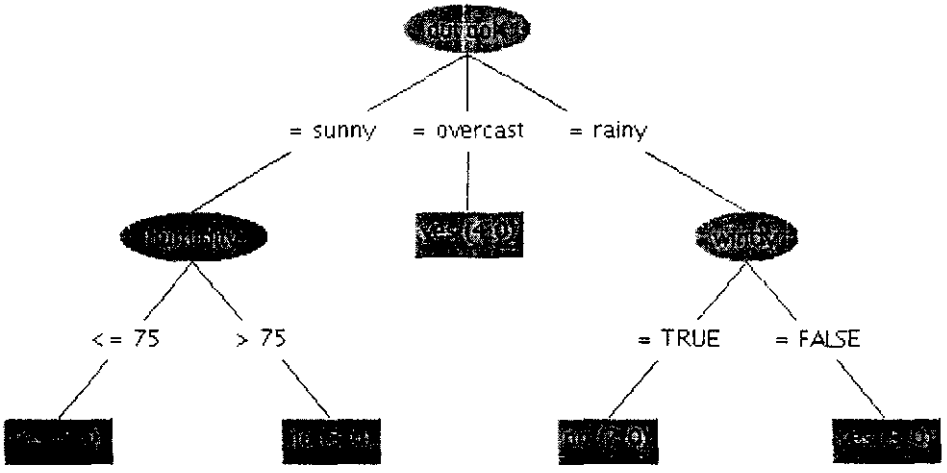
Tham số	Chức năng
-U	Sử dụng mô hình cây không tỉa nhánh
-C	Xác định ngưỡng tin cậy cho qua trình tỉa nhánh
-M	Xác định số lượng nhỏ nhất các thể hiện trên một lá
-R	Sử dụng việc tỉa nhánh để giảm số lượng lỗi
-N	Xác định số lượng fold sử dụng cho việc giảm lỗi.
-S	Chỉ sử dụng trong phân chia nhị phân

Bảng 8-1: các tham số đặc biệt sử dụng cho thuật toán J48

```
J48 pruned tree
------------------

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :     5
Size of the tree :      8


=== Error on training data ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
Mean absolute error                      0
Root mean squared error                  0
Total Number of Instances               14

=== Confusion Matrix ===

 a b   <-- classified as
 9 0 | a = yes
 0 5 | b = no

=== Stratified cross-validation ===

Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %
Mean absolute error                      0.3036
Root mean squared error                  0.4813
Total Number of Instances               14

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

Hình 8-32: Kết quả thực hiện thuật toán J48 với dữ liệu đầu vào là weather.arff

Phần tiếp theo của kết quả đưa ra các đánh giá về việc dự đoán trước các kết quả đạt được, và công đoạn này được đảm nhận bởi mô đun evaluation của Weka. Các đánh giá đầu tiên đưa ra được nhận được từ phần dữ liệu huấn luyện (training) .



Hình 8-33: Hiển thị kết quả được thực hiện dưới dạng cây quyết định

Các đánh giá được đưa ra này có độ lạc quan cao và nó thể hiện mức độ đúng đắn cao về dự đoán trước kết quả đạt được. Trong trường hợp có lỗi xảy ra đối với sự phân lớp, mô đun evaluation đưa ra các kết quả tính toán theo xác suất về giá trị sai số trung bình tuyệt đối (mean absolute) và căn bậc 2 của bình phương trung bình - sai phương bậc 2 (root meansquared). Trong ví dụ trên thì cả 2 giá trị đều nhận giá trị là 0 vì các xác suất phân lớp đúng cho các thể hiện đều là 1 và xác suất phân lớp sai cho các thể hiện đều là 0, bởi vì việc phân lớp được thực hiện hoàn toàn chính xác.

Tổng kết các kết quả đạt được trong quá trình huấn luyện thể hiện trong một ma trận gọi là confusion matrix. Trong ma trận này thể hiện bao nhiêu thể hiện của mỗi lớp này được phân nhầm sang lớp kia, và như kết quả thể hiện thì chỉ có các phần tử của đường chéo chính của ma trận là nhận các kết quả khác 0 là vì do quá trình phân lớp được thực hiện hoàn toàn chính xác.

Đoạn cuối cùng của kết quả thể hiện các kết quả đạt được bằng việc sử dụng phương pháp đánh giá chéo ten-fold (phụ lục). Chiến lược này được thực hiện một cách tự động trong quá trình thực hiện thuật toán nếu như đầu vào không xác định file kiểm thử(test file). Như kết quả đạt được, có hơn 30% thể hiện (5 trong 14) bị phân lớp sai trong quá trình đánh giá chéo.

Khi đánh giá các mô hình dự đoán số (numeric), Weka đưa ra các cách đánh giá khác. Ví dụ, có một tập dữ liệu về sự hoạt động của các CPU là *cpu.arff*.

Bằng việc thực hiện lệnh sau:

```
java weka.classifiers.m5.M5Prime -t cpu.arff
```

```
Pruned training model tree:

MMAX <= 14000   : LM1 (141/4.18%)
MMAX >  14000   : LM2 (68/51.8%)

Models at the leaves (smoothed):

  LM1:  class = 4.15
              - 2.05vendor=honeywell,ipl,ibm,cdc,ncr,basf,
                  gould,siemens,nas,adviser,sperry,amdahl
              + 5.43vendor=adviser,sperry,amdahl
              - 5.78vendor=amdahl + 0.0063*MYCT
              + 0.0015*MMIN + 0.0034*MMAX
              + 0.552CACH + 1.14CHMIN + 0.0945CHMAX
  LM2:  class = -113
              - 56.1vendor=honeywell,ipl,ibm,cdc,ncr,basf,
                  gould,siemens,nas,adviser,sperry,amdahl
              + 10.2vendor=adviser,sperry,amdahl
              - 10.9vendor=amdahl
              + 0.012MYCT + 0.0145MMIN + 0.0089MMAX
              + 0.809CACH + 1.29CHMAX

=== Error on training data ===

Correlation coefficient                  0.9853
Mean absolute error                     13.4072
Root mean squared error                 26.3977
Relative absolute error                 15.3431 %
Root relative squared error             17.0985 %
Total Number of Instances              209

=== Cross-validation ===

Correlation coefficient                  0.9767
Mean absolute error                     13.1239
Root mean squared error                 33.4455
Relative absolute error                 14.9884 %
Root relative squared error             21.6147 %
Total Number of Instances              209
```

Hình 8-34: Kết quả thực hiện của thuật toán M5 với dữ liệu là file cpu.arff

Kết quả thực hiện của thuật toán thật đáng ngạc nhiên, gốc cây quyết định là cây quyết định nhị phân cấp 1, phân lớp theo thuộc tính MMAX. Gắn vào gốc đó là 2 mô hình tuyến tính, mỗi mô hình tương ứng với một lá. Cả 2 đều liên quan tới thuộc tính dạng tên/danh (nominal) gọi là "*vendor*"-nhà cung cấp.

Biểu thức vendor=adviser, sperry,amdahl được mô tả như sau: nếu nhà cung cấp (*vendor*) là một trong các nhà cung cấp sau: adviser, sperry, hoặc amdahl, thì biểu thức cho kết quả là 1, ngược lại thì nó cho kết quả là 0. Sau đây là mô tả về mô hình cây đạt được của thuật toán: cũng như với tất cả các giá trị đầu ra của các thuật toán về cây quyết định, phần đầu tiên là kết quả nhận được từ phần dữ liệu huấn luyện và phần thứ 2 là các đánh giá có được từ việc dùng chiến lược đánh giá chéo ten-fold.

Ở các ví dụ trên, tham số "-t" được sử dụng để xác định file dữ liệu huấn luyện cho các thuật toán học máy. Sau đây là bảng các tham số lựa chọn khác nhau được sử dụng trong các thuật toán học máy.

Tham số	Chức năng
-t <file huấn luyện>	Xác định file dữ liệu trong quá trình huấn luyện
-T <file kiểm thử >	Xác định file dữ liệu dùng trong quá trình kiểm thử. Nếu tham số này không được lựa chọn thì quá trình đánh giá được thực hiện theo chiến lược đánh giá chéo với file dữ liệu là file dữ liệu huấn luyện.
-c <chỉ số của thuộc tính *class*>	Tham số này xác định chỉ số của thuộc tính *class* trong file dữ liệu.
-x <số các fold>	Tham số này chỉ ra số lượng fold được phân chia từ file dữ liệu huấn luyện để sử dụng trong quá trình đánh giá chéo.
-v	Đưa ra các kết quả từ sự thực hiện của thuật toán không thuộc về lĩnh vực thống kê đối với file dữ liệu huấn luyện.
-d	Tham số này chỉ ra địa chỉ lưu trữ kết quả của quá trình thực hiện thuật toán. Kết quả đó được lưu dưới dạng file **.model*
-o	Tham số này thì lại xác định rằng chỉ đưa ra cac thông tin thuộc về lĩnh vực thống kê đối với dữ liệu đầu vào.
-l	Tham số này để chỉ ra các kết quả đầu ra là về các thông tin về sự sửa lỗi.
-k	Tham số này dùng để đưa ra các thông tin về lý thuyết thống kê.
-p	Tham số này dùng để đưa ra các dự đoán về các thực thể kiểm thử.
-r	Tham số này đưa ra các thông tin về lũy tích dự trữ phân tán.

Bảng 8-2: Các tham số cơ bản sử dụng trong các thuật toán phân lớp.

Các tham số được cho ở bảng trên quyết định rằng dữ liệu nào sẽ được sử dụng để huấn luyện và dữ liệu nào được dùng để kiểm thử, thuật toán học máy phân lớp đó được đánh giá như thế nào, và các thông tin thống kê nào sẽ được kết suất ở đầu ra của thuật toán. Người dùng có thể đưa dữ liệu từ bên ngoài vào để kiểm thử quá trình thực hiện của thuật toán bằng sử tham số -T, ngoài ra sự thực hiện của thuật toán còn được đánh giá thông qua chiến lược đánh giá chéo với dữ liệu chính là file dữ liệu dùng trong quá trình huấn luyện. Trong một số file dữ liệu đầu vào dạng ARFF thì thuộc tính *class* không nằm ở vị trí cuối cùng, người dùng cần phải khai báo lại vị trí của thuộc tính *class* đó trong file dữ liệu đó thông qua tham số -c

Khi đánh giá chéo ten-fold được thực hiện, với tập dữ liệu lớn người dùng muốn giảm số các fold mặc định trong đánh giá chéo là 10 xuống thì tham số được sử dụng ở đây là –x, số theo sau ngay tham số -x sẽ chính là số fold được phân chia trong đánh giá chéo mà người dùng mong muốn. Kết quả đầu ra của thuật toán có thể được lưu lại dưới dạng file *.model bằng cách sử dụng tham số -d, kết quả đầu ra được tạo ra từ việc huấn luyện dữ liệu đó sẽ lưu vào địa chỉ được chỉ ra sau tham số -d. Ngược lại, nếu người dùng muốn cập nhật lại thông tin của file *.model đó hoặc dùng file *.model đó như một tham số đầu vào để thực hiện quá trình gán nhãn (labeling) cho một file dữ liệu khác, tham số -l sẽ được sử dụng để tải về file *.model đó. Người dùng thường muốn biết giá trị dự đoán của thuộc tính class của các thể hiện trong file dữ liệu kiểm thử. Tham số -p được sử dụng để in ra số các thể hiện trong file dữ liệu kiểm thử, mức độ tin tưởng đối với sự dự đoán, và giá trị của thuộc tính class được dự đoán trên các nút lá của cây quyết định (chú ý rằng tham số -p chỉ được sử dụng trong kiểm thử sự thực hiện của thuật toán).

Sau đây là danh sách tất cả các thuật toán học máy có trong gói Weka. Trong đó có mô tả một cách sơ lược về các thuật toán như tên, các tham số quan trọng đặc biệt và tác dụng của từng tham số đó đối với các thuật toán

Thuật toán dạng	Thuật toán thực hiện	Tham số	Chức năng của tham số
Majority/average predictor	weka.classifiers.ZeroR	Không có	
1R	weka.classifiers.OneR	-B<>	Xác định kích thước tối thiểu của bucket
Naive Bayes	weka.classifiers. NaiveBayes	-K	Sử dụng để ước lượng mật độ của các thể hiện
Decision table	weka.classifiers.DecisionTable	-X<>	Xác định số lượng fold sử dụng trong đánh giá chéo
		-S<>	Xác định ngưỡng để dừng việc tìm kiếm
Instance-based learner	weka.classifiers.IBk	-D	Được xác định bằng nghịch đảo của khoảng cách
		-F	Đượcxác định bằng (1-khoảng cách)

		-X	Chỉ ra rằng sử dụng đánh giá chéo
C4.5	weka.classifiers.trees.J48	Đã mô tả ở trên	
PART rule learner	weka.classifiers.trees.PART	Như đối với J4.8	
Support vector machine	weka.classifiers.SMO	C<>	Xác định cận trên cho các weight
		-E<>	Xác định cấp của hàm đa thức sử dụng trong thuật toán
Linear regression	weka.classifiers.Linear regression	-S<>	Xác định phương thức lựa chọn các thuộc tính
M5' model tree learner	weka.classifiers.m5. M5Prime	O<>	Xác định kiểu mô hình
		-U	Sử dụng cây không chỉnh sửa
		-F<>	Xác định yếu tố bị chặt bỏ
		-V<>	Xác định cách diễn đạt kết quả
Locally weighted regression	weka.classifiers.LWR	-K<>	Xác định số lượng các láng giềng của một thể hiện
One-level decision trees	weka.classifiers. DecisionStump	Không có	

Bảng 8-3: Danh sách các thuật toán về phân cụm trong Weka

Thuật toán đơn giản nhất trong các thuật toán học máy về lĩnh vực phân lớp là thuật toán ZeroR: thuật toán đưa ra danh sách các giá trị của thuộc tính class, và đưa ra giá trị trung bình của các thuộc tính này nếu như chúng là thuộc tính dạng số. Thông thường thuật toán được dùng để làm chuẩn cho các thuật toán khác.

Các thuật toán được sắp xếp trên bảng trên theo thứ tự tăng dần độ phức tạp sự thực hiện của thuật toán. Thuật toán tiếp theo đó là OneR, thuật toán tạo ra các luật đơn giản chỉ dựa vào một thuộc tính duy nhất. Với OneR các luật được tạo ra phải thỏa mãn luật đó phù hợp với ít nhất một số lượng xác định các thể hiện có trong dữ liệu huấn luyện (giá trị ngưỡng đó được xác định mặc định là 6).

NaiveBayes thực hiện phân lớp theo phân bố Naive Bayesian. Thông thường thuật toán thực hiện xây dựng mô hình cho các thuộc tính dạng số, với tham số -K thuật toán sẽ thực hiện đưa ra các ước lượng và đánh giá về mức độ trù mật của sự tập trung của các thuộc tính.

Thuật toán học tiếp theo đó là thuật toán tạo bảng quyết định (DecisionTable), việc thực hiện tạo bảng quyết định sử dụng phương pháp *wrapper* để xây dựng các tập thuộc tính. Phương pháp *wrapper* sử dụng phương pháp tìm kiếm best-first. Số lượng tập các thuộc tính được tạo ra bởi phương pháp wrapper được điều khiển bởi tham số -S (số lượng tập các thuộc tính được tạo ra mặc định là 5). Số lượng fold sử dụng trong đánh giá chéo sự thực hiện của thuật toán được điều khiển theo tham số –X (mặc định sử dụng leave-one-out có nghĩa là mỗi một thuộc thể hiện tương ứng với một fold). Thông thường, bảng quyết định được tạo ra từ quá trình huấn luyện được sử dụng để kiểm thử các thể hiện trong tập dữ liệu kiểm thử có phù hợp với một phần tử nào trong bảng quyết định đó không. Tuy nhiên, khi sử dụng tham số -I, thuật toán sẽ lựa chọn phần tử gần đúng nhất đối với các thể hiện trong tập dữ liệu kiểm thử. Thuật toán IBk một trong những thuật toán thuộc lớp các thuật toán *instance-based learn* thực hiện quá trình phân lớp thông qua quá trình thực hiện *k-nearest-neighbors* (lựa chọn k thể hiện gần nhất với một thể hiện nào đó trong quá trình phân lớp). k nhận giá trị mặc định là 1, tuy nhiên giá trị này có thể được tùy chọn thông qua sử dụng tham số -K hoặc sử dụng phương pháp đánh giá chéo leave-one-out.

Chúng ta đã tìm hiểu về thuật toán J4.8 và các tham số của nó; thuật toán PART cũng được thực hiện tương tự như vậy. Tuy nhiên, thuật toán PART là thuật toán thực hiện tỉa cành để giảm lỗi (*reduced-error pruning)*, quá trình này sẽ làm giảm kích thước của cây quyết định so với sự thực hiện của thuật toan J4.8 và nó cũng đồng nghĩa với việc số luật tạo ra trong quá trình thực hiện thuật toán cũng giảm đi. Với PART thì quá trình thực hiện của thuật toán sẽ được rút ngắn hơn đối với J4.8 tuy nhiên quá trình tỉa cành để giảm lỗi sẽ làm giảm độ chính xác bởi vì số lượng các thể hiện trong dữ liệu huấn luyện bị cắt giảm. Tuy nhiên, khi thực hiện một khối lượng dữ liệu tương đối lớn với thuật toán PART thì sự cắt giảm dữ liệu đó là không đáng kể và thuật toán vẫn đưa ra các kết quả hoàn hảo. Lớp SMO thực hiện thuật toán học máy (*sequential minimal optimization)*. Mặc dù đây là thuật toán thực hiện nhanh nhất trong các thuật toán học máy hỗ trợ phương pháp véc tơ, các thuật toán học máy *sequential minimal optimization* thường thực hiện chậm đặc biệt với các tập dữ liệu không thể chuyển từ dạng phi tuyến sang dạng tuyến tính. Tiếp theo là ba thuật toán học máy trong

lĩnh vực đưa ra các dự đoán thuộc về số. Thứ nhất đó là thuật toán M5¢, thời gian thực hiện của thuật toán tỷ lệ tuyến tính với số các thuộc tính. Thuật toán thứ hai được thực hiện bởi lớp LWR . Sự thực hiện của thuật toán thông qua việc tính khoảng cách của thể hiện kiểm thử tới k phần tử gần với nó nhất. k được xác định thông qua tham số –K.

Thuật toán cuối cùng là DecisionStump - thuật toán xây dựng các cây quyết định một mức. Nó giải quyết trường hợp các giá trị bị thiếu bằng cách thêm một nhánh vào gốc cây quyết định, hay nói cách khác nó giải quyết trường hợp giá trị thiếu bằng cách tạo ra thêm giá trị.

- Chức năng tiền xử lý dữ liệu (filter) của WEKA

Một gói quan trọng nữa trong Weka là *filters*. Ví dụ sau thực hiện xóa các thuộc tính được xác định là các thuộc tính thứ 1 và thứ 2 của file dữ liệu weather.arff

java weka.filters.AttributeFilter -R 1,2 -i weather.arff

Kết quả thực hiện được thể hiện như hình sau:

```
@relation weather-weka.filters.AttributeFilter-R1_2

@attribute humidity real
@attribute windy {TRUE,FALSE}
@attribute play {yes,no}

@data

85,FALSE,no
90,TRUE,no
...
```

Hình 8-35: Kết quả thực hiện tiền xử lý dữ liệu

Như kết quả trả về thì các thuộc tính thứ 1 và thứ 2 là outlook và temperature đã được xóa. Kết quả thực hiện có thể được lưu lại vào một file mới là *weather.new.arff* bằng thực hiện lệnh sau:

java...AttributeFilter -R 1,2 -i weather.arff -o weather.new.arff

Cách sử dụng đối với tất cả các thuật toán trong gói filter là như nhau. File đầu vào được xác định bằng tham số -i và kết quả thực hiện được lưu lại trong một file mới bằng tham số -o. Chỉ số của thuộc tính class được xác định bằng tham số -c.

Thuật toán Filter đọc vào file *.ARFF, biến đổi chúng, và lưu kết quả thực hiện vào một file *.arff khác. Bảng 8-4 thể hiện danh sách các thuật toán trong lĩnh vực tiền xử lý dữ liệu mà Weka cung cấp.

Thuật toán filter	Tham số	Chức năng của thuật toán
weka.filters.AddFilter	-C <>	Xác định chỉ số của tham số mới
	-L <>	Xác định nhãn cho các thuộc tính định

	-N <>	Xác định tên của thuộc tính mới
weka.filters.AttributeSelectionFilter	-E <>	Xác định kiểu loại đánh giá
	-S <>	Xác định kiểu tìm kiếm
	-T <>	Xác định ngưỡng để loại bỏ các thuộc tính
weka.filters.AttributeFilter	-R <>	Xác định các thuộc tính sẽ được loại bỏ
weka.filters.DiscretizeFilter	-B <>	
	-R <>	Xác định các thuộc tính được rời rạc hóa trong quá trình filter.
	-D	Đưa ra các thuộc tính dạng nhị phân
weka.filter.MakeIndicatorFilter	-C <>	Xác định chỉ số của thuộc tính
	-V <>	Xác định giá trị chỉ số
	-N	Đưa ra các thuộc tính dạng định danh
weka.filter.MergeTwoValuesFilter	-C <>	Xác định chỉ số của thuộc tính
	-F <>	Xác định chỉ số giá trị đầu tiên
	-S <>	Xác định chỉ số của giá trị thứ 2
weka.filters.NominalToBinaryFilter	-N	Đưa ra các thuộc tính dạng định danh
weka.filters.ReplaceMissingValuesFilter		
	-C <>	Xác định chỉ số thuộc tính
weka.filters.InstanceFilter	-S <>	Xác định giá trị thuộc dạng số
	-L <>	Xác định các giá tri thuộc dạng định
	-V	Invert matching sense
weka.filters.SwapAttributeValuesFilter	-C <>	Xác định chỉ số của thuộc tính
	-F <>	Xác định chỉ số giá trị đầu tiên
	-S <>	Xác định chỉ số của giá trị thứ 2
weka.filters.NumericTransformFilter	-R <>	Xác định các thuộc tính được chuyển đổi
	-V	Invert matching sense
	-C <>	Xác định lớp java
	-M <>	Xác định phương thức thực hiện biến đổi.
weka.filters.SplitDatasetFilter	-R <>	Xác định phạm vi của các thể hiện
	-V	Invert matching sense

-N <>	Xác định số lượng các fold
-F <>	Xác định fold được trả lại

Bảng 8-4: các thuật toán về filter trong Weka

Thuật toán đầu tiên là AddFilter nó được dùng để thêm một thuộc tính vào một vị trí xác định trong một file *.arff. Khi thực hiện thuật toán AddFilter với một file *.arff thì tất cả các thể hiện trong file đó đều có thuộc tính mới đó và giá trị của thuộc tính đó ở các thể hiện là chưa được thiết lập. Nếu có sử dụng tham số -L và theo sau là một tập giá trị thuộc tính dạng định danh (các giá trị đó cách nhau bởi dấu phẩy) thì thuộc tính mới thêm đó là một thuộc tính dạng định danh, trong trường hợp còn lại thì thuộc tính này sẽ được mặc định là thuộc tính dạng số. Tên của thuộc tính mới đó được thiết lập thông qua tham số -N.

Thuật toán AttributeFilter như đã được sử dụng ở ví dụ trên. Một trong những thuật toán quan trọng sử dụng trong filter là thuật toán DiscretizeFilter. Nó bao gồm 2 phương pháp rời rạc hóa có giám sát và không có giám sát. Số mẫu được lựa chọn trong quá trình rời rạc hóa tương ứng với số khoảng lấy mẫu, số khoảng lấy mẫu này được thiết lập thông qua tham số -B. Tuy nhiên, khi tham số –O được sử dụng thì số lượng khoảng lấy mẫu được xác định một cách tự động bằng việc sử dụng thủ tục đánh giá chéo nhằm cực tiểu hóa sự sai lệch đối với dữ liệu trong quá trình lấy mẫu. Trong quá trình rời rạc hóa có giám sát nếu có sử dụng tham số -c để chỉ ra vị trí của thuộc tính class thì thuật toán sẽ được thực hiện bởi phương pháp MDL của Fayyd và IRani (1993).

Đối với một số thuật toán học máy chỉ xử lý được dữ liệu về những thể hiện có các thuộc tính được mô tả dưới dạng các giá trị nhị phân. Khi đó filter cung cấp cho chúng ta thuật toán NominalToBinaryFilte để chuyển các thuộc tính từ dạng định danh sang dạng nhị phân, thay thế một thuộc tính dạng định danh có n giá trị khác nhau bởi (n-1) thuộc tính dạng nhị phân.

Để giải quyết việc loại bỏ các thuộc tính chưa được thiết lập giá trị trước khi áp dụng các thuật toán học máy bằng cách dùng thuật toán ReplaceMissingValuesFilter dùng để loại bỏ các thuộc tính đó.

Thuật toán InstanceFilter dùng để loại bỏ các thể hiện khỏi một cơ sở dữ liệu nào đó khi mà giá trị một thuộc tính nào đó của nó vượt quá hoặc nhỏ hơn một giới hạn nào đó. Trong một số các ứng dụng cần phải chuyển đổi thuộc tính dạng số trước khi áp dụng thuật toán học máy ví dụ như bình phương giá trị của các thuộc tính dạng số. Các thao tác này được thực hiện thông qua thuật toán NumericTransformFilter, nó sẽ thực hiện việc thay đổi với tất cả các thuộc tính dạng số được lựa chọn. Ví dụ để thực hiện bình phương giá trị của tất cả các thuộc tính dạng số ta thực hiện dòng lệnh sau: java weka.filters.NumericTransformFilter -C java.lang.Math -M sqrt...

trong đó tham số -C xác định địa chỉ gói chứa chức năng cần sử dụng, -M xác định phương thức biến đổi cần dùng ở ví dụ này thì đó là hàm sqrt

Thuật toán cuối cùng được xem xét đó là thuật toán AttributeSelectionFilter. Việc thực hiện thuật toán này cho phép người dùng có thể lựa chọn ra tập các thuộc tính của các thể hiện từ một file dữ liệu phù hợp với mục đích công việc của người sử dụng. Để thực hiện thuật toán này nó cần phải sử dụng một số lớp trong gói

weka.attributeSelection. Sử dụng tham số –c để thiết lập chỉ số class cho thuộc tính mà người dùng quan tâm. Dùng tham số –E để chỉ định cho thuật toán filter phương pháp đánh giá các thuộc tính, hay thiết lập các thuộc tính, phương pháp đó được thực hiện bởi các lớp trong gói weka.attributeSelection; ngoài ra cần xác định kỹ thuật tìm kiếm cần sử dụng thông qua tham số –S.

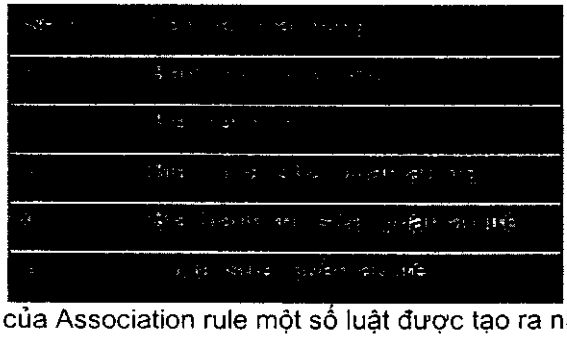
- Chức năng tập hợp luật (association rules) của WEKA

+ Định nghĩa chức năng kết tập luật (association rules)

Tập hợp là sự khám phá những mối quan hệ giữa một tập các phần tử dữ liệu nào đó. Các tập hợp thường được thể hiện dưới các luật thông qua các điều kiện về giá trị của các thuộc tính của dữ liệu. Các luật được tạo ra được biểu diễn dưới dạng X => Y, trong đó X là tập hợp các điều kiện còn Y là kết quả có được thông qua các điều kiện đó. Phân tích tập hợp luật được dùng rộng rãi trong quá trình phân tích dữ liệu cho các chiến lược tiếp thị, thiết kế phân loại hoặc trong các công việc thực hiện ra quyết định.

Một ví dụ đơn giản về việc áp dụng Association rule vào phân tích mối quan hệ giữa các mặt hàng trong quá trình mua bán của khách hàng.

Giả sử có một file dữ liệu về danh sách các mặt hàng được mua bởi từng khách hàng như sau:



Khi áp dụng thuật toán của Association rule một số luật được tạo ra như:

{sữa} {quần áo trẻ} : có nghĩa là khi người tiêu dùng mua "sữa" thì mặt hàng được mua cùng rất có thể là "quần áo trẻ".

{quần áo trẻ, sữa} {bia} : có nghĩa là khi người tiêu dùng mua " quần áo trẻ và sữa" thì mặt hàng được mua cùng rất có thể là "bia".

+ Các thuật toán về tập hợp luật trong WEKA

Weka bao gồm cả các thuật toán về lĩnh vực xây dựng tập hợp luật, các thuật toán đó có trong gói weka.associations trong đó thuật toán đáng quan tâm nhất đó là thuật toán APRIORI. Thực hiện sự thực thi của thuật toán thông qua dòng lệnh.

java weka.associations.Apriori –t weather.nominal.arff

trong đó dữ liệu đầu vào weather.nominal.arff là một file dữ liệu về thời tiết. Mục đích sự thực hiện của thuật toán là với các dữ liệu đã có về thời tiết được cung cấp trong file weather.nominal.arff nó sẽ đưa ra một số quy luật về thời tiết với một độ chính xác nào đó. (Thuật toán APRIORI chỉ thực hiện với dữ liệu mà các thuộc tính của nó thuộc dạng định danh. Kết quả thực hiện:

```
Apriori
=======

Minimum support: 0.2
Minimum confidence: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. humidity=normal windy=FALSE 4 ==> play=yes 4 (1)
2. temperature=cool 4 ==> humidity=normal 4 (1)
3. outlook=overcast 4 ==> play=yes 4 (1)
4. temperature=cool play=yes 3 ==> humidity=normal 3 (1)
5. outlook=rainy windy=FALSE 3 ==> play=yes 3 (1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 (1)
7. outlook=sunny humidity=high 3 ==> play=no 3 (1)
8. outlook=sunny play=no 3 ==> humidity=high 3 (1)
9. temperature=cool windy=FALSE 2 ==> humidity=normal play=yes 2 (1)
10. temperature=cool humidity=normal windy=FALSE 2 ==> play=yes 2 (1)
```

Hình 8-36: Kết quả thực hiện của thuật toán APRIORI với dữ liệu weather.nominal.arff

Kết quả thực hiện của thuật toán là tập hợp các luật. Số đứng trước ký hiệu ==> trong các luật được tạo ra nhằm thể hiện số lượng các thể hiện trong tập dữ liệu có các thuộc tính thỏa mãn tiền đề của luật đó. Tương tự thì số theo sau các luật chính là số các thể hiện trong tập dữ liệu thỏa mãn luật đó. Ví dụ luật có được từ thực hiện thuật toán Apriori với file dữ liệu weather.nominal.arff

humidity=normal windy=FALSE 4 ==> play=yes 4 (1)

Giá trị trong ngoặc ở cuối mỗi luật chính là mức độ tin cậy đối với luật được tạo.

Thứ tự các luật được tạo ra từ thuật toán APRIORI được sắp xếp giảm dần theo cả độ tin cậy và số lượng các thể hiện trong dữ liệu phù hợp với luật đó. Thuật toán APRIORI mặc

định tạo ra mười luật, độ tin cậy tối thiểu đối với mỗi luật là 0.9, số lượng thể hiện trong dữ liệu phù hợp với luật được tạo ra là 10%. Trong quá trình thực hiện thuật toán sẽ kết thúc khi một trong hai điều kiện: số lượng luật được yêu cầu đã tạo đủ hoặc số lượng thể hiện trong dữ liệu phù hợp với luật vừa tạo ra nhỏ hơn 10%.

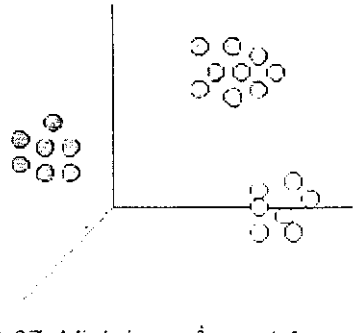
Tham số	Chức năng
-t	Xác định dữ liệu đầu vào dạng *.arff cho quá trình huấn luyện của thuật toán học máy.
-T	Xác định dữ liệu đầu vào dạng *.arff cho quá trình kiểm thử sự thực hiện của thuật toán học máy.
-x	Xác định số fold dùng trong quá trình kiểm thử chéo của thuật toán.
-i	Xác định dữ liệu đầu vào dạng *.model cho quá trình thực hiện của thuật toán.
-d	Xác định dữ kết quả thực hiện của thuật toán được lưu ở dưới dạng *.model.
-p	Đưa ra các dự đoán về các thể hiện trong file kiểm thử.

Bảng 8-5: Các thuật toán về association của Weka.

- Chức năng phân cụm (cluster) của WEKA

+ Định nghĩa phân cụm (cluster)

Quá trình phân tích phân cụm để xác định các cụm có trong dữ liệu theo một tập các điều kiện nào đó, trong đó một cụm là một tập hợp cá đối tượng dữ liệu tương tự nhau. Sự tương tự được xác định bằng các khoảng cách đặc biệt ví dụ như khoảng cách Oclit. Một phương pháp phân cụm được gọi là tốt khi sự tương tự nhau giữa các đối thuộc các cụm khác nhau là nhỏ trong khi sự tương tự giữa các đối tượng trong cùng một cụm là rất cao.



Hình 8-37: Minh họa về sự phân cụm

Các phương pháp phân cụm được chia làm hai loại chính đó là:

Phân cụm theo kiểu cây thừa kế: với phương pháp này sau khi một tập dữ liệu được chia thành các cụm theo một số điều kiện nào đó, chúng lại được tiếp tục phân chia thành các cụm nhỏ hơn với các điều kiện tùy thuộc từng cụm nhỏ đó.

Tìm ra một cụm dữ liệu từ các cụm dữ liệu đã được phân chia thành k cụm theo một số điều kiện nào đó.

Ví dụ phải phân cụm các loại nhà trong một vùng theo các chỉ tiêu như loại nhà, diện tích sàn nhà, và vị trí địa lý.

+ Các thuật toán về phân cụm (cluster)của WEKA

Weka cung cấp các thuật toán thuộc dạng phân cụm trong gói *weka.clusterers*. Các thao tác thực hiện đối với các thuật toán dạng phân cụm cũng tương tự như đối với các thao tác thực hiện các thuật toán về phân lớp trong gói *weka.classifiers*. Các tham số lựa chọn trong quá trình thực hiện các thuật toán được chia thành các tham số thông thường được trình bày ở bảng trên có thể được dùng cho tất cả các thuật toán và các tham số sử dụng trong các thuật toán xác định.

Tham số	Chức năng
-t	Xác định dữ liệu đầu vào dạng *.arff cho quá trình huấn luyện của thuật toán học máy.
-T	Xác định dữ liệu đầu vào dạng *.arff cho quá trình kiểm thử sự thực hiện của thuật toán học máy.
-x	Xác định số fold dùng trong quá trình kiểm thử chéo của thuật toán.
-i	Xác định dữ liệu đầu vào dạng *.model cho quá trình thực hiện của thuật toán.
-d	Xác định dữ kết quả thực hiện của thuật toán được lưu ở dưới dạng *.model.
-p	Đưa ra các dự đoán về các thể hiện trong file kiểm thử.

Bảng 8-6: Các tham số sử dụng trong các thuật toán phân cụm

Sau khi một thuật toán phân cụm thực hiện với các thể hiện của một tập dữ liệu nào đó, kết quả thực hiện của thuật toán được lưu ở dưới dạng file *.model. Sự thực hiện chính xác của thuật toán được đánh giá thông qua mức độ phù hợp của kết quả đối với dữ liệu kiểm thử, mức độ phù hợp càng lớn thì sự chính xác trong thực hiện của thuật toán là càng cao.

Người thực hiện có thể lựa chọn một trong hai kiểu đánh giá sự thực hiện của thuật toán: thứ nhất người dùng có thể sử dụng một file dữ liệu đưa vào thông qua sử dụng tham số -T để kiểm thử. Thứ hai, người dùng có thể sử dụng chính file dữ liệu sử dụng trong huấn

luyện thuật toán để thực hiện đánh giá chéo thông qua tham số -x, số fold sử dụng trong đánh giá chéo tùy thuộc sự lựa chọn của người dùng. Kết quả thực hiện của các thuật toán có cả các thông tin về số các thể hiện của dữ liệu trong từng cụm. Với các thuật toán không xây dựng mô hình phân phối xác suất thì nó đưa ra các thông tin thống kê. Weka cung cấp hai thuật toán thực hiện phân cụm và xây dựng mô hình dựa theo các phân phối xác suất của giá trị các thuộc tính đó là hai thuật toán trong gói *weka.clusterers:* weka.clusterers.EM và weka.clusterers.Cobweb. Hai thuật toán trên có thể thực hiện đối với cả các thuộc tính thuộc dạng số và các thuộc tính thuộc dạng định danh. Giống như giải thuật Naive Bayes, EM giả định rằng các thuộc tính là các biến ngẫu nhiên tự do. Câu lệnh thực hiện từ dòng lệnh đối với thuật toán EM như sau:

```
java weka.clusterers.EM -t weather.arff -N 2.
```

Kết quả thực hiện của thuật toán như sau:

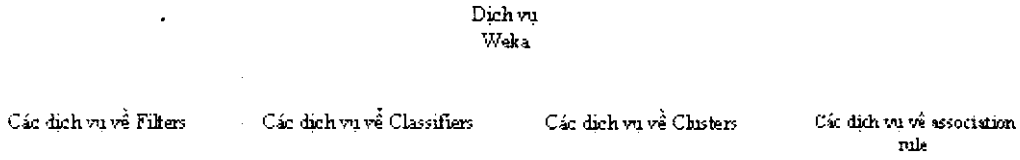
```
EM
==

Number of clusters: 2

Cluster: 0 Prior probability: 0.2816

Attribute: outlook
Discrete Estimator. Counts =  2.96 2.98 1  (Total = 6.94)
Attribute: temperature
Normal Distribution. Mean =  82.2692 StdDev =  2.2416
Attribute: humidity
Normal Distribution. Mean =  83.9788 StdDev =  6.3642
Attribute: windy
Discrete Estimator. Counts =  1.96 3.98  (Total = 5.94)
Attribute: play
Discrete Estimator. Counts =  2.98 2.96  (Total = 5.94)

Cluster: 1 Prior probability: 0.7184

Attribute: outlook
Discrete Estimator. Counts =  4.04 3.02 6  (Total = 13.06)
Attribute: temperature
Normal Distribution. Mean =  70.1616 StdDev =  3.8093
Attribute: humidity
Normal Distribution. Mean =  90.7271 StdDev =  11.6349
Attribute: windy
Discrete Estimator. Counts =  6.04 6.02  (Total = 12.06)
Attribute: play
Discrete Estimator. Counts =  8.02 4.04  (Total = 12.06)


=== Clustering stats for training data ===

Cluster Instances
      0         4  (29 %)
      1        10  (71 %)

Log likelihood: -9.01881
```

Hình 8-38: Kết quả thực hiện thuật toán EM

8.3.2. Lưới hóa ứng dụng WEKA

8.3.2.1. Nguyên tắc chung

Để lưới hoá ứng dụng Weka, ta thực hiện chuyển tất cả các mô-đun chức năng của ứng dụng này thành các dịch vụ; các dịch vụ này sau đó được triển khai tại các nút tính toán trong môi trường lưới. Mô hình tổng quan các dịch vụ của ứng dụng lưới Weka (grid weka) được mô tả trong hình sau.



Hình 8-39: Tổng quát hóa các thành phần của Weka

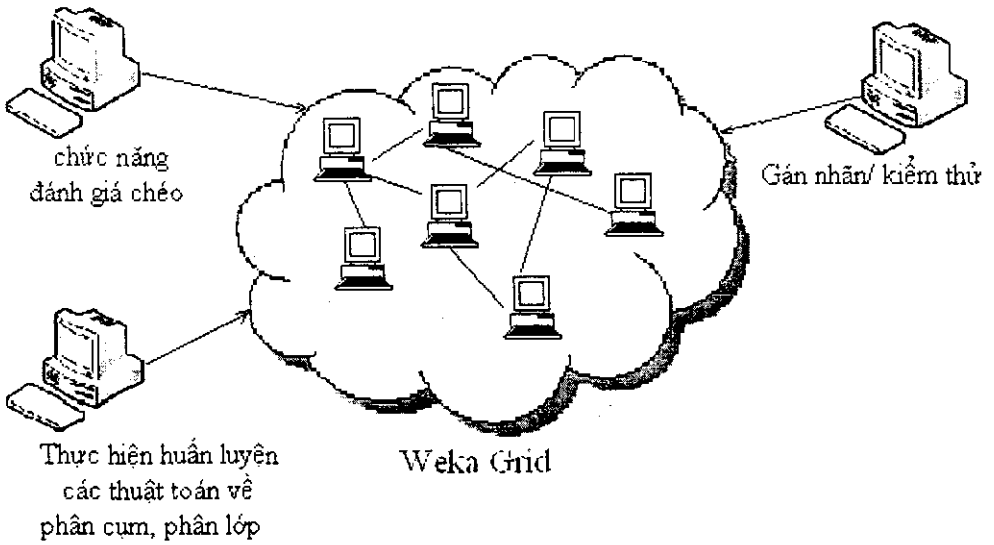
Dịch vụ Grid Weka bao gồm bốn thành phần của khai phá dữ liệu như ở trên. Trong đó các dịch vụ về tiền xử lý dữ liệu (filter) bao gồm các thuật toán về tiền xử lý dữ liệu, dịch vụ về phân lớp (classifier) bao gồm các thuật toán về phân lớp, dịch vụ về phân cụm (cluster) bao gồm các thuật toán về phân cụm và dịch vụ về kết tập luật (association rule) bao gồm các thuật toán về kết tập luật.

Grid Weka đã được tích hợp thành công trong hệ thống tính toán lưới BKGrid 2005 dưới dạng một dịch vụ về khai phá dữ liệu.

8.3.2.2. Mô tả chi tiết về các bước cần thiết trong quá trình triển khai một ứng dụng thành một dịch vụ lưới

1. Định nghĩa giao diện của dịch vụ. Sự định nghĩa giao diện của dịch vụ thông qua các mô tả trong file *GWSDL*
2. Xây dựng file thực hiện dịch vụ. File đó được thực hiện bởi ngôn ngữ Java.
3. Định nghĩa các tham số cần thiết trong quá trình triển khai. Các định nghĩa này được thực hiện trên file *WSDD*
4. Hoàn thành các bước trên và thực hiện đóng gói dịch vụ dưới dạng file GAR .
5. Triển khai dịch vụ.

Cụ thể ở đây chúng ta triển khai thuật toán J4.8 trong gói classifier của Weka thành dịch vụ lưới. Chi tiết công việc được mô tả trong các phần tiếp sau đây.



Hình 8-40: Quá trình sử dụng các dịch vụ

Sau khi Weka được lưới hóa trên hệ thống BKGrid2005 nó sẽ hoạt động như một dịch vụ lưới thực thụ. Người dùng có thẩm quyền sau khi đăng nhập qua portal của hệ thống sẽ có quyền sử dụng các dịch vụ khác nhau mà Weka Grid cung cấp. Người dung sử dụng dịch vụ Weka classifier Grid hoặc Weka cluster Grid để thực hiện quá trình huấn luyện sinh ra file model, quá trình này có thể thực hiện đồng thời với nhiều tập dữ liệu khác nhau trên các nút tính toán khác nhau. Đối với quá trình đánh giá chéo mỗi một bước lặp sẽ được thực hiện trên các tài nguyên tính toán khác nhau, quá trình này được thực hiện cho đến khi các bước lặp kết thúc và kết quả cuối cùng được trả về cho người dùng. Tương tự như vậy quá trình gán nhãn được thực hiện trên lưới bằng cách phân chia dữ liệu cần gán nhãn thành các phần nhỏ và mỗi phần nhỏ sẽ được thực hiện gán nhãn trên các nút tài nguyên khác nhau trong lưới.

8.3.2.3. Xây dựng mô đun định nghĩa giao diện dịch vụ (gwsdl)

Thành phần đầu tiên cần phải xem xét trong quá trình xây dựng một dịch vụ lưới đó chính là thành phần giao diện của dịch vụ. Chúng ta cần phải xác định các chức năng của dịch vụ được cung cấp. Ở đây, ta chưa quan tâm đến chi tiết cài đặt bên trong của các dịch vụ (các chi tiết liên quan đến thuật toán mà dịch vụ sử dụng, cở sở dữ liệu mà dịch vụ truy cập,...); thay vì điều đó, chúng ta chỉ quan tâm đến các thao tác dịch vụ sẽ cung cấp. Trong các dịch vụ lưới hay các dịch vụ Web giao diện của chúng được gọi là *port type*.

Giao diện của dịch vụ lưới được định nghĩa trên file GWSDL thông qua ngôn ngữ XML. Để làm ví dụ, ta sẽ xem xét một ví dụ cụ thể với chức năng phân lớp theo thuật toán J48 được cài đặt trong Weka.

Chức năng mà dịch vụ sẽ cung cấp cho người dùng ở đây là phương thức "tree", phương thức này sẽ gọi đến các chức năng cần thiết trong quá trình thực hiện thuật toán. Ta mô tả giao diện này bằng ngôn ngữ Java như sau:

```
public interface J48
{
  public void tree(String a);

}
```

Tham số hình thức a trong phương thức *tree* sẽ đại diện cho các giá trị tham số trong thuật toán J48 như trình bày trong bảng 2-1. Trong ngôn ngữ XML, giao diện này được định nghĩa như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions                                          name="TreeService"
targetNamespace="http://www.Tree/TreeService"
xmlns:tns="http://www.Tree/TreeService"
xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"

xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExt
ensions"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<import location="../../ogsi/ogsi.gwsdl"
  namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>

<types>
<xsd:schema targetNamespace="http://www.Tree/TreeService"
attributeFormDefault="qualified" elementFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema">        <!--BEGIN    ELEMENT
DEFINITIONS - DO NOT MODIFY THIS BLOCK!!! -->
      <xsd:element name="tree">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="arg1" type="xsd:string"/>
```

```
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="treeResponse">
            <xsd:complexType/>
        </xsd:element>
        <!--END ELEMENT DEFINITIONS -->
</xsd:schema>
</types>

<!--BEGIN MESSAGE DEFINITIONS - DO NOT MODIFY THIS BLOCK!!! -->

<message name="treeInputMessage">
    <part name="parameters" element="tns:tree"/>
</message>

<message name="treeOutputMessage">
    <part name="parameters" element="tns:treeResponse"/>
</message>

<!--END MESSAGE DEFINITIONS -->

<gwsdl:portType name="TreePortType" extends="ogsi:GridService">
    <!--BEGIN OPERATION DEFINITIONS - DO NOT MODIFY THIS BLOCK!!! -
->
    <operation name="tree">
        <input message="tns:treeInputMessage"/>
        <output message="tns:treeOutputMessage"/>
        <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
    <!--END OPERATION DEFINITIONS -->
</gwsdl:portType>
```

```
</definitions>
```

Phần đầu tiên trong file GWSDL đó là mô tả về tên vùng của dịch vụ lưới khi chúng được triển khai:

```
http://www.Tree/TreeService
```

Ngoài ra còn đưa ra các định nghĩa về các tên vùng của OGSI:

```
xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExt
ensions"
```

Tiếp theo là định nghĩa import file ogsi.gwsdl, là file gwsdl chuẩn định nghĩa cho tất cả các kiểu OGSI, các thông báo, và các giao diện (portType).

```
<import location="../../ogsi/ogsi.gwsdl"
namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
```

Ngôn ngữ sử dụng trong GWSDL rất gần với ngôn ngữ tự nhiên. Ngoài ra, ngôn ngữ sử dụng trong file GWSDL không phụ thuộc vào ngôn ngữ sử dụng trong file thực thi dịch vụ hoặc ngôn ngữ sử dụng trong file thực thi yêu cầu người dùng. Trong quá trình sinh các lớp stub các tên vùng xác định trong file GWSDL sẽ được ánh xạ tới các gói các lớp Java trong chương trình. Với ví dụ trên ta có:

```
http://www.Tree/TreeService =          I48.stubs.J48Service

http://www.Tree/TreeService/bindings=
       J48.stubs.J48Service.bindings

http://www.Tree/TreeService /service=
        J48.stubs.J48Service.service
```

Địa chỉ đầu tiên là địa chỉ tên miền của file GWSDL . Hai địa chỉ tên miền tiếp theo được sinh ra bởi công cụ Globus Tool Kit trong quá trình biên dịch. Các lớp stub được sinh ra được tập hợp trong gói:

J48.stubs

Dịch vụ chúng ta định nghĩa là J48, địa chỉ tên miền của file GWSDL định nghĩa cho dịch vụ được ánh xạ vào trong gói:

J48.stubs.*J48*

8.3.2.4. Xây dựng mô đun triển khai dịch vụ

Sau khi đã định nghĩa giao diện của dịch vụ, bước tiếp theo đó là thực thi công việc mà giao diện đã định nghĩa. Sự thực thi của dịch vụ lưới được thể hiện bởi ngôn ngữ Java thông qua một file, file này mô tả các thao tác định nghĩa trong file GWSDL, và các yêu cầu cần thiết.

File thực thi dịch vụ được định nghĩa với tên là J48Impl đặt tại thư mục

/usr/local/eclipse/wordspace/J48/s c/

Các lớp cần thiết phải import vào trong định nghĩa của file thực thi dịch vụ:

Đầu tiên là tên gói chứa dịch vụ, và các lớp cơ bản nhất phải cần thiết trong quá trình thực thi dịch vụ.

```
package J48.impl;


import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import J48.stubs.J48Service.J48PortType;
import java.rmi.RemoteException;
```

Tiếp theo là định nghĩa lớp J48Impl, lớp thực thi dịch vụ lưới.

```
public class J48Impl extends GridServiceImpl implements J48PortType
```

Chú ý :

Lớp J48Impl là lớp con của lớp GridServiceImpl. Tất cả các dịch vụ lưới phải extend lớp GridServiceImpl. Lớp GridServiceImpl được gọi là lớpskeleton. GridServiceImpl thực thi tất cả các thao tác cơ bản cần thiết cho quá trình thực thi các dịch vụ lưới.

Dịch vụ lưới mà chúng ta định nghĩa thực thi các định nghĩa trong giao diện J48PortType.

Tiếp theo đó là hàm tạo cho dịch vụ lưới. Hàm tạo này thường gọi đến hàm tạo của GridServiceImpl. Mô tả chi tiết như sau:

```
Public J48Impl()
{
        super("Dịch vụ J48 của gói Weka");
}
```

Cuối cùng, các phương thức định nghĩa trong giao diện được định nghĩa chi tiết.

```
public void tree(String a) throws RemoteException
{
// định nghĩa chi tiết
  }
```

Toàn bộ file định nghĩa sự thực thi của dịch vụ như sau:

```
package J48.impl;

import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import J48.stubs.J48Service.J48PortType;
import java.rmi.RemoteException;
import org.globus.ogsa.GridServiceBase;
import org.globus.ogsa.GridServiceException;
import org.globus.ogsa.OperationProvider;
import org.globus.ogsa.GridContext;
import java.rmi.RemoteException;
import javax.xml.namespace.QName;
import weka.classifiers.trees.j48.*;
import java.util.*;
import weka.core.*;
import weka.classifiers.*;
import J48.impl.J48;
import java.lang.*;

public class J48Impl extends GridServiceImpl implements J48PortType
{
  private int value = 0;

  public J48Impl()
  {
    super("Simple J48Service");
  }

  public void tree(String a) throws RemoteException
  {
public void tree(String a) throws RemoteException
    {
```

```
        try {

                String arg[]=new String[20];



                int pos=a.indexOf(' ');
                int pos1=0;
                while (pos >=0)
                {
    // quá trình xử lý tham số đầu vào dưới dạng một chuỗi
  // các tham số khi được đưa vào thuật toán được cách nhau bởi
  //các dấu trắng  cần chuẩn hóa
                    String s1=a.substring(0,pos);
                    arg[pos1++]=s1.trim();
                    a=a.substring(pos+1);
                    pos=a.indexOf(' ');

                }
                arg[pos1++]=a;
                int i1=0;

                for (i1=pos1; i1 < arg.length;i1++)
                    arg[i1]="";
// lưu kết quả thực hiện của thuật toán dưới dạng file text
                FileOutputStream        outFile        =        new
FileOutputStream("/tmp/J48tmp.txt");

                DataOutputStream        outData        =        new
DataOutputStream(outFile);
// khởi tạo một đối tượng J48 và thực hiện thuật toán
    outData.writeUTF(Evaluation.evaluateModel(new J48(), arg));
                outData.close();
```

```
                }
            catch (Exception e)
                {
                    System.err.println(e.getMessage());
                    }
        }
}
```

Bước tiếp theo đó là xây dựng mô đun cấu hình sự triển khai dịch vụ

8.3.2.5. Xây dựng mô đun cấu hình triển khai dịch vụ (wsdd)

Một trong những thành phần cấu thành trong quá trình triển khai dịch vụ đó là mô đun mô tả triển khai dịch vụ web (web service deployment descriptor). Các mô tả được thể hiện trong file wsdd chính là cấu hình cho việc xuất bản dịch vụ lưới. Mô tả triển khai cho dịch vụ lưới J48 được mô tả như sau:

```
<?xml version="1.0"?>
<deployment                                    name="defaultServerConfig"
xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="J48/J48Service" provider="Handler" style="wrapped">
    <parameter name="name" value="J48Service"/>
    <parameter                                          name="className"
value="J48.stubs.J48Service.J48PortType"/>

    <parameter name="baseClassName" value="J48.impl.J48Impl"/>
    <parameter                name="schemaPath"                value="
J48\schema\gt3ide\J48Service /J48service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*"/>
    <parameter name="persistent" value="true"/>
    <parameter                                        name="handlerClass"
value="org.globus.ogsa.handlers.RPCURIProvider"/>
  </service>
```

```
</deployment>
```

Trong file cấu hình triển khai dịch vụ các thành phần sau quyết định tính riêng biệt cho mỗi dịch vụ:

Tên dịch vụ 'service name'

```
<service name="J48/J48Service" provider="Handler" style="wrapped">
```

Xác định địa chỉ của dịch vụ J48Service. Khi sử dụng middleware Globus Tool Kit địa chỉ thực của dịch vụ sẽ được xác định thông qua địa chỉ URL cơ sở của Grid Service Container :http://localhost:8080/ogsa/services. Khi đó địa chỉ của dịch vụ sẽ là:

```
http://localhost:8080/J48/J48Service
```

Tên dịch vụ 'service name' (tiếp)

```
<parameter name="name" value="J48Service"/>
```

Tham số className và baseClassName

```
<parameter name="className" value=" J48.stubs.J48Service.J48PortType
"/>
<parameter name="baseClassName" value="J48.impl.J48Impl "/>
```

8.3.2.6. Đóng gói dịch vụ dùng công cụ Ant

Phần tiếp theo trong quá trình triển khai dịch vụ lưới đó là đóng gói dịch vụ. Các mô đun cần thiết cho việc triển khai dịch vụ lưới sẽ được đóng thành một gói GAR thông qua công cụ Ant.

Các mô đun cần thiết cho quá trình đóng gói GAR bao gồm: file định nghĩa giao diện dịch vụ GWSDL, các file thực thi dịch vụ (J48Impl.java, và J48.java), và file mô tả triển khai dịch vụ lưới WSDD.

Quá trình đóng gói dịch vụ lưới thành dạng file GAR phải thực hiện các chức năng sau:

Ánh xạ file GWSDL sang dạng file WSDL

Tạo ra các lớp stub từ file WSDL vừa tạo ra

Biên dịch các lớp stub vừa tạo ra

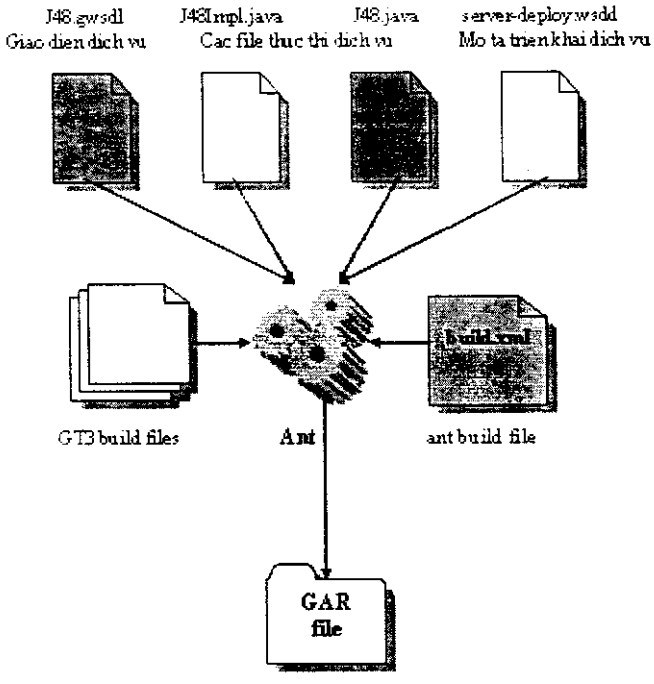
Biên dịch các file thực thi dịch vụ

Tổ chức các file cần thiết thành các thư mục riêng để phục vụ cho quá trình triển khai dịch vụ.

Tất cả các bước thực hiện ở phía trên được hỗ trợ bởi middleware Globus Tool Kit và công cụ Ant.

Ant, là viết tắt của Apache Software Foundation , là một bộ công cụ được xây dựng trên ngôn ngữ Java. Trong quá trình đóng gói Ant thực hiện biên dịch tất cả các file nguồn được

cung cấp bởi các mô đun. Mỗi một dịch vụ có một file định nghĩa riêng gọi là *build file*. File này trực tiếp điều khiển Ant trong quá trình biên dịch và đóng gói dịch vụ.



Hình 8-41: Mô hình đóng gói dịch vụ với công cụ Ant

Như trên hình vẽ đã thể hiện, Ant tạo ra file GAR trực tiếp từ các mô đun đã tạo ở phía trên. Trong các dịch vụ triển khai trên nền GT3, Ant sử dụng hai tập các build file: một tập các file build file được cung cấp bởi middleware Globus Tool Kit, và tập còn lại do người phát triển ứng dụng định nghĩa. Các build file cung cấp bởi GT3 thực hiện điều khiển tất cả các bước quan trọng trong quá trình đóng gói dịch vụ (tạo ra file WSDL từ file GWSDL, sinh ra các lớp stub, ...). build file do người phát triển ứng dụng chứa tất cả các tham số của dịch vụ lưới đang được xây dựng và các lời gọi đến các GT3 build file.

- Tạo ra J48Service GAR

Câu lệnh thực hiện quá trình đóng gói file GAR.như sau:

```
./ant buid  <đường dẫn chỉ đến thư mục chứa dịch vụ> <tên file GWSDL
của dịch vụ>
```

Với dịch vụ J48 mà chúng ta đang triển khai câu lệnh thực hiện cụ thể như sau:

```
$ GLOBUS_LOCATION/ant build /<đường dẫn tới thư mục chứa dịch
vụ>/J48.gwsdl
```

- Triển khai dịch vụ

GAR file vừa được tạo ra bao gồm tất cả các file và các thông tin cần thiết để triển khai dịch vụ lưới. Việc triển khai dịch vụ cũng được thực hiện bởi công cụ Ant, quá trình triển khai là

quá trình bung gói GAR và copy tất cả các file cần thiết vào thư mục xác định được tạo ra trong thư mục cài GT3.2.1 (phiên bản 3.2.1 của Globus Tool Kit).

```
ant deploy -Dgar.name=<Đường dẫn tới GAR file>
```

Bước tiếp theo là viết một file client để sử dụng dịch vụ vừa được tạo.

Chương 9: Dịch vụ tính toán

Bên cạnh các dịch vụ Weka, BK Grid 2005 còn cung cấp dịch vụ chạy ứng dụng từ xa. Người dùng chỉ việc viết sẵn ứng dụng theo chuẩn MPI sau đó upload lên portal của BK Grid, dịch vụ tính toán trên cluster sẽ chịu trách nhiệm điều khiển thực thi ứng dụng này và trả lại kết quả đã xử lý cho người dùng. Mô-đun đệ trình công việc lên lưới đưa ra nhằm đơn giản hóa quá trình thực hiện công việc cho người dùng. Mô-đun này cung cấp các chức năng sau:

- Cung cấp giao diện Web thân thiện cho phép người dùng người dùng đệ trình công việc lên lưới.
- Chạy được các ứng dụng đơn và các ứng dụng song song lập trình theo thư viện truyền thông điệp MPI.
- Quản lý tình trạng các công việc do người dùng đệ trình lên lưới.
- Lưu lại kết quả trên máy chủ và trả lại kết quả cho người dùng khi có yêu cầu.

Việc triệu gọi ứng dụng từ xa trở nên trong suốt đối với người dùng không chuyên về lưới nhờ việc giải phóng họ khỏi các công việc đặc thù của lưới như. Để sử dụng mô-đun này trước tiên người sử dụng phải thực hiện đăng nhập vào lưới, đưa các thông tin yêu cầu và ủy quyền cho lưới thực hiện công việc, mô-đun đệ trình công việc sẽ nhận các thông tin này, sử dụng dịch vụ lập lịch lưới thực hiện tìm kiếm thông tin về các tài nguyên lựa chọn tài nguyên thích hợp nhất để thực hiện công việc. Môđun đệ trình công việc thực hiện theo dõi tình trạng công việc và trả lại kết quả cho người dùng khi có yêu cầu.

9.1. Tổng quan về PBS

Công nghệ tính toán song song phân cụm và công nghệ tính toán lưới đang được xem như là chiếc chìa khóa để giải quyết các bài toán yêu cầu khối lượng tính toán lớn. Phần đầu của chương này trình bày những vấn đề cơ bản nhất về hệ thống tính toán song song phân cụm, hệ thống tính toán lưới, và nhu cầu cần thiết của việc kết hợp hai hệ thống tính toán này.

9.1.1. PBS và tính toán song song phân cụm

9.1.1.1. Tính toán song song phân cụm

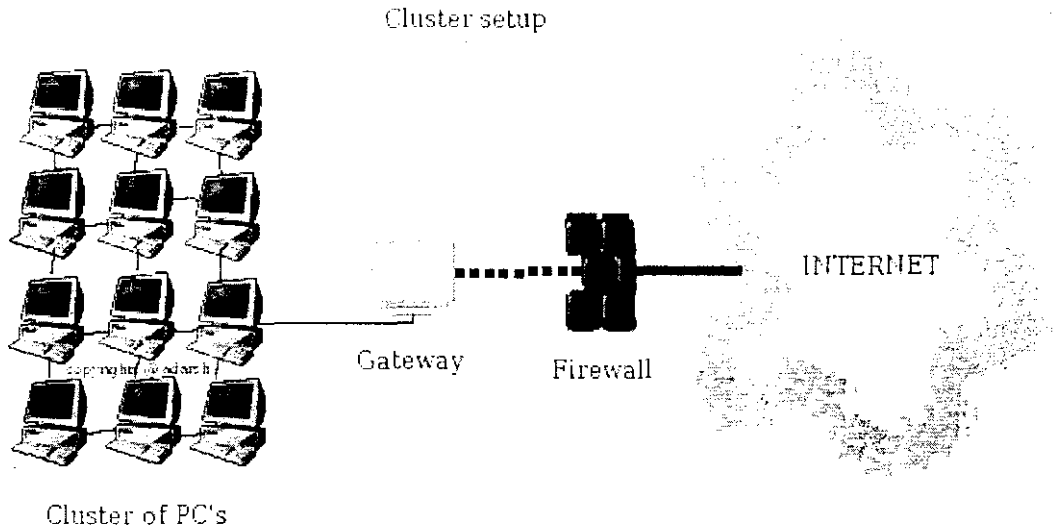
Ngày nay cùng với sự bùng nổ của khoa học kỹ thuật, thì các bài toán cỡ lớn giải quyết các vấn đề khoa học kỹ thuật đặt ra ngày càng nhiều. Các bài toán này thường yêu cầu một khối lượng tính toán lớn ví dụ như những bài toán dự báo động đất, bài toán dịch chuyển của các tầng địa chất hay bài toán nghiên cứu vật lý hạt nhân v.v... Một trong những giải pháp đưa ra để giải quyết vấn đề này là tính toán song song, vậy tính toán song song là gì? Tính toán

song song là việc sử dụng đồng thời nhiều tài nguyên tính toán để giải quyết một bài toán. Các tài nguyên tính toán có thể bao gồm một máy tính với nhiều bộ vi xử lý, một tập các máy tính được kết nối với nhau qua mạng hay là một sự kết hợp của hai dạng trên. Bài toán được giải quyết trên hệ thống tính toán song song thường có các đặc tính sau: có thể tách thành nhiều bài toán nhỏ có thể được giải quyết đồng thời, có thể thực hiện tại cùng một thời điểm nhiều lệnh của chương trình. Việc thực hiện bài toán trên nhiều tài nguyên tính toán sẽ nhanh hơn so với thực hiện trên một tài nguyên tính toán.

Để thực hiện các bài toán song song người ta sử dụng một hệ thống gọi là hệ thống tính toán song song phân cụm. Đó là một hệ thống máy tính cục bộ bao gồm một tập các máy tính độc lập và một mạng liên kết giữa các máy tính. Một hệ thống tính toán song song phân cụm thường mang tính chất cục bộ nghĩa là toàn bộ các nút tính toán của hệ thống nằm trong một khu vực địa lý hẹp (thường là 1 mạng LAN) và được quản lý tập trung như là một hệ thống thống nhất. Các nút trong hệ thống tính toán song song phân cụm có thể là sự kết hợp của các máy tính đơn bộ xử lý với các máy đa bộ xử lý đối xứng (SMP - symmertric multiprocesor).

Ưu điểm lớn nhất của hệ thống tính toán song song phân cụm là giá thành của hệ thống. Cùng với sự phát triển nhanh chóng của máy tính cá nhân (PC), hiện nay một hệ thống dựa trên công nghệ LAN và PC có thể đạt được hiệu năng đến vài chục thậm chí là vài trăm Gigaflops trong khi giá thành lại rẻ hơn rất nhiều so với các siêu máy tính.Ưu điểm thứ hai của hệ thống bó là sự linh hoạt của cấu hình. Số lượng nút, dung lượng bộ nhớ trên mỗi nút, số lượng bộ xử lý trên mỗi nút, và cấu hình mạng và hàng loạt các tham số khác đều mang tính tuỳ biến cao.

Nếu giá thành rẻ là lợi thế của hệ thống tính toán song song phân cụm thì việc triển khai phần mềm trên hệ thống này lại khá phức tạp. Nhà quản trị phải thực hiện liên kết các máy trong phạm vi một mạng LAN, cài đặt tường lửa (firewall) để tránh các truy cập từ bên ngoài, còn cả hệ thống được quản lý bởi nhà quản trị, như vậy vấn đề bảo mật hoàn toàn được kiểm soát, tất cả mọi thiết lập của người quản trị nhằm đạt được hiệu năng tính toán cao nhất.



Hình 9-1: Thiết lập cluster

Các hệ thống bó thường được triển khai trên các hệ điều hành dòng Unix như Linux, Unix, Solaris, AIX... Với sự ra đời của Linux và xu hướng lập trình mã nguồn mở, hiện nay có rất nhiều công cụ, thư viện mã nguồn mở và các thành phần phần mềm khác hỗ trợ việc triển khai hệ thống tính toán song song phân cụm với khả năng tuỳ biến cao. Trong đó thư viện lập trình song song và phần mềm quản lý phân tải là hai thành phần không thể thiếu.

- Thư viện lập trình

Hệ thống tính toán song song là hệ thống máy tính có bộ nhớ phân tán, do vậy truyền thông giữa các nút trong quá trình tính toán thường là truyền thông điệp. Các thư viện truyền thông điệp dành cho hệ thống bó hiện nay có thể kể đến là MPI, PVM, Active Message và CMMD (Connected Machine Message...), trong đó MPI với nhiều ưu điểm có thể coi là sự lựa chọn tốt cho các hệ thống tính toán bó. Hiện nay, có nhiều phiên bản cài đặt cho MPI với các chức năng chuẩn của MPI 1.2 và một số chức năng mở rộng riêng. Trong đó MPICH-1.2.5.2 và LAM-MPI-7.0.6 là hai thư viện truyền thông điệp được sử dụng phổ biến nhất. Hệ thống tính toán hiệu năng cao trên trung tâm HPCC cũng sử dụng hai thư viện truyền thông điệp này.

- Các phần mềm quản lý tài nguyên và phân tải

Trong khi các thư viện và các ngôn ngữ lập trình song song tạo cho người lập trình môi trường lập trình trên hệ thống tính toán song song, thì các phần mềm quản lý tài nguyên và phân tải lại tạo ra một môi trường tính toán hiệu quả và ổn định. Việc sử dụng các phần mềm này thực sự có ý nghĩa khi số lượng các nút tính toán càng lớn và số lượng các chương trình thực hiện trên hệ thống là nhiều. Phân tải là việc phân bổ các tiến trình tính toán trên tài nguyên hệ thống sao cho hiệu năng hoạt động của hệ thống là tối ưu. Do đó trong chính các phần mềm phân tải phải sử dụng các phần mềm quản lý tài nguyên và lập lịch. Phần mềm quản lý tài nguyên còn được sử dụng trong việc quản trị hệ thống, đánh giá

hiệu năng hệ thống v.v... Một số phần mềm phân tải và quản lý tài nguyên thường được sử dụng trong các hệ thống tính toán song song hiện nay là:

PBS (Portable Batch System) là hệ thống phân tải và quản lý tài nguyên rất mạnh do NASA phát triển. Hệ thống này được sử dụng rất phổ biến trong các hệ thống tính toán hiệu năng cao. Chức năng chính của hệ thống là tiếp nhận các công việc (job) từ người sử dụng, bảo vệ công việc đó cho đến khi nó được thực hiện, thực hiện công việc và trả lại kết quả cho người sử dụng. PBS có thể được cấu hình chạy trên một hệ thống đơn hoặc một hệ thống đa máy tính. Sự linh động này làm cho PBS có khả năng đáp ứng cho nhiều hệ thống tính toán.

CODINE là một phần mềm phân tải dùng cho các hệ thống máy tính máy tính đồng bộ như SMP hoặc các siêu máy tính dạng vector. Phần mềm này còn cung cấp các cơ chế để quản lý các chính sách của hệ thống bó và cân bằng tải động cho các công việc

CONDOR thường dùng để phân tải tài nguyên và lập lịch cho các công việc trên hệ thống máy trạm (workstations). Nó hỗ trợ sử dụng các điểm kiểm tra (checkpoint) và chuyển giao các công việc giữa các máy trạm.

9.1.1.2. Sự cần thiết của việc kết nối Globus-based grid và PBS-based cluster

Tính toán song song phân cụm vẫn giữ được vai trò của nó khi tính toán lưới ra đời và phát triển. Trong khi tính toán song song phân cụm trong khi tính toán song song phân cụm tập trung giải quyết bài toán trong phạm vi một tổ chức (phạm vi hẹp) thì tính toán lưới lại tìm cách phân bổ bài toán trên một tập tài nguyên rộng phân tán về mặt địa lý (phạm vi rộng). Vì vậy khi nói đến các cluster ta thấy ngữ cảnh của chúng cũng hẹp hơn tính toán lưới rất nhiều thể hiện ở các điểm sau:

Trước hết các cluster thuộc về một tổ chức, các tài nguyên của cluster là các máy tính thường là đồng nhất về mặt kiến trúc, hệ điều hành, khả năng xử lý...và thường phân bố trên một giới hạn địa lý rất hẹp (thường là trong một phòng máy) và được liên kết với nhau bởi các mạng LAN tốc độ cao (cỡ vài trăm Mbits – Gbits) do vậy có độ tin cậy rất cao. Trái lại lưới bao gồm tập hợp rất nhiều tài nguyên đa dạng không đồng nhất thuộc nhiều tổ chức phân tán trên diện rộng về mặt địa lý được liên kết với nhau bởi các mạng diện rộng tốc độ thấp hơn và mức độ tin cậy kém hơn.

Trong cluster các vấn đề về an ninh không được coi trọng. Vấn đề chủ yếu là làm sao để có được hiệu năng cao và không cần thiết phải hy sinh một phần hiệu năng vì các thủ tục an ninh (vì cluster chỉ thuộc về một tổ chức và chỉ nằm trong một phòng máy). Trong khi đó vấn đề an ninh là một trong những vấn đề quan trọng nhất mà tính toán lưới phải giải quyết.

Trong cluster người ta có thể làm mọi thứ để đổi lấy hiệu năng, ví dụ tối ưu hóa phần cứng phần mềm, sửa đổi nhân hệ điều hành hoặc viết hẳn những hệ điều hành chuyên biệt, thay đổi giao thức... Tính toán lưới không đặt vấn đề hiệu năng lên hàng đầu mà chủ yếu tập trung vào việc làm sao phân bổ hiệu quả các nguồn tài nguyên. Và tất nhiên khi đã đạt được

mục tiêu đó thì tính toán lưới cũng mang lại hiệu năng rất cao, thậm chí hơn nhiều lần các siêu máy tính hiện có và điều đó chủ yếu là do quy mô cực kỳ rộng lớn của nó.

Cuối cùng cũng cần phải nói thêm rằng tính toán lưới không phải là chìa khóa vạn năng dùng để giải quyết mọi vấn đề. Nó được dùng để bổ trợ chứ không phải là thay thế hoàn toàn các công nghệ tính toán hiện tại. Vì vậy việc kết nối lưới với cluster đóng vai trò cực kỳ quan trọng trong việc cung cấp một môi trường tính toán mạng lưới cung cấp cho người dùng.

Vì vậy sự kết hợp của hai mô hình tính toán này sẽ đem lại một sức mạnh tính toán vô cùng to lớn. Với sự kết hợp này ta sẽ tận dụng được sức mạnh sẵn có của các cluster và sử dụng tính toán lưới như là phương tiện để phối hợp các cluster tại các miền địa lý khác nhau cùng giải quyết một bài toán.

9.2. Kết nối Globus-based Grid và PBS-based Cluster

Trong mục này ta sẽ phân tích kỹ hai thành phần đóng vai trò kết nối giữa hai hệ thống Globus-based grid và PBS-based cluster là thành phần GRAM trong Globus và thành phần PBS_Server trong PBS; đây là hai thành phần đóng vai trò quan trọng kết nối hai hệ thống Globus-based grid và PBS-based cluster. Sau đó sẽ thực hiện việc kết nối thông qua một số bước cấu hình để cho hai hệ thống có thể hiểu nhau.

9.2.1. GRAM

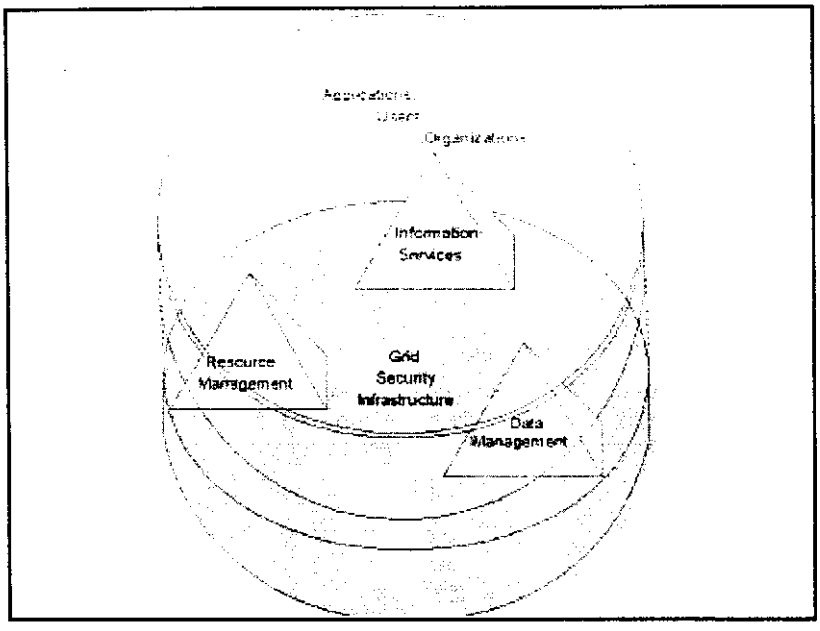
9.2.1.1. Giới thiệu GRAM

Như đã trình bày, Globus Toolkit có ba thành phần quan trọng nhất được ví như là "ba kim tự tháp" (hình 9-2) xây dựng trên cơ sở hạ tầng an ninh lưới đó là:

Thành phần quản lý tài nguyên (Resource management).

Thành phần quản lý dữ liệu (Data management).

Thành phần dịch vụ thông tin (Information services).

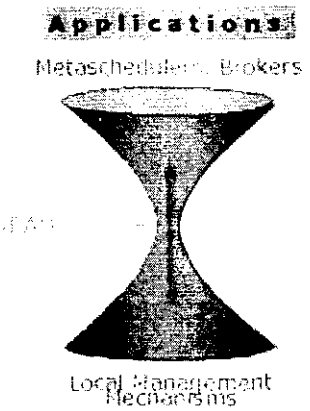


Hình 9-2: Ba "kim tự tháp" trong Globus Toolkit

Thành phần quản lí tài nguyên trong bộ công cụ Globus Toolkit GRAM (Globus Resource Allocation Management), có nhiệm vụ thực thi các yêu cầu thực hiện công việc trên tài nguyên bằng ngôn ngữ đặc tả tài nguyên RSL, trong trường hợp không có các tài nguyên yêu cầu thì GRAM thực hiện báo lỗi và hủy bỏ công việc (FAILED), nếu tài nguyên đó đang bận thì GRAM thực hiện đưa công việc việc vào trạng thái chờ (PENDING), hoặc tạo ra các tiến trình thỏa mãn yêu cầu đặt ra (ACTIVE).

Mọi hoạt động thực hiện với GRAM đều phải thông qua cơ chế bảo mật lưới GSI. Cơ sở hạ tầng bảo mật lưới GSI - Grid Security Infrastructure cung cấp một cơ chế chứng thực chung cho cả người sử dụng và tài nguyên từ xa, qua đó GRAM có được một cơ chế xác nhận đơn giản dựa trên định danh người dùng (GSI identify) và một cơ chế để ánh xạ định danh người dùng với tài khoản người dùng (grid-mapfile) trước khi thực hiện công việc.

Ta có thể coi GRAM như là phần bó hẹp của một cái đồng hồ cát (hình 2-2), với các ứng dụng và dịch vụ ở các mức cao hơn (chẳng hạn như là bộ môi giới tài nguyên - resource brokers hoặc bộ lập lịch lưới - metaschedulers) ở phía trên và các cơ chế truy nhập, điều khiển địa phương ở phía dưới. Cả hai phía chỉ cần làm việc với GRAM, bởi vậy sự ảnh hưởng lẫn nhau, các hàm API, và các giao thức cần sử dụng được giảm đi nhiều.



Hình 9-3: Vai trò của GRAM

GRAM không cung cấp khả năng lập lịch cũng như là môi giới tài nguyên. GRAM cũng không cung cấp các đặc trưng về kế toán và lập hóa đơn. Điều này là vì các đặc trưng sẽ được cung cấp bởi các cơ chế quản lý địa phương chẳng hạn như là hệ thống hàng đợi hoặc bộ lập lịch. GRAM làm việc như một giao tiếp trừu tượng cho các nguồn tài nguyên không đồng nhất trên lưới, cụ thể nó chịu trách nhiệm:

Phân bố tài nguyên cho các công việc.

Đệ trình các công việc, chạy ứng dụng trên các máy từ xa và nhận kết quả trả về.

Quản lý trạng thái công việc và các tiến trình.

Như vậy có thể thấy được GRAM đóng vai trò như thành phần trung gian làm nhiệm vụ giao tiếp giữa môi trường tính toán toàn cục với các thành phần cục bộ có khả năng tạo ra các tiến trình chạy. Vì vậy một khả năng quan trọng của GRAM phải đáp ứng đó là việc giao tiếp với các bộ quản lí tài nguyên địa phương chẳng hạn như: Condor, Fork, PBS, LSF, NQE, v.v... trong đó phần mềm quản lý tài nguyên và phân tải PBS được sử dụng rộng rãi nhất, thông tin chi tiết về phần mềm được cung cấp tại trang chủ PBSPro www.pbspro.org.

Để thực hiện yêu người dùng GRAM sử dụng kĩ thuật ước lượng thời gian bằng hàng đợi. Khi một công việc nào đó được chấp nhận thì một thẻ (handle) duy nhất được trả về và hệ thống sẽ dùng thẻ này để thực hiện các công việc theo dõi và điều khiển tiến trình tương ứng với công việc đó. Hơn thế nữa lời gọi chấp nhận công việc còn có thể thông báo tới các tiến trình được tạo ra để hỗ trợ lời gọi ngược (*callback URL*). Các thẻ tiến trình có thể được sử dụng để truyền như tham số tới các tiến trình khác; khả năng này là một đặc điểm quan trọng của GRAM trong chiến lược xây dựng các bộ môi giới tài nguyên ở mức cao (ví dụ trong một số trường hợp bộ môi giới hay bộ định vị được yêu cầu từ một ứng dụng, trong khi ứng dụng đó theo dõi và điều khiển tiến trình được tạo ra từ yêu cầu đó).

- Một số thuật ngữ

Trước khi tìm hiểu rõ kiến trúc hoạt động của GRAM, xin được trình bày qua các thuật ngữ là tên của các thành phần trong GRAM.

+ Virtual Host Environment Redirector

Là thành phần tiếp nhận các thông điệp SOAP (SOAP message) và gửi chúng đến cho môi trường người dùng UHE (User Host Environment).

+ Master Managed Job Factory Service - MMJFS

Master Managed Job Factory Service chịu trách nhiệm cung cấp dịch vụ chạy ứng dụng GRAM cho các yêu cầu từ bên ngoài.

+ Managed Job Factory Service - MJFS

Managed Job Factory Service chịu trách nhiệm tạo ra một dịch vụ quản lý công việc MJS (Managed Job Service) mới ngay khi nó nhận được một yêu cầu tạo dịch vụ. MJFS tồn tại từ khi khởi tạo môi trường người dùng (UHE) cho đến khi phiên làm việc kết thúc.

+ Managed Job Service - MJS

Là một dịch vụ OGSI cho phép các đặc tả công việc có thể đệ trình công việc tới hệ thống lập lịch địa phương, theo dõi tình trạng và đưa ra các thông báo. MJS sẽ khởi tạo hai dịch vụ thao tác file (File Streaming Factory Service - FSFS), một cho biết tình trạng đầu ra của công việc, một cho biết các lỗi xảy ra. MJS khởi tạo một tập các FSS được chỉ rõ trong yêu cầu của công việc.

+ File Stream Factory Service – FSFS

File Stream Factory Service được coi như là một giao diện nhận yêu cầu, chịu trách nhiệm cài đặt một dịch vụ về file (File Stream Service) mới khi nó nhận được yêu cầu tạo dịch vụ.

+ File Stream Service – FSS

Một dịch vụ OGSI cho một địa chỉ URL đích sẽ đưa file địa phương vừa được tạo đến địa chỉ URL đích.

+ Resource Information Provider Service – RIPS

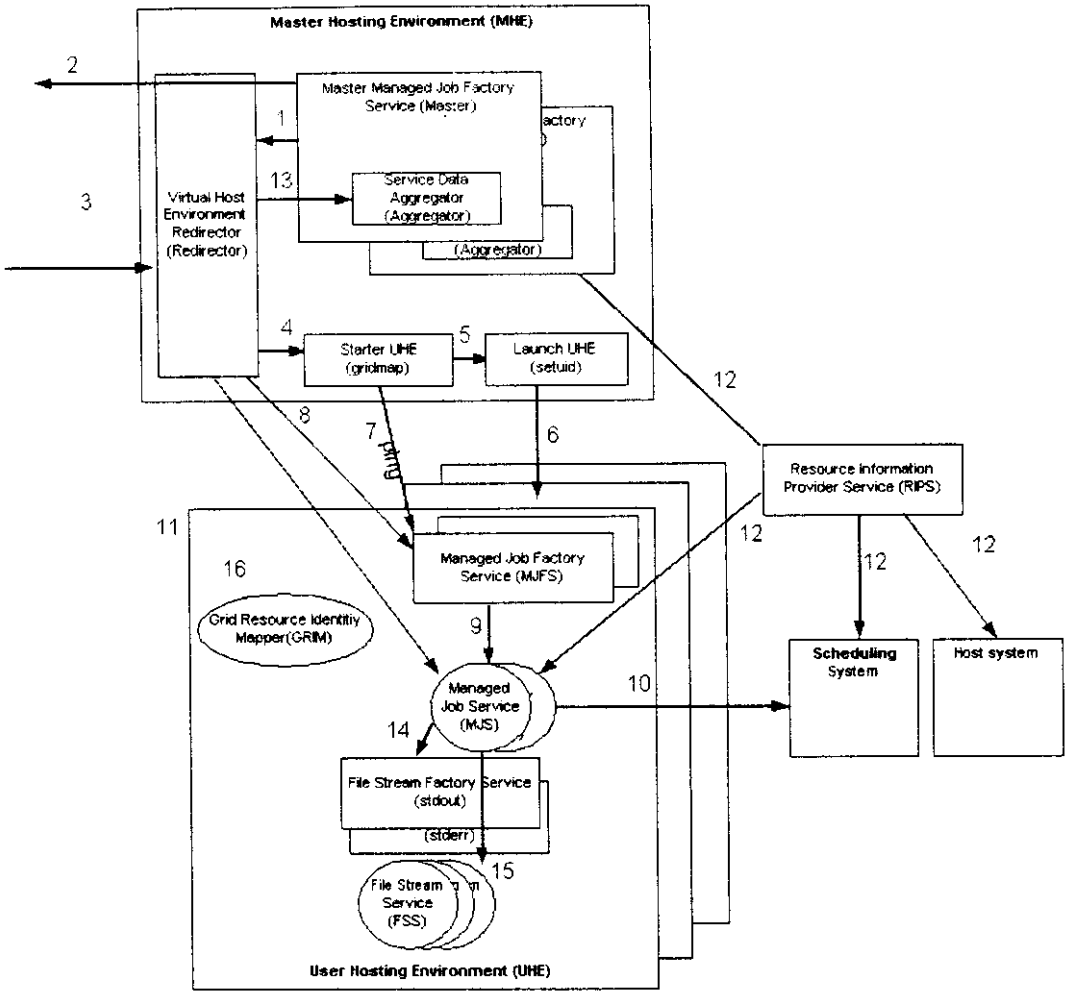
RIPS là một dịch vụ khai báo cung cấp thông tin về hệ thống lập lịch tài nguyên, hệ thống file, hệ thống máy chủ, v...v...

+ Grid Resource Identity Mapper – GRIM

GRIM là thành phần chịu trách nhiệm tạo ra giấy chứng nhận cho người dùng sau khi đã kiểm tra grid-mapfile.

9.2.1.2. Kiến trúc GRAM

Hình sau trình bày hoạt động của dịch vụ GRAM nhìn từ phía tài nguyên và người sử dụng:



Hình 9-4: Kiến trúc GRAM

Master được cấu hình để sử dụng Redirector định hướng lại các yêu cầu đến nó. Trong trường hợp UHE của người dùng chưa chạy thì Master sử dụng mô-đun Starter UHE khởi tạo lại UHE cho người dùng. Một Service sẽ được tạo ra để phục vụ cho yêu cầu của người dùng.

Master đưa ra những điều khiển đến một bộ đăng ký từ xa (remote registry) (*thao tác này có thể thực hiện hoặc không*).

Client thực hiện một yêu cầu đến đến Service đã được tạo ra, yêu cầu này được nhận bởi Redirector.

Redirector gọi đến lớp Starter UHE để chứng thực yêu cầu thông qua grid-mapfile, xác định tên người sử dụng, xây dựng một URL đích.

Redirector cố gắng chuyển yêu cầu đến URL đích. Nếu không chuyển yêu cầu bởi vì UHE không được khởi tạo thì một mô-đun Launch UHE sẽ được gọi.

Launch UHE tạo một UHE mới dưới sự xác nhận người sử dụng.

Starter UHE chờ đợi UHE được khởi tạo (thực hiện vòng lặp) và trả lại URL đích cho Redirector.

Redirector chuyển yêu cầu từ Service được tạo ra đến MJFS.

MJFS tạo ra một MJS (Managed Job Service) mới.

MJS đệ trình công việc vào hệ thống lập lịch.

Sau đó, yêu cầu đến MJS từ client sẽ được gửi lại một lần nữa thông qua Redirector.

RIPS cung cấp dữ liệu cho MJS và Master. Bản thân RIPS thì thu thập dữ liệu từ hệ thống lập lịch địa phương, hệ thống file, thông tin máy chủ, ...

Dịch vụ tìm kiếm dữ liệu yêu cầu Master cho kết quả trong một SDE khác hoặc chuyển đến MJFS trong môi trường của người sử dụng UHE.

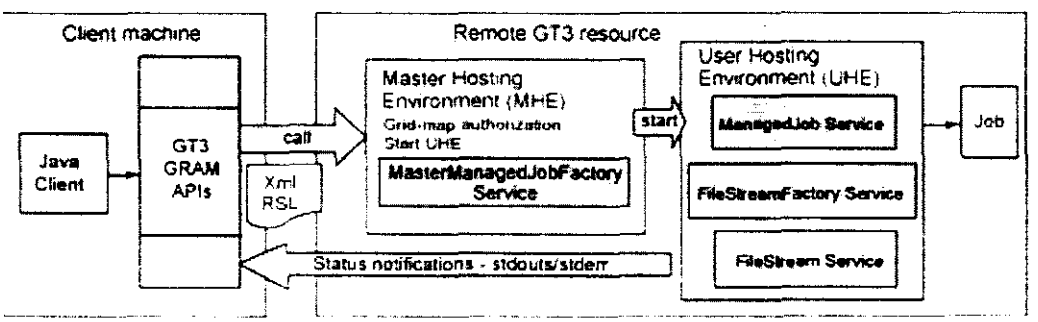
Trong trường hợp muốn đưa thông tin stdout/stderr trở lại cho client, MJS sẽ tạo ra 2 File Stream Factory Services (FSFS), một cho đầu ra chuẩn stdout và một cho các lỗi nếu có stderr.

Sau đó MJS sẽ tạo ra một thực thể File Stream Services (FSS) như là đã được chỉ định trong công việc đã yêu cầu.

Trình điều khiển GRIM chạy trong UHE sẽ tạo ra một giấy chứng nhận cho người sử dụng. Giấy chứng nhận này sẽ được sử dụng để xác nhận qua lại giữa dịch vụ MJS và client.

9.2.1.3. Đệ trình công việc với GT3 GRAM

GT3 GRAM gồm hai trình chứa dịch vụ: Môi trường trình chủ (MHE) và môi trường người dùng (UHE). Môi trường trình chủ thực hiện nhận yêu cầu thực hiện công việc, xác nhận xem người dùng có được thực hiện công việc hay không (thông qua grid-mapfile), nếu xác nhận người dùng được phép thực hiện công việc thì MHE thực hiện khởi động một UHE cho người dùng. Sau đó chuyển yêu cầu đến cho UHE của người dùng thực hiện. UHE tạo ra một Managed Job Service (MJS) để thực hiện công việc. UHE thông báo cho client về tình trạng công việc và đầu ra của công việc.



Hình 9-5: Đệ trình công việc với GRAM

Khởi tạo dịch vụ: Khi thực hiện đệ trình công việc người dùng thực hiện gọi dịch vụ quản lý công việc MasterManagedJobFactoryService, dịch vụ này thực hiện khởi tạo một dịch vụ dành riêng cho người dùng, dịch vụ này cũng được dùng để quản lý trạng thái thực hiện của các công việc.

Đăng ký thông báo tình trạng công việc: Bất kỳ một công việc nào được thực hiện đều cần phải đăng ký với dịch vụ thông báo NotificationSink – Dịch vụ thông báo này chịu trách

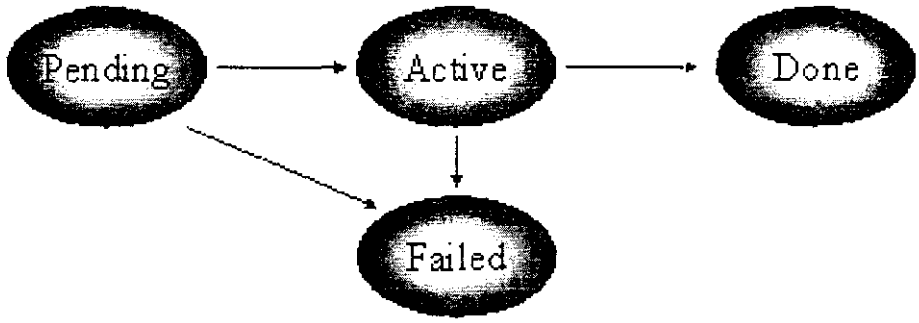
nhiệm thông báo về công việc mỗi khi tình trạng công việc được cập nhật trên tài nguyên ở xa.

Thực hiện công việc: Dịch vụ quản lý công việc ManagedJobService thay mặt người dùng thực hiện công việc trên tài nguyên đã lựa chọn.

Nhận thông báo, kết quả trả về: Khi tình trạng công việc được cập nhật trên tài nguyên ở xa thì dịch vụ thông báo NotificationSink sẽ thông báo cho người sử dụng biết, trong trường hợp công việc đã hoàn thành thì trả về kết quả đầu ra, hay thông báo lỗi nếu có.

- Trạng thái công việc được đệ trình

Hình dưới đây mô tả biểu đồ chuyển trạng thái của một công việc.



Hình 9-6: Biểu diễn trạng thái của một công việc

Sau khi được chấp nhận công việc ở trong trạng thái chờ đợi (*pending*) có nghĩa là các tài nguyên lưới cần thiết cho công việc chưa được xác định phục vụ sự thực thi của công việc này. Sau khi các tài nguyên đã được cấp phát, công việc sẽ được kích hoạt, công việc chuyển sang trạng thái hoạt động (*active*), sau khi tiến trình ứng với công việc kết thúc công việc có thể có hai trạng thái đã được thực hiện (*done*) hoặc thất bại (*failed*). Sự thất bại có thể do lỗi trong định dạng yêu cầu, lỗi do hệ thống quản lí tài nguyên .v.v Nguyên nhân của lỗi được ghi trong thông báo chuyển trạng thái của công việc. Sau khi tất cả các tiến trình đã kết thúc và các tài nguyên được giải phóng thì công việc trở về trạng thái được thực hiện (*done*).

9.2.2. Ngôn ngữ đặc tả tài nguyên - RSL

Các thành phần khác nhau của GRAM vận dụng ngôn ngữ RSL để thực hiện chức năng quản lý của chúng trong lúc hoạt động cùng các thành phần khác trong hệ thống. Ngôn ngữ RSL cung cấp những cú pháp cơ bản sử dụng để thực hiện các miêu tả tài nguyên phức tạp, với các thành phần tài nguyên đó, ghép chúng thành từng cặp <thuộc tính, giá trị>. Mỗi thuộc tính trong một phục vụ đặc tả tài nguyên tương đương với một thông số điều khiển hành vi của một hoặc nhiều thành phần trong hệ thống quản lý tài nguyên.

Bản chất ngôn ngữ RSL là sự kết hợp mối quan hệ giữa một thuộc tính và một giá trị được mô tả trong thẻ XML tương ứng. Các thuộc tính thường được sử dụng khi đệ trình một công việc là:

+ (directory=value) xác định đường dẫn được Job manager sẽ dùng làm thư mục mặc định để chạy công việc được yêu cầu.

+ (executable=value) tên của file thực thi sẽ chạy trên máy ở xa. Nếu giá trị là một GASS URL, file sẽ được truyền đến gass cache của máy đó trước khi thực hiện công việc và xóa bỏ sau khi đã kết thúc.

+ (arguments=value [value] [value] ...) các tham số dòng lệnh cho file thực thi. (stdin=value) Tên file được dùng làm đầu vào chuẩn cho file thực thi trên máy ở xa. Nếu giá trị là một GASS URL, file được truyền đến gass cache trước khi thực hiện công việc và xóa bỏ khi đã kết thúc.

+ (stdout=value) Tên của file trên máy thực hiện, lưu đầu ra chuẩn của công việc. Nếu giá trị là một GASS URL, đầu ra chuẩn của công việc sẽ được tự động chuyển về máy yêu cầu trong suốt quá trình thực hiện công việc.

+ (stderr=value) Tên của file lưu các lỗi chuẩn của công việc trên máy thực hiện. Nếu giá trị là một GASS URL, lỗi chuẩn của công việc được truyền một cách động về máy yêu cầu.

+ (file_stage_in=value [value] ...) danh sách các file cần chuyển đến máy được lựa chọn cùng file thực thi.

+ (file_clean_up=value [value] ...) danh sách các file cần xóa bỏ khi công việc kết thúc.

+ (count=value) Số lần thực hiện file thực thi. Mặc định là 1.

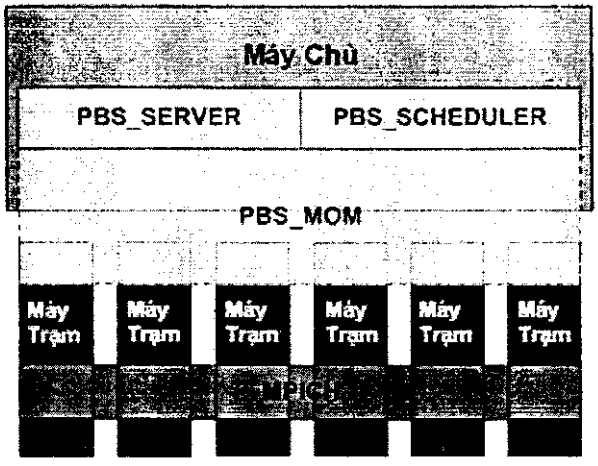
Ngoài ra còn các thuộc tính khác như: (environment=(var value) [value)] ...), (maxTime = value), (maxWallTime = value), (maxCpuTime = value), (jobType = single|multiple|mpi|condor), (gramMyJob = independent|collective), (queue = value), (project = value), (hostCount = value), (dryRun = yes|no), (minMemory = value), (maxMemory = value), (save_state = yes|no), (two_phase = <int>), (restart = <old JM contact>), (stdout_position = <int>), (stderr_position = <int>), (remote_io_url = <url base>). Thông tin chi tiết về các thẻ XML và các thông số cần thiết tương ứng có thể tham khảo ở Phụ lục.

9.2.3. PBS

9.2.3.1. Giới thiệu PBS

PBS (Portable Batch System) là hệ thống phân tải và quản lý tài nguyên rất mạnh do NASA phát triển. Hệ thống này được sử dụng rất phổ biến trong các hệ thống tính toán song song phân cụm. Nhiệm vụ chính của hệ thống PBS đó là cung cấp các khả năng về khởi tạo và lập lịch cho việc thực thi các công việc và sắp xếp các công việc đó trên các máy trạm (host). Hệ thống cũng cho phép xác định trước các nguồn tài nguyên sẽ được sử dụng khi thực thi công việc. Đồng thời cung cấp cơ chế để người sử dụng có thể chắc chắn là công việc của

mình sẽ được dùng các tài nguyên mà mình yêu cầu. Mô hình quan niệm logic của một hệ thống PBS đơn giản được trình bày trong hình sau:



Hình 9-7: Mô hình logic hệ thống cluster sử dụng PBS.

Như mô tả trên hình vẽ, PBS đảm nhận giao tiếp giữa Server và các máy trạm, máy chủ thông qua PBS quản lý hoạt động của các máy trạm. PBS bao gồm ba mô đun chính: PBS_Server, PBS_Scheduler, PBS_MOM [7]. PBS cung cấp rất nhiều chức năng cho các hệ thống tính toán hiệu năng cao. Sau đây là một số chức năng quan trọng nhất của PBS:

+ Chia sẻ tài nguyên: cung cấp một cơ chế lập lịch cho các công việc một cách trong suốt trên bất kỳ một hệ thống PBS nào, bởi bất kỳ người sử dụng nào có thẩm quyền. Công việc có thể được yêu cầu từ một máy client bất kỳ, cục bộ hay từ xa.

+ Giao diện đồ hoạ: giúp người sử dụng chuyển các yêu cầu tính toán ở chế độ lô (batch) hoặc chế độ tương tác (interactive); truy vấn các công việc, hàng đợi công việc và tình trạng hệ thống; và theo dõi sự tiến triển của các công việc. Ngoài ra PBS cũng hỗ trợ giao diện sử dụng chế độ dòng lệnh cho những người sử dụng chuyên nghiệp.

+ Cơ chế bảo mật: cho phép quản trị thiết lập hoặc huỷ bỏ việc truy cập đến PBS của một người sử dụng, một nhóm người, một máy hoặc một mạng nào đó.

+ Nhật ký: cho phép ghi lại tất cả các hoạt động trên của hệ thống theo từng người, từng nhóm người hoặc từng máy.

+ Tự động chuyển tệp: là cơ chế sao chép các tệp cần thiết cho việc thực hiện một công việc trên các máy trạm tính toán. Các tệp cần chuyển có thể là các tệp dữ liệu hoặc các tệp thực thi .

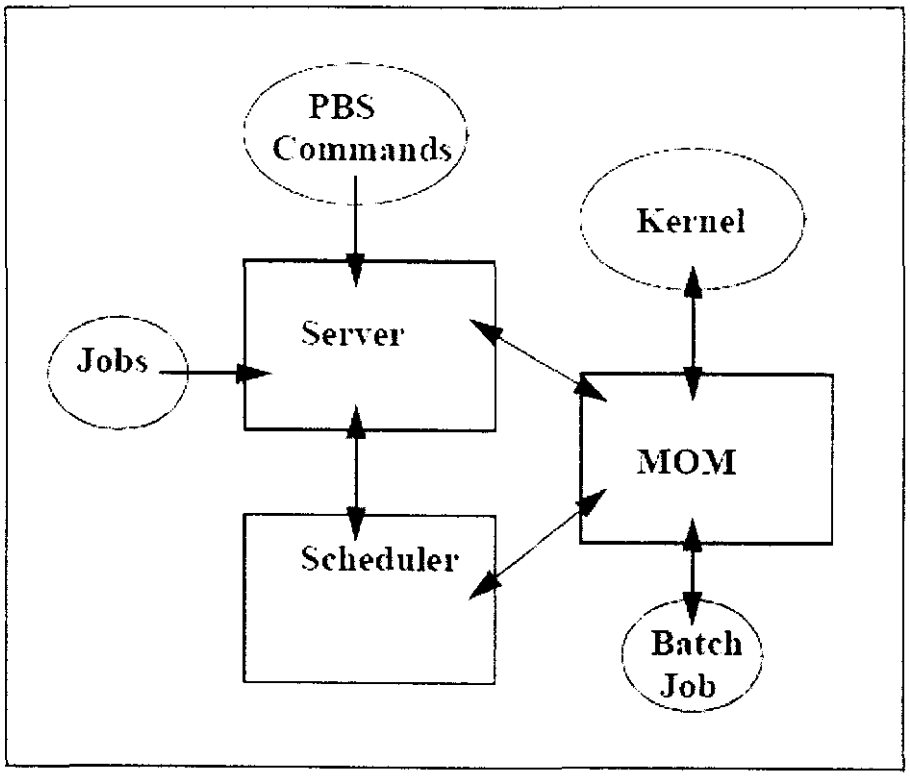
+ Hỗ trợ các công việc song song: cho phép hoạt động cùng với các thư viện lập trình song song như MPICH, MPI-LAM. Các chương trình có thể được lập lịch để chạy trên các hệ đa bộ xử lý hoặc trên các hệ thống đa máy tính.

+ Hỗ trợ tính toán lưới: cung cấp công nghệ siêu tính toán (meta-computing) và tính toán lưới, bao gồm việc hỗ trợ cho GGT (Globus Grid Toolkit). Việc hỗ trợ này chỉ được thực hiện trong phiên bản thương mại PBS Pro.

+ Giao diện lập trình được PBS: cung cấp để người lập trình có thể tự viết các lệnh mới cho PBS, tích hợp PBS vào các ứng dụng của họ hoặc cài đặt các cơ chế lập lịch riêng.
+ Tự động phân tải: là cơ chế cho phép phân tải các công việc trên các tài nguyên của hệ thống Cluster.

9.2.3.2. Các thành phần cơ bản của PBS

Hình 9-8 mô tả các thành phần chính của PBS: Mô-đun quản lý công việc (Job Server); Module thực hiện công việc (PBS Mom); và Module quản lý tài nguyên (Job Scheduler).
- Mô-đun quản lý công việc (Job Server) là thành phần trung tâm của PBS. Tất cả các thành phần khác của PBS giao tiếp với module quản lý công việc qua một địa chỉ IP duy nhất. Chức năng của module quản lý công việc là cung cấp các dịch vụ, như là nhận/tạo ra các công việc lô (batch công việc), thay đổi các công việc, bảo vệ công việc khi có sự cố hệ thống, và thực hiện công việc (chuyển công việc cho module thực thi công việc). Module này quản lý một hoặc nhiều hàng đợi công việc (queue), một công việc phải thuộc vào một hàng đợi. Các hàng đợi được server quản lý bởi một tập thuộc tính như kiểu, tài nguyên, tên,...
- Module thực hiện công việc (PBS Mom) được chia nhỏ thành các thành phần chức năng sau:
+ Job Executor (JE): Job Executor là tiến trình ngầm chịu trách nhiệm thực thi công việc. Khi thực hiện, JE nhận một bản copy công việc từ Server, xử lý công việc, và sau đó có thể sẽ đảm nhận luôn công việc trả kết quả về cho user nếu được Server yêu cầu. Mỗi một tiến trình ngầm JE chạy trên một máy tính trạm trong mạng.
+ Resource Monitor (RM): là bộ phận chịu trách nhiệm theo dõi, kiểm tra các nguồn tài nguyên của hệ thống và báo cho bộ lập lịch Scheduler.

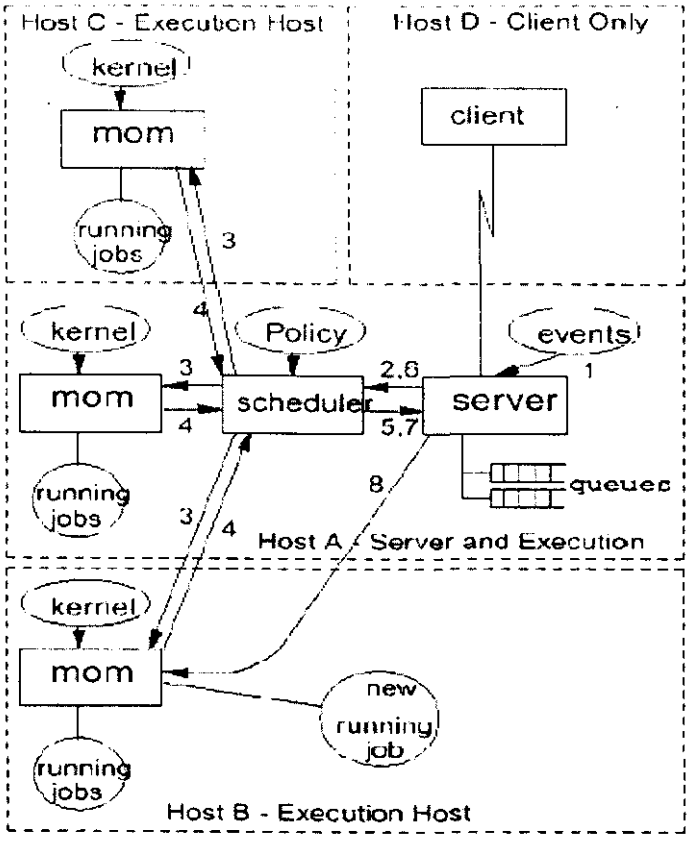


Hình 9-8: Các thành phần trong PBS

- Mô-đun quản lý tài nguyên (Job Scheduler) là thành phần chịu trách nhiệm xác định xem công việc nào sẽ chạy và chạy trên tài nguyên nào.

Tương tác giữa các module trên có thể được mô tả như hình 9-9:

1. Sự kiện kích hoạt module quản lý công việc bắt đầu một vòng lập lịch.
2. Module quản lý công việc gửi lệnh lập lịch cho bộ lập lịch.
3. Module lập lịch yêu cầu thông tin về tài nguyên từ module quản lý tài nguyên.
4. Module quản lý tài nguyên trả về các thông tin yêu cầu.
5. Bộ lập lịch yêu cầu các thông tin từ module quản lý công việc.
6. Module quản lý công việc gửi thông tin trạng thái của các công việc tới bộ lập lịch, bộ lập lịch đưa ra chính sách quyết định việc thực hiện công việc.
7. Bộ lập lịch gửi yêu cầu thực hiện công việc đến module quản lý công việc.
8. Module quản lý công việc gửi công việc đến các module thực thi công việc để thực hiện.



Hình 9-9: Cơ chế hoạt động của PBS

9.2.3.3. Đệ trình công việc trong PBS

Người sử dụng có thể thực hiện lệnh trực tiếp trên màn hình console để truyền công việc cũng như các yêu cầu của mình tới Server. Các shell chuẩn của linux có thể giúp ta biên dịch mã lệnh để thực hiện các job. Sau đây là một ví dụ dùng Bourne Shell (sh):

```
1    #!/bin/sh
2    #PBS -l walltime=1:00:00
3    #PBS -l mem=400mb
4    #PBS -l ncpus=4
5    #PBS -j oe
6
7    ./subrun
```

Lệnh 1: Cố định cho mọi shell script. Nó chỉ ra shell nào sẽ được sử dụng để thực hiện script.

Lệnh 2-5. Điều khiển PBS: PBS sẽ đọc file script cho đến khi nó không còn thấy xuất hiện dòng điều khiển nữa thì thôi, và coi các dòng còn lại là các lệnh mà user muốn thực hiện. Trong trường hợp này thì dòng 6, 7 được coi là các dòng lệnh.

Trong ví dụ ở trên, dòng 2 đến 4 chỉ ra các yêu cầu tài nguyên sau "-l". Cụ thể là yêu cầu 1 giờ walltime, 400 mb bộ nhớ và 4 CPU. Dòng 5 không phải là dòng điều khiển tài nguyên. Nó yêu cầu PBS hợp nhất các luồng ra stdout và stderr vào một luồng duy nhất ("-j oe"). Các điều khiển PBS chia thành hai loại: yêu cầu tài nguyên và lựa chọn hành vi công việc. Dòng 7 là lệnh cần thực hiện.

Lệnh qsub là lệnh dùng để đệ trình công việc mình muốn thực hiện lên server. Cấu trúc lệnh như sau:

% qsub [option] mysubrun

trong đó option là các lựa chọn khi thực hiện công việc và mysubrun là công việc cần thực hiện. Các tuỳ chọn dùng để mô tả tài nguyên; trong ví dụ sau:

% qsub -l ncpus=16, walltime=4:0:0 mysubrun

ncpus là số CPU sử dụng để thực hiện công việc

walltime là thời gian dự kiến thực hiện công việc

Các công việc cũng có thể được miêu tả trong một file đặc tả, trong file thực hiện mô tả tất cả các thông số tài nguyên cần sử dụng, đầu vào, đầu ra vân ... vân ... Ta lấy một ví dụ:

```
1  #!/bin/sh
2  #PBS -l walltime=1:00:00
3  #PBS -l mem=400mb
4  #PBS -l ncpus=4
5  #PBS -j oe
```

Khi đó việc đệ trình công việc trong PBS sẽ được thực hiện bằng một lệnh đơn giản như sau:

```
% qsub  mysubrun
```

Trong đó mysubrun là file mô tả công việc. Như vậy để có thể đưa một công việc từ trên lưới xuống cho PBS_Server thì ta phải làm thông suốt quá trình chuyển đổi công việc từ file đặc tả RSL của Globus sang file miêu tả công việc của PBS, vấn đề này sẽ được đề cập trong phần tiếp theo của đồ án.

9.2.4. Các yêu cầu đối với thành phần kết nối

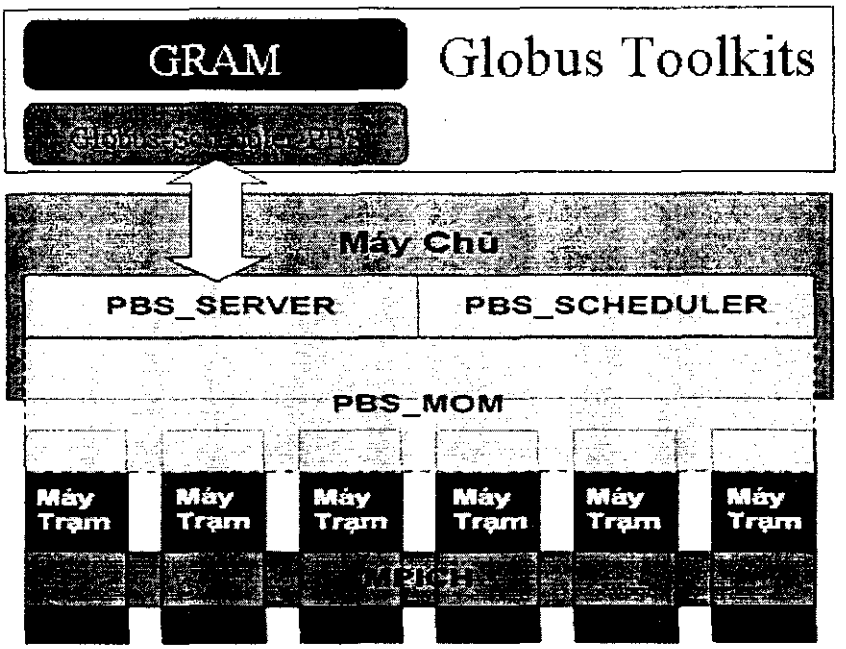
Thực chất của việc kết nối một nút lưới với một Cluster chính là coi Cluster là một tài nguyên lưới. Tài nguyên này cũng như các tài nguyên khác, cung cấp một giao diện truy cập thống nhất cho các thành viên lưới khác sử dụng. Tất nhiên là các tài nguyên lưới khác nhau thì các dịch vụ mà chúng có khả năng cung cấp cũng khác nhau, ví dụ tài nguyên cluster có khả năng chạy các chương trình song song theo chuẩn MPI còn tài nguyên là máy tính đơn lẻ thì không có khả năng này. Về mặt công nghệ, Globus toolkit cung cấp các modul cho phép các

thành phần của globus giao tiếp được với hầu hết các hệ quản lý cluster phổ biến như PBS, LSF, Condor...

Như vậy để biến một cluster thành một tài nguyên lưới ta thực hiện cài Globus Toolkit lên nút chủ của cluster, thêm một mô-đun thực hiện giao tiếp với nút chủ của cluster và thực hiện các cấu hình cần thiết để cho Globus có thể giao tiếp được với thành phần quản lý tài nguyên địa phương của cluster. Để có thể thấy được sự liên kết thực hiện công việc giữa lưới và Cluster, chúng ta sẽ xem xét một công việc thực hiện trên Cluster thông qua lưới như thế nào?

Thông thường ở trên lưới luôn tồn tại hai loại công việc có sự khác nhau rõ rệt. Thứ nhất là những công việc được thực hiện qua dịch vụ lưới. Những dịch vụ lưới này được các lưới cung cấp sẵn cho người sử dụng, người sử dụng chỉ việc đưa ra một bộ dữ liệu đầu vào qua GridPortal và nhận kết quả trả về, dịch vụ lưới được triển khai trên nút nào và được thực hiện như thế nào thì người dùng hoàn toàn không biết. Sau khi đưa dữ liệu thông qua GridPortal thì lưới sẽ thực hiện tìm kiếm, môi giới tài nguyên và lập lịch cho người sử dụng, sau đó sẽ đệ trình công việc đến các dịch vụ lưới tương ứng.

Loại công việc thứ hai là những công việc có tính chuyên môn cao như bài toán dự báo động đất, nghiên cứu chuyển động của các hành tinh, ... đòi hỏi khối lượng tính toán lớn, đối với những công việc này thì người sử dụng phải biết rõ về tài nguyên sẽ thực hiện công việc (siêu máy tính hay Cluster), viết chương trình thực hiện đối với loại tài nguyên tương ứng, sau đó đệ trình công việc thông qua giao diện GridPortal, lưới sẽ tìm ra tài nguyên tương ứng dựa trên thời gian thực hiện (walltime) và số nút tính toán (node), lưới thực hiện truyền công việc cho cluster cùng với các tham số để thực hiện, cluster thực hiện công việc và sau đó trả lại kết quả cho lưới sau đó lưới mới thực hiện trả kết quả về cho người dùng. Việc kết nối giữa lưới dựa trên GT và cluster dựa trên PBS thực chất là thực hiện giao tiếp giữa thành phần GRAM trong lưới và thành phần PBS Server như mô tả trong hình sau.



Hình 9-10: Giao tiếp giữa GRAM & PBS_Server

Trong đó:

+ GRAM: cung cấp một giao diện chuẩn cho việc yêu cầu và sử dụng tài nguyên hệ thống từ xa để thực hiện các công việc. Đối với các công việc chỉ yêu cầu máy tính đơn hay siêu máy tính thì GRAM thực hiện công việc ngay trên máy tính đó. Đối với những công việc yêu cầu thực hiện trên Cluster thì GRAM sẽ chuyển nội dung công việc cho thành phần Globus-scheduler-pbs.

+ Globus-scheduler-pbs: thành phần này có nhiệm vụ chuyển nội dung công việc sang dạng mà PBS_Server có thể hiểu được.

+ PBS_Server: là nút chủ thực hiện lập lịch (địa phương), quản lý và thực hiện phân chia công việc thực hiện trên các nút con.

+ MPI: thư viện truyền thông điệp giữa các nút con trong Cluster.

Như vậy có thể thấy việc kết nối cluster vào lưới không có nghĩa là chúng ta phải quản lý đến từng máy đơn trong cluster mà quản lý cluster một cách tổng thể. Lưới sẽ quản lý cả cụm máy tính trong cluster như một tài nguyên đơn nhất thông qua bộ quản lý tài nguyên địa phương PBS.

9.2.5. Thành phần Globus Scheduler PBS

9.2.5.1. Giới thiệu Globus Scheduler PBS

Globus Toolkit có cung cấp cho chúng ta một số dịch vụ để giao tiếp với PBS_Server. Những dịch vụ này nằm trong gói cài đặt scheduler-pbs-3.2-src_bundle.tar.gz [11]. Khi chúng ta thực hiện cài đặt gói này thì các thành phần sau sẽ được thêm vào:

mmjfs_pbs_setup	Gói này tạo ra: $GLOBUS_LOCATION/setup/globus/setup-mmjfs-pbs. Chứa các thông tin cấu hình để thêm service: MasterPbsManagedJobFactoryService vào trong các service của GT.
mjs_pbs_setup	Gói này tạo ra: $GLOBUS_LOCATION/setup/globus/setup-mjs-pbs. Chứa các thông tin cấu hình để thêm service: PbsManagedJobFactoryService vào trong các service của GT.
rips_pbs_provider_setup	Gói này tạo ra: $GLOBUS_LOCATION/setup/globus/setup-pbs-provider. Chứa mã Bourne Shell để yêu cầu PBS thực hiện các công việc và báo cáo cho dịch vụ cung cấp thông tin RIPS để xử lý.
globus_gram_job_manager_setup_pbs	Gói này tạo ra: $GLOBUS_LOCATION/setup/globus/setup-globus-job-manager-pbs. Chứa mã Perl để tương tác với PBS. Dịch ngôn ngữ RSL thành công việc cho PBS.

Bảng 9-1: Các thành phần của Globus-Scheduler-PBS

Sau khi cài đặt xong thì việc tiếp theo là cấu hình cho hệ thống. Đây thực sự là một vấn đề không đơn giản và đã chiếm mất khá nhiều thời gian trong quá trình làm đồ án. Chi tiết về việc cấu hình và xây dựng một shell script cấu hình tự động sẽ được trình bày trong chương III của đồ án. Khi việc cấu hình hoàn tất, thực hiện khởi động trình chứa (globus-start-container) thì trong các dịch vụ của Globus Toolkit sẽ có thêm hai dịch vụ:

http://hostname:8080/ogsa/services/base/gram/PbsManagedJobFactoryService

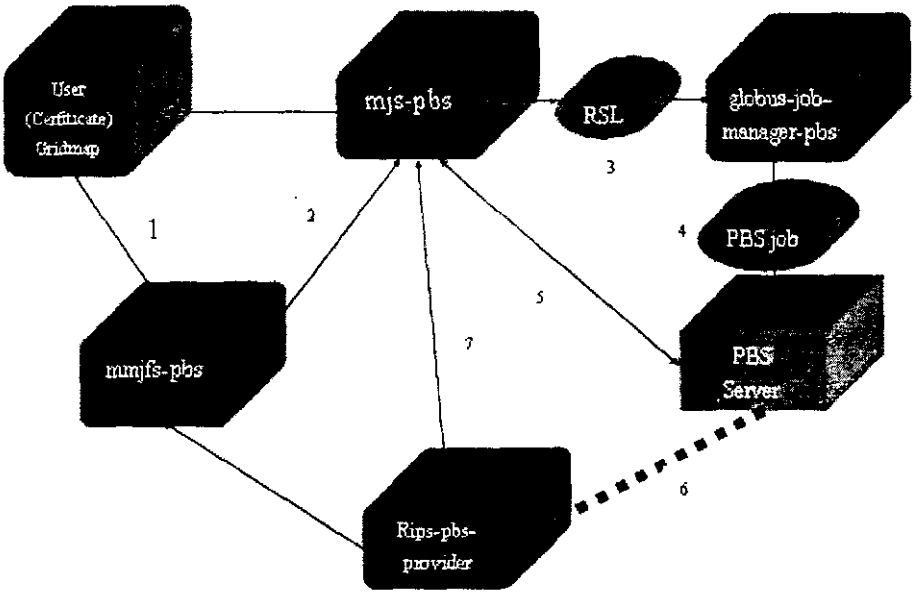
http://hostname:8080/ogsa/services/base/gram/MasterPbsManagedJobFactoryService

Dịch vụ MasterPbsManagedJobFactoryService đóng vai trò là một giao diện đệ trình công việc, trực tiếp nhận yêu cầu công việc từ phía client. Đối với từng công việc dịch vụ MasterPbsManagedJobFactoryService sẽ gọi đến dịch vụ PbsManagedJobFactoryService sẽ sinh ra các thể hiện (instance) tương ứng trực tiếp làm việc với client.

Như vậy các thành phần dịch vụ này sẽ giúp cho việc đệ trình công việc trên lưới trở nên đơn giản hơn, người sử dụng không cần phải biết các lệnh thực hiện, theo dõi công việc trên PBS mà vẫn có thể đệ trình công việc và nhận kết quả trả về.

9.2.5.2. Hoạt động của Globus Scheduler Pbs

Như đã trình bày ở trên, Globus Scheduler Pbs gồm 4 thành phần (mjs-pbs, mmjfs-pbs, rips-pbs-provider, globus-job-manager-pbs) giúp cho việc hoạt động đệ trình công việc lên PBS_Server thực hiện được. Giao tiếp giữa các thành phần này được mô tả trong hình sau.



Hình 9-11: Hoạt động của Globus Scheduler Pbs

(1). Người dùng có giấy chứng nhận được cấp bởi nhà thẩm quyền, được thực hiện trên tài nguyên lưới (Quy định trong Grid-mapfile) thực hiện đệ trình công việc với trình quản lý công việc MMJFS-PBS.

(2). Trình quản lý công việc MMJFS-PBS thực hiện sinh ra một thể hiện (instance) quản lý công việc phục vụ cho người dùng MJS-PBS.

(3). Trình quản lý công việc MJS-PBS thực hiện lấy file đặc tả công việc của người dùng và chuyển cho thành phần biên dịch Globus-job-manager-pbs.

(4). Thành phần Globus-job-manager-pbs thực hiện chuyển file đặc tả người dùng thành công việc cho PBS_Server.

(5). Trình quản lý công việc người dùng yêu cầu PBS_Server thực hiện công việc.

(6). PBS_Server thực hiện công việc và đưa kết quả trả về cho thành phần quản lý thông tin tài nguyên RIPS-PBS.

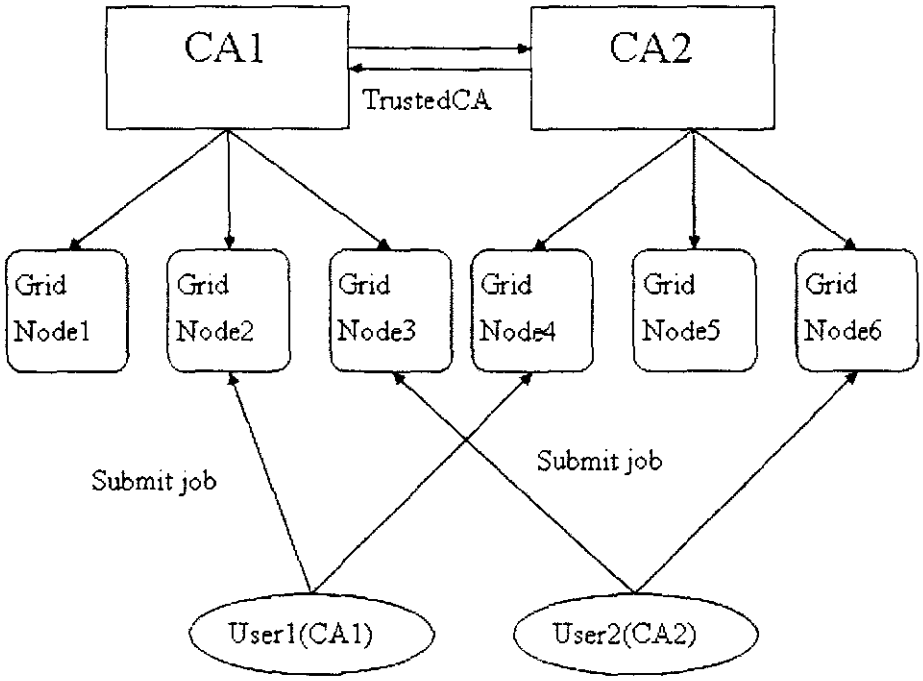
(7). RIPS-PBS trả kết quả về cho trình quản lý công việc người dùng.

(8). Trình quản lý công việc người dùng MJS-PBS trả kết quả về cho người sử dụng.

9.2.6. Kết nối lưới với lưới

Để đáp ứng nhu cầu tính toán ngày càng tăng của người sử dụng, các lưới con đã liên kết với nhau thành một môi trường lưới có phạm vi toàn cầu, tạo điều kiện cho người sử dụng có nhiều khả năng lựa chọn nguồn tài nguyên hơn khi thực hiện công việc của mình. Lưới là

một cộng đồng chia xẻ tài nguyên do vậy thực chất việc kết nối giữa hai lưới chính là làm sao cho các thành viên của lưới này tin tưởng và sử dụng được các tài nguyên của lưới kia và ngược lại. Việc này được thực hiện bằng cách tạo các tài khoản người dùng cần thiết cho việc sử dụng tài nguyên qua lại giữa các thành viên trong hai lưới, cấp các giấy chứng nhận, trao chuyển giấy chứng nhận của các nhà thẩm quyền của hai lưới để hai bên tin tưởng lẫn nhau...



Hình 9-12: Sự "tin tưởng" lẫn nhau giữa các nhà thẩm quyền

Để mô phỏng việc kết nối giữa hai lưới ta sẽ chia các tài nguyên trong lưới thành hai nhóm (coi như thuộc về hai lưới). Thông thường mỗi một lưới khi xây dựng đều cử ra một nhà chứng nhận có thẩm quyền (CA). CA này có trách nhiệm cung cấp giấy chứng nhận cho tài nguyên, dịch vụ và người dùng, khi đó các tài nguyên, dịch vụ, người dùng này được "tin tưởng" trên lưới con đó. Khi người sử dụng của lưới con này đem giấy chứng nhận của mình sang lưới con khác thì nhà CA của lưới đó không hề biết đến giấy chứng nhận này. Người sử dụng muốn sử dụng tài nguyên trên lưới thứ hai này thì lại phải được nhà thẩm quyền bên này chấp nhận. Như vậy, chúng ta đã có thể thấy rõ bản chất của việc hai lưới con liên kết với nhau là nhà CA của hai bên phải tin tưởng lẫn nhau (hoặc hai lưới có thể sử dụng giấy chứng nhận của một nhà CA nào đó có uy tín trên thế giới).

Mỗi một nhà thẩm quyền (CA) đều có một thư mục riêng gọi là Trusted CA, trong đó có chứa thông tin về các loại giấy chứng nhận mà nhà thẩm quyền này "tin tưởng". Khi nhà thẩm quyền CA1 tin tưởng nhà thẩm quyền CA2 thì nhà thẩm quyền CA1 sẽ thực hiện lưu thông tin về giấy chứng nhận do nhà thẩm quyền CA2 ký, và ngược lại nhà thẩm quyền CA2 cũng

phải làm tương tự, khi đó hai nhà thẩm quyền này đã hoàn toàn tin tưởng lẫn nhau. Hay nói theo một cách khác là môi trường lưới đã được thiết lập chung, người sử dụng thuộc một trong hai lưới con có thể hoàn toàn sử dụng tài nguyên trên cả hai lưới.

9.3. Xây dựng công cụ kết nối tự động

Sau khi tiến hành cấu hình thành công để lưới nhận cluster là một tài nguyên của lưới, nhiệm vụ tiếp theo của đồ án là cung cấp một công cụ cấu hình tự động giúp người dùng có thể giảm tối thiểu thời gian trong việc cấu hình. Trong chương này chúng ta sẽ lần lượt xem xét các cấu hình cần thiết phải thiết lập cho cluster, cho lưới từ đó tổng hợp các lệnh cấu hình trong một file shell script. Với shell script này người dùng chỉ phải chạy một lần để cho lưới nhận cluster là tài nguyên của nó. Phần cuối của chương sẽ nếu các kết quả thực hiện trên trung tâm HPCC, mô hình lưới được thiết lập trên trung tâm.

9.3.1.1. Cấu hình cluster-based PBS

Công việc cấu hình cluster-based PBS được thực hiện trên nút chủ của cluster. PBS đã cung cấp cho chúng ta môt lệnh quản lý cấu hình PBS dùng cho việc cấu hình trên giao diện dòng lệnh (console) đó là lệnh qmgr [7]. Khi thực hiện lệnh này trên màn hình (console) sẽ xuất hiện dấu nhắc, khi đó mọi lệnh thực hiện sẽ thực thi sẽ được dùng cho việc cấu hình PBS.

Các lệnh cấu hình PBS

Cấu trúc của một lệnh cấu hình có dạng như dưới đây:

command server [names] [attr OP value[,...]]

command queue [names] [attr OP value[,...]]

command node [names] [attr OP value[,...]]

Ở đây "command" thực hiện một tác động lên các đối tượng "server", "queue", "node". Các tác lệnh tương ứng là:

Lệnh	Miêu tả
active	Kích hoạt một đối tượng.
create	Tạo một đối tượng mới (lệnh này chỉ thực hiện với hàng đợi (queue) và nút thực hiện (node))
delete	Xóa một đối tượng (lệnh này cũng chỉ thực hiện với hàng đợi (queue) và nút thực hiện (node))
help	Hiển thị thông tin giúp đỡ hoặc thông tin chi tiết về lệnh nào đó.
list	Liệt kê tất cả các thuộc tính và giá trị của một đối tượng.

Lệnh	Miêu tả
print	Hiển thị các hàng đợi đã được thiết lập và các thuộc tính của máy chủ (server)
set	Định nghĩa hay thêm vào các giá trị cho thuộc tính của một đối tượng.
unset	Loại bỏ thuộc tính của một đối tượng.

Bảng 9-2: Các lệnh cấu hình PBS

Các đối số của lệnh

Đối số	Miêu tả
names	Danh sách tên các đối tượng chịu sự tác động của lệnh.
attr	Thuộc tính của đối tượng sẽ được thay đổi.
OP	Tác động được thực hiện với đối tượng.
=	Thiết lập giá trị thuộc tính cho đối tượng
+	Tăng giá trị của thuộc tính
-	Giảm giá trị của thuộc tính
value	Giá trị được thiết lập cho thuộc tính

Bảng 9-3: Các đối số của lệnh cấu hình PBS

9.3.1.2. Cấu hình cho đối tượng Server:

Thuộc tính của Server có rất khá nhiều, từ các thuộc tính cung cấp thông tin cho đến các thuộc tính điều khiển. Sau đây sẽ xem xét chi tiết một số thuộc tính phục vụ cho việc cấu hình:

Thuộc tính	Miêu tả
scheduling	Kích hoạt hay không kích hoạt bộ lập lịch của PBS. Nếu được kích hoạt (scheduling = True) thì bộ lập lịch sẽ được gọi mỗi khi yêu cầu một công việc. Nếu không được kích hoạt (scheduling = False) thì công việc yêu cầu sẽ được thực hiện theo quy định của nhà quản trị.
max_user_run	Số công việc tối đa một người dùng có thể thực hiện tại một thời điểm.
acl_host_enable	Cho phép Server sử dụng danh sách máy trong acl_hosts

Thuộc tính	Miêu tả
acl_hosts	Danh sách các máy có thể yêu cầu dịch vụ từ Server set server acl_hosts = *.parallel
default_queue	Hàng đợi mặc định được sử dụng trong trường hợp yêu cầu không chỉ ra tên hàng đợi.
log_events	Độ dài xâu ký tự sẽ ghi lại các sự kiện xảy ra trong hệ thống
mail_from	Định danh của người dùng mà Server sẽ dùng để gửi thư cho người dùng.
query_other_jobs	Cho phép người dùng không sở hữu công việc (job) xem tình trạng của công việc.
resources_default	Tập các giá trị được gán cho các công việc chạy trên Server khi chúng không chỉ ra các thông số cụ thể.
scheduler_iteration	Khoảng thời gian tính bằng giây giữa các lần thực lập lịch các công việc.
default_node	Nút tính toán mặc định được dùng để chạy công việc trong trường hợp yêu cầu không chỉ rõ tên nút.

Bảng 9-4: Thuộc tính của lệnh cấu hình PBS

Trên cơ sở các thuộc tính của Server việc cấu hình cho server được thực hiện bằng các lệnh sau:

Set server attributes.

set server scheduling = True → thực hiện lập lịch.

set server max_user_run = 6 → số người dùng lớn nhất là 6

set server acl_host_enable = True → cho phép sử dụng acl_hosts

set server acl_hosts = *.parallel → các máy acl_hosts là *.paralell

set server default_queue = default → hàng đợi mặc định là default

set server log_events = 63 → kích thước bản ghi log khi có sự kiện

set server mail_from = adm

set server query_other_jobs = True

set server resources_default.cput = 01:00:00 → thời gian sử dụng mặc định là 1 giờ

set server resources_default.neednodes = 1 → số nút cần mặc định là 1

set server resources_default.nodect = 1

set server resources_default.nodes = 1

set server scheduler_iteration = 60

set server default_node = 1 → nút thực hiện mặc định là nút số 1

9.3.1.3. Cấu hình đối tượng hàng đợi (queue)

Trong PBS có hai kiểu hàng đợi khác nhau: hàng đợi định tuyến (routing) và hàng đợi thực thi (execution). Một hàng đợi định tuyến dùng để chuyển công việc từ hàng đợi này sang một hàng đợi khác trong PBS Server. Còn hàng đợi thực thi dùng để thực hiện công việc, công việc tồn tại ở trong hàng đợi trong suốt quá trình nó chạy. Thông thường một PBS Server muốn hoạt động tốt phải có cả hai loại hàng đợi này.

Các thuộc tính của hàng đợi chia thành ba nhóm:

Nhóm thuộc tính chỉ áp dụng với hàng đợi thực thi

Nhóm thuộc tính chỉ áp dụng với hàng đợi định tuyến

Nhóm thuộc tính áp dụng với cả hai kiểu hàng đợi

Hệ thống Cluster được thiết lập với một hàng đợi định tuyến dùng để phân chia công việc theo mức độ lớn nhỏ khác nhau và 4 hàng đợi thực thi công việc dùng để thực hiện các công việc với mức độ lớn nhỏ tương ứng.

Hàng đợi định tuyến được thiết lập với các giá trị thuộc tính sau:

Thuộc tính	Miêu tả
queue_type = Route	Kiểu hàng đợi là hàng đợi định tuyến, dùng để phân chia công việc đến các hàng đợi khác.
max_running	Số lượng công việc tối đa nằm trong hàng đợi tại một thời điểm
route_destinations	Danh sách các hàng đợi mà công việc sẽ được chuyển đến.
enabled	Nếu enabled = True thì hàng đợi sẽ nhận thêm các công việc mới, ngược lại thì sẽ không nhận thêm các công việc mới.
started	Công việc sẽ được lập lịch để thực thi từ hàng đợi này. Nếu started = False thì hàng đợi sẽ ngừng thực hiện.

Bảng 9-5: Các thuộc tính của hàng đợi định tuyến

Create and define queue default

create queue default → hàng đợi mặc định

set queue default queue_type = Route → hàng đợi định tuyến

set queue default max_running = 10 → số lượng công việc tối đa = 10

set queue default route_destinations = small → lựa chọn hàng đợi thực thi

set queue default route_destinations += medium

set queue default route_destinations += long

set queue default route_destinations += verylong

set queue default enabled = True

set queue default started = True

Các hàng đợi thực hiện công việc với các thuộc tính:

Thuộc tính	Miêu tả
queue_type = Execution	Kiểu hàng đợi là hàng đợi thực thi công việc, dùng để thực hiện các công việc được gửi đến nó.
Priority	Mức độ ưu tiên của hàng đợi này so với các hàng đợi khác trong PBS Server
max_running	Số lượng công việc tối đa nằm trong hàng đợi tại một thời điểm
resources_max	Lượng tài nguyên lớn nhất có thể được yêu cầu cho một công việc đơn.
resources_min	Lượng tài nguyên nhỏ nhất được yêu cầu cho một công việc đơn.
resources_default	Tập các giá trị được gán cho các công việc chạy trên hàng đợi khi chúng không chỉ ra các thông số cụ thể.
enabled	Nếu enabled = True thì hàng đợi sẽ nhận thêm các công việc mới, ngược lại thì sẽ không nhận thêm các công việc mới.
started	Công việc sẽ được lập lịch để thực thi từ hàng đợi này. Nếu started = False thì hàng đợi sẽ ngừng thực hiện.

Bảng 9-6: Các thuộc tính của hàng đợi thực thi

Các hàng đợi tương ứng với các mức độ ưu tiên công việc khác nhau được tạo ra như sau:

Ta thực hiện tạo ra một hàng đợi công việc thực hiện những công việc nhỏ, hàng đợi có tên là small, là hàng đợi thực thi, mức độ ưu tiên 100, dùng để thực hiện các công việc có thời gian yêu cầu ít hơn 20 phút.

Create and define queue small

create queue small → hàng đợi thực hiện loại nhỏ

set queue small queue_type = Execution → hàng đợi thực thi

set queue small Priority = 100 → độ ưu tiên 100

set queue small max_running = 10

set queue small resources_max.cput = 00:20:00 → thời gian yêu cầu thực hiện

set queue small resources_default.cput = 00:20:00 → thời gian yêu cầu thực hiện

set queue small enabled = True

set queue small started = True

Đối với các công việc vừa, đòi hỏi thời gian thực hiện nhiều hơn từ khoảng lớn hơn 20 phút đến nhỏ hơn 2 giờ ta tạo một hàng đợi có tên là medium, hàng đợi này có mức độ ưu tiên thực hiện là 80.

Create and define queue medium

create queue medium → hàng đợi cỡ trung bình

set queue medium queue_type = Execution → hàng đợi thực thi

set queue medium Priority = 80 → độ ưu tiên 80

set queue medium max_running = 10

set queue medium resources_max.cput = 02:00:00 → thời gian thực hiện

set queue medium resources_min.cput = 00:20:01

set queue medium resources_default.cput = 02:00:00

set queue medium enabled = True

set queue medium started = True

Đối với các công việc đòi hỏi thời gian thực hiện dài hơn, lớn hơn 2 giờ và nhỏ hơn 12 giờ ta tạo ra một hàng đợi công việc tên là long với mức độ ưu tiên là 60.

Create and define queue long

create queue long → hàng đợi công việc cỡ lớn

set queue long queue_type = Execution → hàng đợi thực thi

set queue long Priority = 60 → độ ưu tiên 60

set queue long max_running = 10

set queue long resources_max.cput = 12:00:00 → thời gian thực hiện

set queue long resources_min.cput = 02:00:01

set queue long resources_default.cput = 12:00:00

set queue long enabled = True

set queue long started = True

Đối với các công việc đòi hỏi thời gian sử dụng tài nguyên lớn hơn 72 giờ ta tạo ra một hàng đợi công việc tên là verylong với độ ưu tiên là 40 để thực hiện các công việc này.

Create and define queue verylong

create queue verylong → hàng đợi công việc rất dài

set queue verylong queue_type = Execution → hàng đợi thực thi

set queue verylong Priority = 40 → độ ưu tiên 40

set queue verylong max_running = 10

set queue verylong resources_max.cput = 72:00:00 → thời gian thực hiện

set queue verylong resources_min.cput = 12:00:01

set queue verylong resources_default.cput = 72:00:00

set queue verylong enabled = True

set queue verylong started = True

Trên đây là các thông số cấu hình được sử dụng thực tế trên trung tâm tính toán hiệu năng cao HPC, với những thông số này hệ thống Cluster ở đây đã hoạt động khá hiệu quả, đã xử lý tốt các bài toán do Viện vật lý kỹ thuật yêu cầu. Do vậy đây là những thông số mặc định

gắn cứng vào shell script cấu hình tự động, trong trường hợp cần tùy biến theo yêu cầu của người sử dụng thì có thể thay đổi các thông số này.

9.3.2. Cấu hình lưới dựa trên GT

Trước khi thực hiện cấu hình cho lưới ta cần phải thực hiện cài đặt Globus Toolkits 3.2.1 trên nút chủ của cluster, cài đặt gói phần mềm scheduler-pbs-3.2-src_bundle.tar.gz. Như ta đã biết, Globus Toolkit 3.2.1 và Kiến trúc dịch vụ lưới mở (OGSA) cung cấp một nền tảng chuẩn cho việc phát triển các dịch vụ lưới. Trong số các dịch vụ do GT3.2.1 cung cấp, dịch vụ Master Managed Job Factory Service (MMJFS) là một dịch vụ quản lý công việc (job manager). MMJFS cung cấp khả năng đệ trình công việc và sử dụng các nguồn tài nguyên ở xa cho việc thực hiện công việc [16]. Trình quản lý công việc là một hệ thống thực hiện việc đệ trình, điều khiển và theo dõi tình trạng công việc trên một hay nhiều tài nguyên. Mỗi một công việc được lập lịch cho việc thực hiện theo các chính sách hệ thống cung cấp và các nguồn tài nguyên hiện có. Một trong những trình quản lý công việc được sử dụng rộng rãi hiện nay là phần mềm Quản lý và phân tải PBS. Sau đây sẽ là một số bước cấu hình cần thiết để thiết lập PBS là trình quản lý công việc cho lưới.

```
% gpt-postinstall
```

lệnh này thực hiện khởi tạo các cấu hình mặc định cho lưới.

```
$GLOBUS_LOCATION/setup/globus/setup-mmjfs-pbs
```

lệnh này thực hiện cấu hình để thêm dịch vụ MasterPbsManagedJobFactoryService vào trong các dịch vụ của Globus Toolkit.

```
$GLOBUS_LOCATION/setup/globus/setup-mjs-pbs
```

Lệnh này thực hiện cấu hình để thêm dịch vụ PbsManagedJobFactoryService vào trong các dịch vụ của Globus Toolkit.

```
$GLOBUS_LOCATION/setup/globus/setup-pbs-provider
```

Lệnh này thực hiện các cấu hình để thành phần RIPS (Resource Information Provider) trong lưới làm việc với PBS. RIPS nhận các thông tin trả về từ PBS_Server và cung cấp cho trình quản lý công việc.

$GLOBUS_LOCATION/setup/globus/setup-globus-job-manager-pbs --cpu-per-node=1

Lệnh này thiết lập số lượng các bộ vi xử lý (CPU) trên một nút đơn trong cluster, thông số này phải phù hợp với tài nguyên tương ứng trong cluster.

$GLOBUS_LOCATION/setup/globus/setup-globus-job-manager-pbs --remote-shell=rsh

Lệnh này thực hiện việc chuyển đổi remote-shell mặc định thành rsh, thông thường các hệ thống tính toán song song phân cụm thường dùng rsh để truy cập các tài nguyên trong cluster.

$GLOBUS_LOCATION/setup/globus/setup-globus-job-manager-pbs –validate-queues=yes

Lệnh này thực hiện kiểm tra hàng đợi tương ứng trong PBS phù với yêu cầu hay không.

9.3.3. Shell script thực hiện việc kết nối

Quá trình triển khai kết hợp tính toán song song phân cụm vào tính toán lưới đòi hỏi người thực hiện phải có những kiến thức khá sâu về cả hai mô hình tính toán này. Đây thực sự là một điều khó khăn đối với những người mới làm quen với tính toán lưới và tính toán song song phân cụm. Shell script thực hiện cấu hình tự động được đưa ra nhằm đơn giản hóa vấn đề này, đối với cluster based PBS ta thực hiện thiết lập các thông số cho máy chủ PBS, tạo ra các hàng đợi định tuyến (routing queue) và các hàng đợi thực hiện công việc (execution queue) còn đối với grid based Globus ta thực hiện thiết lập các thông số để làm việc với cluster như shell thực hiện truy nhập, thông số kiểm tra hàng đợi, v.v... Có thể tổng kết các chức năng của shell script như sau:

Thiết lập các hàng đợi thực thi công việc (execution queue) với các mức độ ưu tiên khác nhau để thực hiện các công việc khác nhau tùy theo khoảng thời gian tương ứng.

Thiết lập một hàng đợi định tuyến (routing queue) dùng để kiểm tra các công việc rồi đưa công việc vào các hàng đợi thực thi tương ứng.

Thực hiện thiết lập các thông số cho PBS_Server để nó hoạt động hiệu quả.

Cấu hình để Grid based-on Globus hiểu tài nguyên nó sẽ thực hiện là một cluster.

9.3.4. Triển khai tại trung tâm HPCC

Sau khi nghiên cứu các nền tảng về mặt công nghệ của tính toán lưới và tính toán song song phân cụm, việc triển khai thử nghiệm được tiến hành bằng cách thiết lập các mạng con trên trung tâm HPCC, mỗi mạng con thiết lập sẽ được coi là một hệ thống tính toán song song phân cụm được đặt tại một nơi nào đó có sự cách xa trung tâm HPCC về mặt địa lý. Thực hiện kết nối các mạng con này vào lưới với mục đích thể hiện các tính chất của lưới.

Mục đích chia thành mạng con:

+ Thể hiện tính phân tán về mặt địa lý của các tài nguyên lưới.

+ Thể hiện tính đa dạng về tài nguyên tính toán, các tài nguyên tính toán có thể là máy chủ, máy tính cá nhân, cluster hay các hệ thống máy tính đa bộ xử lý đối xứng.

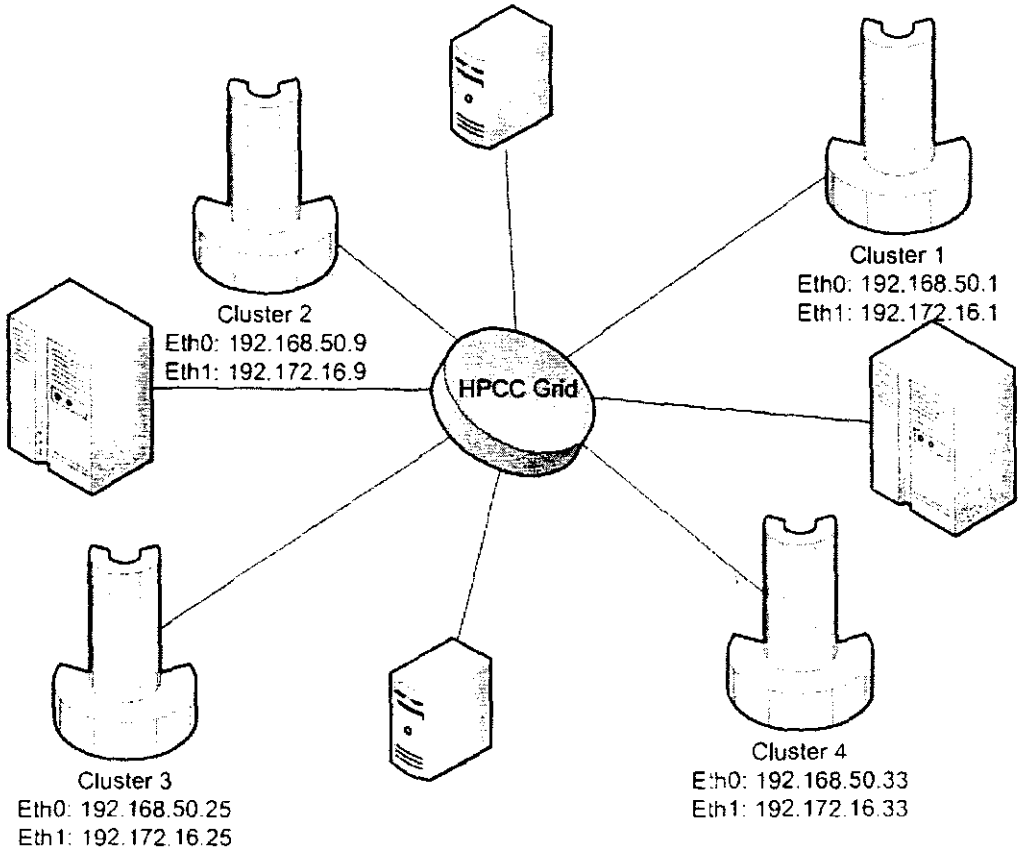
+ Thể hiện tính bảo mật khác nhau của các hệ thống tính toán phân cụm (các chính sách bảo mật của từng hệ thống là khác nhau)

+ Cho phép áp dụng các cấu hình khác nhau trên từng mạng con

Hệ thống máy tính ở đây đã được triển khai thành 4 cluster và các máy tính đơn để thể hiện một môi trường lưới giống với thực tế.

Cluster 1 gồm 6 máy tính được thiết lập lại địa chỉ IP từ 192.172.16.1 đến 192.172.16.6, trên hệ thống cluster này ta thực hiện cài đặt Globus Toolkits, GlobusPbsScheduler, PBS và MPICH. Hệ thống cluster này sử dụng máy địa chỉ 192.172.16.1 làm máy chủ, máy chủ này có một địa chỉ IP dùng để giao tiếp với bên ngoài là 192.168.50.1.

Cluster 2 cũng gồm 6 máy tính được thiết lập địa chỉ IP từ 192.172.16.9 đến 192.172.16.14, trên hệ thống cluster này ta thực hiện cài đặt Globus Toolkits, GlobusPbsScheduler, PBS tuy nhiên với thư viện truyền thông điệp thì ta dùng LAM-MPI. Trên cluster này ta chọn máy 192.172.16.9 làm máy chủ và thực cung cấp cho nó một giao diện để giao tiếp với bên ngoài là IP 192.168.50.9.



Hình 9-13: Triển khai lưới tại trung tâm HPC

Cluster 3 được thiết lập giống với cluster 1 về phần mềm, dải địa chỉ IP sử dụng cho cluster 3 là từ 192.172.16.25 đến 192.172.16.30. Trong cluster này ta chọn máy 192.172.16.25 làm máy chủ cung cấp cho nó thêm một giao diện để giao tiếp với bên ngoài là địa chỉ IP 192.168.50.25.

Cluster 4 được thiết lập giống với cluster 2 về mặt phần mềm, dải địa chỉ sử dụng trong cluster này là 192.172.16.33 đến 192.172.16.38. Trong cluster này ta chọn máy

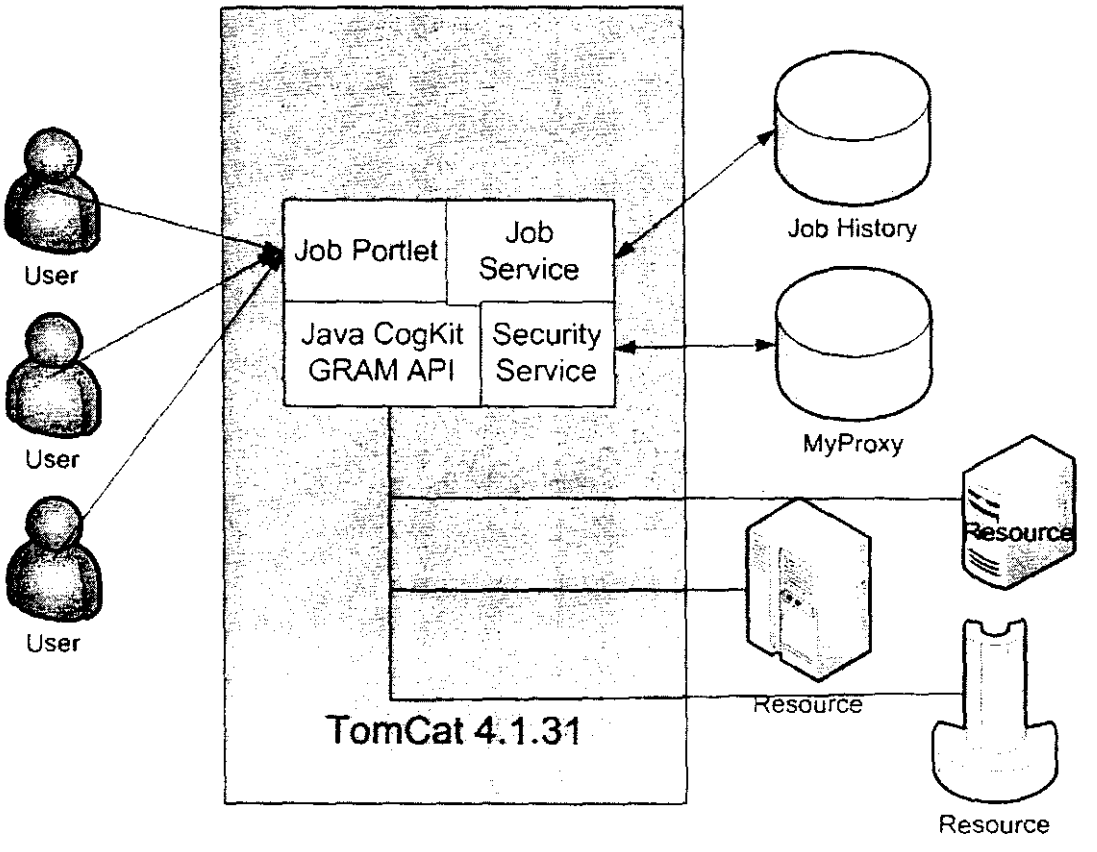
192.172.16.33 làm máy chủ và cung cấp cho nó một giao diện để giao tiếp với bên ngoài là IP 192.168.50.33.

Bên cạnh các cluster được thiết lập, hệ thống lưới tính toán trên trung tâm tính toán hiệu năng cao HPC còn bao gồm các máy tính đơn và máy chủ, điều này thể hiện rất rõ tính đa tài nguyên của một lưới tính toán.

9.4. Xây dựng mô-đun đệ trình công việc

9.4.1. Vị trí và vai trò của mô-đun đệ trình công việc

Bên cạnh dịch vụ khai phá dữ liệu Weka, hệ thống BKGrid2005 còn cung cấp dịch vụ tính toán hiệu năng cao, đây được coi là một hướng phát triển mang tính thời sự của BKGrid2005 bởi nó tận dụng được sức mạnh sẵn có của các cluster, kết hợp với cluster để thể hiện hết các đặc tính của lưới (đa tài nguyên, đa miền quản trị, đa người dùng).



Hình 9-14: Đệ trình công việc thông qua Job Portlet

Trong đó mô-đun đệ trình công việc là giao diện chính để người dùng đệ trình các công việc của mình lên lưới. Với vị trí đó, mô-đun đệ trình công việc có các nhiệm vụ sau:

Cung cấp giao diện Web thân thiện cho phép người dùng người dùng đệ trình công việc lên lưới.

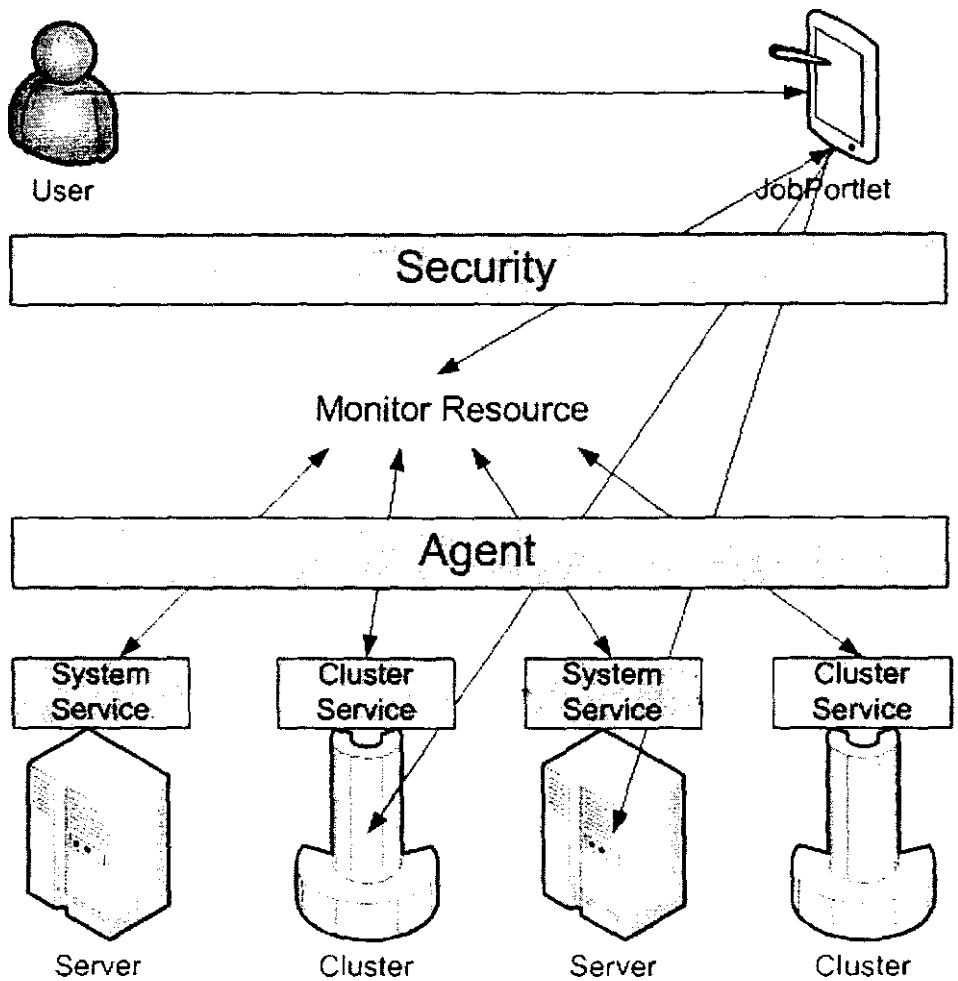
Chạy được các ứng dụng đơn và các ứng dụng song song lập trình theo thư viện truyền thông điệp MPI.

Quản lý tình trạng các công việc do người dùng đệ trình lên lưới.

Lưu lại kết quả trên máy chủ và trả lại kết quả cho người dùng khi có yêu cầu.

9.4.2. Tương tác với các thành phần hệ thống

Để có thể thực hiện tốt nhiệm vụ của mình, mô-đun đệ trình công việc phải tương tác với các thành phần hệ thống khác, thực hiện tốt các công việc cho người dùng. Sau đây sẽ trình bày sự tương tác đó:



Hình 9-15: Tương tác của Job Portlet với hệ thống

- Dịch vụ bảo mật

Để người dùng có thể thực hiện công việc trên lưới thì người dùng phải có được một giấy ủy quyền (proxy), mỗi khi đăng nhập vào lưới thông qua Portal thì người dùng phải xin một giấy mới vì thông thường giấy ủy quyền chỉ có hiệu lực trong một khoảng thời gian nhất định (12 giờ), mô-đun đệ trình công việc sẽ kiểm tra xem người dùng có giấy ủy quyền hợp lệ hay không, nếu không thì buộc người dùng phải xin giấy ủy quyền từ dịch vụ bảo mật, khi đã xác

nhận và tin tưởng người dùng thì môdul đệ trình công việc sẽ đại diện cho người dùng đi tìm tài nguyên phù hợp và thực hiện công việc.

- Dịch vụ môi giới tài nguyên

Công việc thực hiện trên lưới rất đa dạng từ các lệnh đơn giản hệ điều hành cung cấp cho đến những chương trình tính toán song song phức tạp do người dùng tự viết. Loại tài nguyên thực hiện công việc chính vì thế cũng rất đa dạng từ các máy tính đơn, máy tính đa bộ xử lý đối xứng cho đến các cluster. Đứng trước sự phức tạp đó mô-đun đệ trình công việc thực hiện tìm kiếm tài nguyên bằng dịch vụ môi giới tài nguyên (Broker), dịch vụ này sẽ nhận các thông tin về công việc như thời gian thực hiện, số nút thực hiện, ... rồi trả về thông tin về tài nguyên thích hợp nhất để thực hiện công việc.

- Dịch vụ thông tin tài nguyên

Mô-đun đệ trình công việc thông qua bộ môi giới tài nguyên tương tác với dịch vụ thông tin bởi dịch vụ thông tin có ảnh hưởng nhiều đến quá trình lựa chọn tài nguyên, vì vậy ảnh hưởng trực tiếp đến quá trình thực hiện công việc Thông tin tài nguyên bao gồm các thông tin như: hệ điều hành, bộ vi xử lý, số lượng RAM, ... những thông tin này đặc trưng cho một loại tài nguyên nhất định và được cung cấp bởi các dịch vụ thông tin như SystemService hay ClusterService.

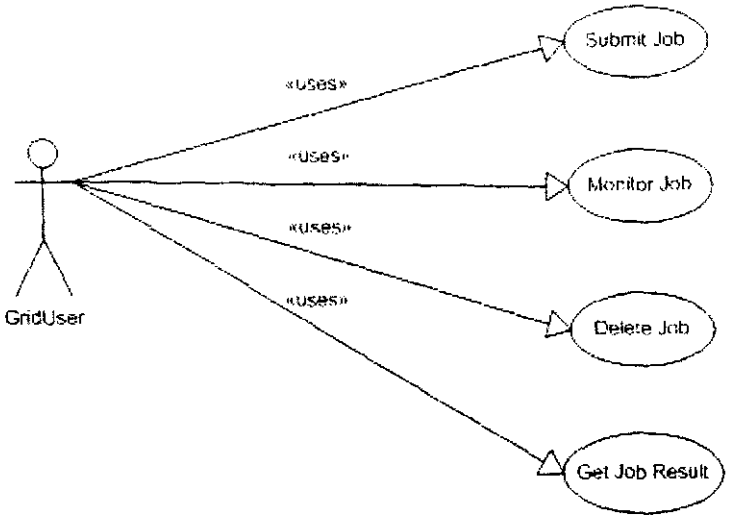
9.4.3. Thiết kế và xây dựng mô-đun đệ trình công việc

Mô-đun đệ trình công việc được xây dựng trên nền gridportal Gridsphere, chạy trên trình chứa Apache Tomcat, cho phép người dùng truy nhập ứng dụng dịch vụ từ xa. Người dùng muốn sử dụng dịch vụ phải thực hiện đăng ký với portal, sau khi đã trở thành một thành viên của lưới và thực hiện ủy quyền cho portal thực hiện (bằng giấy chứng nhận X509) thì người dùng có thể đệ trình những công việc lên trên lưới.

Portal thực hiện lưu giữ công việc mà người dùng đệ trình vào CSDL HSQL, điều này cho phép người dùng xem chi tiết về công việc (mã RSL, đầu ra công việc, tình trạng của công việc, môi trường thực hiện, ...) và đưa thông tin ra màn hình nếu được yêu cầu.

9.4.3.1. Các yêu cầu

Trong phần này ta sẽ thể hiện các chức năng của mô-đun đệ trình công việc thông qua ngôn ngữ mô hình hóa UML. Đứng từ góc độ người sử dụng thì mô-đun đệ trình công việc bao gồm các chức năng sau:



Hình 9-16: Các chức năng của mô-đun đệ trình công việc

Người dùng thực hiện đệ trình công việc lên lưới thông qua giao diện portlet với các thuộc tính của công việc như số nút thực hiện, thời gian dùng CPU, ... portlet sẽ thực hiện các công việc còn lại cho người dùng là tạo một file đặc tả công việc theo ngôn ngữ RSL, đi tìm kiếm tài nguyên, thực hiện công việc và nhận kết quả trả về.

Đối với những công việc đòi hỏi khối lượng tính toán lớn với thời gian thực hiện lâu (từ 1 ngày đến 1 tuần) người dùng được thông báo về tình trạng công việc thông qua chức năng theo dõi công việc. Các thông tin về công việc được lưu trong CSDL để thuận tiện cho các truy vấn về sau của người dùng.

Trong quá trình thực hiện một công việc người dùng có thể hủy bỏ công việc không thực hiện nữa thông qua chức năng xóa công việc.

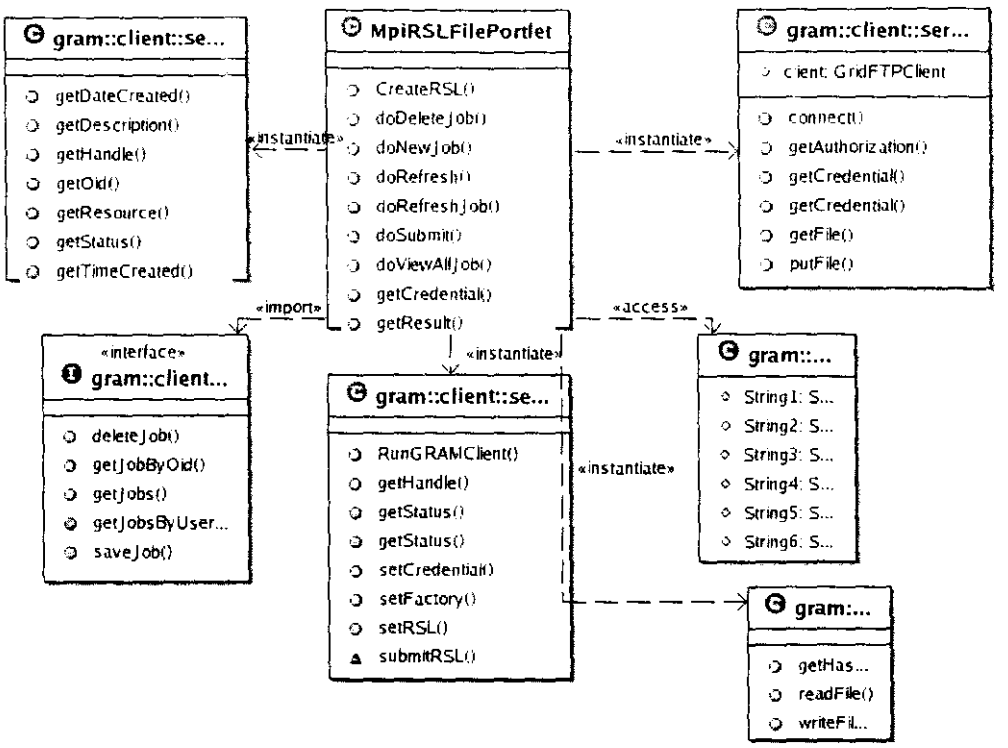
Portlet cho phép người dùng nhận kết quả trả về thông qua giao diện portlet, file kết quả đồng thời được lưu giữ trên thư mục của người dùng trên Server.

9.4.3.2. Thiết kế mô-đun đệ trình công việc

Mô-đun đệ trình công việc được xây dựng bằng công nghệ Portlet và Portlet Service, sử dụng GRAM API và JAVA CogKit cung cấp một giao diện thân thiện cho người sử dụng. Để có thể làm được điều đó chương trình phải sử dụng:

+ Các cơ chế bảo mật cung cấp dưới dạng Portlet Service.
+ Tương tác với bộ môi giới tài nguyên dưới dạng dịch vụ lưới.
+ Sử dụng các thông tin do dịch vụ thông tin cung cấp.
+ Sử dụng dịch vụ truyền file trong môi trường lưới GridFTP

Trên cơ sở đó, chương trình được xây dựng với các lớp sau:



Hình 9-17: Các lớp chính của mô-đun

Các lớp chính của mô-đun đệ trình công việc:

Lớp PortletClient: đây là lớp chính thực hiện việc đệ trình công việc cho các dịch vụ GRAM xử lý, công việc ở đây được thực hiện dưới dạng lô (batch processing) ngoài ra lớp này còn thực hiện theo dõi tình trạng công việc, hủy bỏ một công việc, trả về tính trạng của công việc tại một thời điểm nào đó, hay các thông tin khác về công việc.

Lớp GridFTP: đây là lớp thực hiện việc truyền file trong môi trường lưới, trong quá trình thực hiện công việc nếu có yêu cầu truyền file lên tài nguyên để thực hiện hay nhận kết quả từ tài nguyên về Portal Server thì lớp này được gọi để thực hiện.

Lớp SingleRSLFile: đây là lớp cung cấp các thông tin về công việc đơn, tạo ra các file RSL công việc đơn dùng trong việc đệ trình lên tài nguyên ở xa.

Lớp MpiRSLFile: đây là lớp cung cấp các thông tin về các công việc tính toán song song, tạo ra các file công việc song song đệ trình lên các tài nguyên là cluster.

Lớp MpiRSLFilePortlet: lớp này là lớp chính thực hiện tất cả các thao tác trên portlet, bao gồm tất cả các chức năng của các lớp khác.

Lớp JobSubmissionService: lớp này được viết dưới dạng Portlet Service thực hiện các chức năng thao tác với CSDL như cất các thông tin về công việc trong CSDL, cập nhật tình trạng một công việc hay xóa bỏ một công việc.

Lớp Job: Chứa các đặc tả công việc được xử lý, tình trạng công việc và các thông tin khác có liên quan với công việc.

9.4.4. Chi tiết xây dựng mô-đun đệ trình công việc

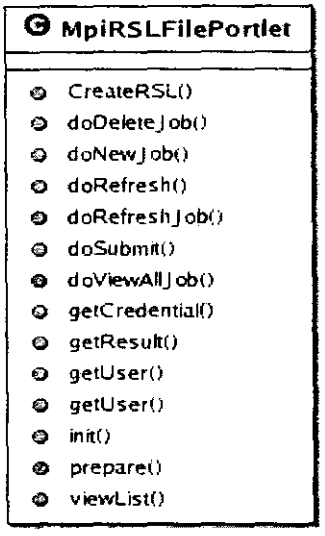
Trong phần mô-đun đệ trình công việc ta xây dựng hai mô-đun độc lập với nhau đó là:

Mô-đun đệ trình công việc đơn

Mô-đun đệ trình công việc song song

Hai mô-đun này có những điểm tương đối giống nhau vì vậy ở đây đồ án xin được trình bày chi tiết về mô-đun đệ trình công việc song song.

- Lớp MpiRSLFilePortlet



Init(): Khi người dùng gọi portlet lần đầu tiên trong phiên làm việc của họ tiên thì phương thức này được gọi, nó thực hiện khởi tạo các dịch vụ phục vụ cho phiên làm việc như dịch vụ quản lý người dùng, dịch vụ quản lý giấy ủy nhiệm, thực hiện khởi tạo một client đệ trình công việc PortletClient dùng trong việc liên lạc với các dịch vụ tầng dưới trong việc đệ trình công việc.

doSubmit(): Sau khi kiểm tra các thông số về công việc mà người dùng nhập vào thì portlet thực hiện thiết lập các thông số cần thiết cho công việc thông qua phương thức CreateRSL() gán các thông số này cho công việc Job cùng với thông tin về giấy ủy nhiệm. Giấy ủy nhiệm phải được kiểm tra bởi dịch vụ cung cấp giấy ủy nhiệm trên Portal.

doDeleteJob(): phương thức này thực gọi đến handle thực hiện công việc, nếu công việc chưa hoàn thành thì nó thực hiện loại bỏ công việc đồng thời loại bỏ thông tin về công việc trong CSDL.

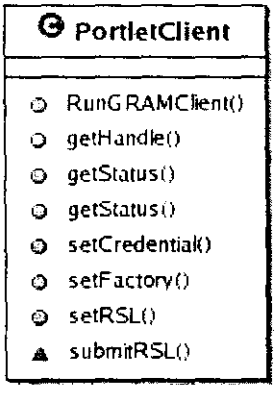
doNewJob(): phương thức này chuyển người dùng đến trang thực hiện công việc để thực hiện một công việc mới.

doRefresh(): Thực hiện kiểm tra tình trạng tất cả công việc, trong trường hợp công việc chưa hoàn thành thì thực hiện cập nhật trạng thái công việc.

doRefreshJob(): Kiểm tra một công việc cụ thể đang được thực hiện như thế nào, trạng thái của nó, thông tin chi tiết cho người dùng về công việc.

doJobResult(): phương thức này cũng thực hiện kiểm tra tình trạng của công việc đang được thực hiện, trong trường hợp công việc đã thực hiện xong "DONE" thì đưa ra kết quả cho người dùng.

- Lớp PortletClient



RunGRAMCLient(): Đây là phương thức chính của đối tượng, mỗi một công việc sau khi được thiết lập các thông số từ portlet đệ trình công việc song song sẽ được chạy bằng phương thức này.

SetCredential(): Thiết lập giấy uỷ quyền dùng trong khi thực hiện công việc.

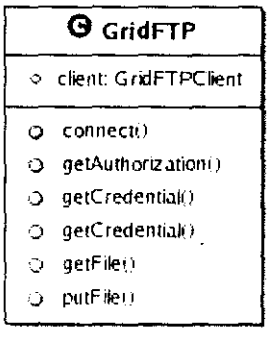
SetFactory(): Địa chỉ dịch vụ dùng trong việc thực hiện công việc.

SetRSL(): Nhận các thông tin về công việc từ portlet đệ trình công việc, thực hiện gán cho công việc.

getHandle(): Mỗi một công việc được đặc trưng bởi một handle, handle này dùng trong việc quản lý trạng thái công việc, phương thức thực hiện lấy handle của công việc sau khi công việc đã chạy ở chế độ lô (batch).

getStatus(): phương thức này thực hiện trả lại tình trạng của công việc ở một thời điểm nào đó khi người dùng muốn biết tình trạng công việc. Phương thức này có thể không có đối số hoặc có đối sỗ là handle của công việc.

- Lớp GridFTP



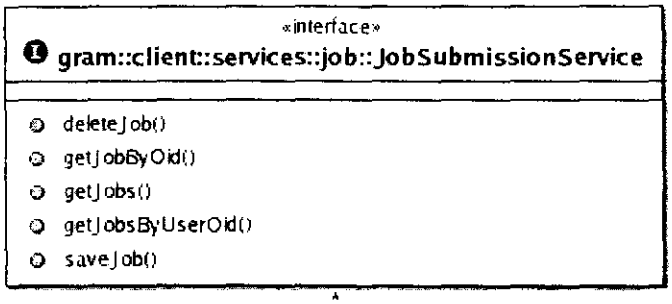
Lớp GridFTP được sử dụng khá nhiều trong quá trình đệ trình công việc lên lưới, lớp này thực hiện truyền file thực thi đến các tài nguyên để thực hiện sau đó lại thực hiện truyền kết quả về.

Connect(): Thực hiện kết nối đến máy chủ để thực hiện một thao tác với file

getFile(): truyền file từ phía máy chủ về máy khách

putFile(): đặt một file lên trên máy chủ

- Lớp JobSubmissionService



Lớp này là một lớp giao diện (interface) được thực thi bởi lớp JobSubmissionServiceImpl, lớp này chủ yếu thực hiện các thao tác với CSDL, trong lớp bao gồm các phương thức sau:

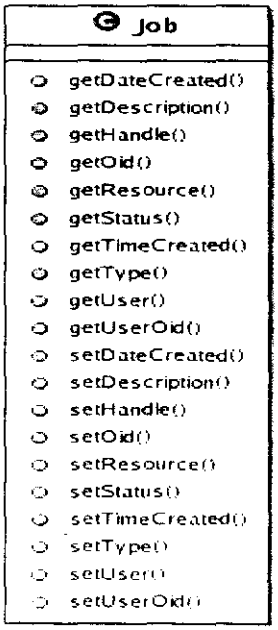
SaveJob(): thực hiện lưu giữ thông tin về công việc trong CSDL

deleteJob(): Xóa bỏ công việc

getJobByOid(): Lấy thông tin về một công việc ra để sử dụng

getJobByUserOid() Lấy tất cả các công việc được thực hiện bởi một người dùng.

- Lớp Job



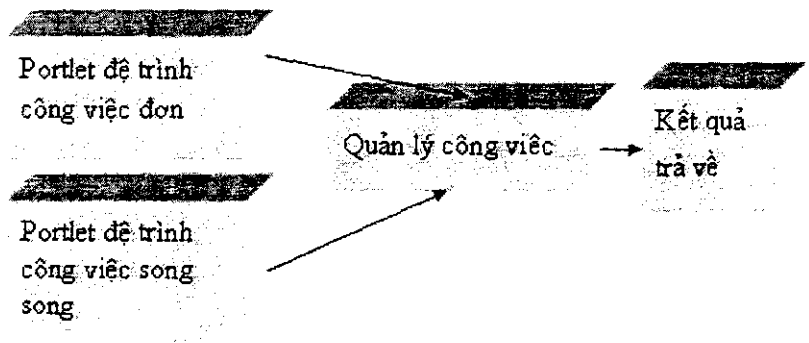
Lớp Job bao gồm tất cả các thông tin về công việc như: định danh công việc, miêu tả công việc, người thực hiện, ngày thực hiện, ... và một tập các phương thức dùng để thiết lập thông tin lên công việc cũng như lấy các thông tin ra từ một công việc.

9.4.5. Triển khai mô-đun

Mô-đun JobSubmission được tích hợp vào hệ thống BKGrid2005 dưới dạng 2 portlet trong toàn bộ hệ thống.

+ Portlet đệ trình công việc đơn: thực hiện đệ trình các công việc đơn lên lưới.

+ Portlet đệ trình công việc song song: thực hiện chạy một file thực thi viết theo thư viện truyền thông điệp MPI



Hình 9-18: Quản lý công việc

Các công việc được đệ trình bằng hai portlet đệ trình công việc sẽ được chuyển sang mô-đun quản lý công việc, mô-đun quản lý công việc thực hiện việc theo dõi và trả về kết quả đối với những công việc đã hoàn thành (DONE).

Quá trình thực hiện công việc có thể diễn ra trong một khoảng thời gian dài nên các kết quả trả về của công việc được lưu trên thư mục của người dùng trên máy chủ, người dùng có thể đăng nhập vào portal và lấy kết quả về bất cứ lúc nào.

Danh mục các từ viết tắt

Từ	Giải thích
ACL	Agent Communication Language
AID	AID - Agent Identification
AMS	Agent Management System
AP	Agent Platform
API	Application Program Interface
ASP	Application Service Provider
CA	Certificate Authority
Cog Kit	Commodity Grid Toolkit
CORBA	Common Object Request Broker Architecture
DF	Directory Facilitator
DMTF	Distributed Management Task Force
DN	Distinguished Name
DS	Digital Signature
DUROC	Updated Request Online Coallocator
EJB	Enterprise Java Bean
FIPA	Foundation for Inteligent Physical Agent
GGF	Global Grid Forum
GIIS	Grid Index Information Servers
GIS	Grid Information Service
GMD	Grid Market Directory
GRACE	Grid Architecture for Computation Economy
GRAM	Grid Resource Access and Management

Từ	Giải thích
GridFTP	Grid File Tranfer Protocol
GRIM	Grid Resource Identity Mapper
GRIP	Grid Resource Information Protocol
GRIS	Grid Resource Information Service
GRRP	Grid Resource Registration Protocol
GSI	Grid Security Infrastructure
GSS	Generic Security Service
GT	Globus Toolkit
GWSDL	Grid Web Service Description Language
HAP	Home Agent Platform
HTTP	Hyper Text Markup Language
IETF	Internet Engineering Task Force
JADE	Java Agent DEvelopment Framework
KQML	Knowledge Query and Manipulation Language
LDAP	Lightweight Directory Access Protocol
LMJFS	Local Managed Job Factory Service
MAS	Multi Agent System
MJS	Managed Job Service
MMJFS	Master Managed Job Factory Service
MPI	Message Passing Interface
MTS	Message Transport Service
OASIS	Organization for the Advancement of Structured Information Standards
OGSA	Open Grid Service Architecture
OGSI	Open Grid Service Infrastructure

Từ	Giải thích
PGP	Partial Global Planning
PK	PK - Public Key
PKI	Public Key Infracstructure
QoS	Quality of Service
RBAC	Roll-based Acess Control
RMI	Remote Method Invoke
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP/IP	Transmision Control Protocol/Internet Protocol
TFE	Task Farming Engine
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
VKR	Virtual Knowledge Base
VO	Virtual Organization
W3C	World Wide Web Consortium
WSDD	Web Service Deployment Descriptor
WSDL	Web Service Description Language
XML	eXtensible Markup Language

TÀI LIỆU THAM KHẢO

[1] Ahmar Abbas. *Grid Computing: A Practical Guide to Technology and Applications*

[2] Albeaus Bayucan, Robert L. Henderson, James Patton Jones, Casimir Lesiak, Bhroam Mann, Bill Nitzberg, Tom Proett. *Portable Batch System Administrator Guide*

[3] Amotz Bar-Noy. *Sum multicoloring of Graphs.*

[4] Amy Apon. *Lecture for Cluster and Grid Computing*. University of Arkansas, 2004

[5] Andreas M. Antonopoulos. *Grid Computing: What is it and Why you should care. 2004*

[6] Bart Jacob, Luis Ferreira, Nobert Bieberstein, Candice Gilzean, Jean-Yves, Girard, Roman Strachowski, Seong (Steve) Yu. *Enabling Application for Grid Computing with Globus*. IBM RedBooks, 2003.

[7] Ben Clifford. *Monitoring and Discovery, USC Grid Computing course.* USC/Information Sciences Institute, 2003.

[8] Borja Sotomayor. *The Globus Toolkit 3 Programmer's Tutorial*

[9] D. Benaech, T. Desprats. *A KQML-CORBA based Architecture for Intelligent Agents Communication in Cooperative Service and Network Management; part 4 - KQML as a communication language.*

http://www.ryerson.ca/~dgrimsha/courses/cps720/kqmlIntro.html

[10] Đào Trần Minh. *Xây dựng bộ lập lịch hướng kinh tế cho BK Grid*

[11] Đặng Văn Đức. *Ngôn ngữ mô hình hóa UML*

[12] Đoàn Văn Ban. *Lập trình hướng đối tượng với Java*

[13] Fu Guangyu. *Learning Material of Grid Service*

[14] Hoàng Thế Hưng. *Xây dựng module theo dõi và phát hiện lỗi trong BK Grid.*

[15] HPCC. *Tổng quan về tính toán song song*

[16] Hyacinth Nwana, Lindon Lee, Nick Jennings. *Coordination in Software System*

[17] Ian Foster. *What is the Grid? A Three Point Checklist*

[18] Ian Foster, Carl Kesselman, Steven Tuecke. *The Anatomy of the Grid*

[19] Ian Foster, Carl Kesselman, Jeffrey M.Nick, Steven Tuecke. *The Phisiology of the Grid*

[20] Ian Foster, Karl Kesselman, Gene Tsudik, Steven Tuecke. *A Security Architecture for Computational Grid*

[21] Ian Foster, Nicholas R. Jennings, Carl Kesselman. *Brain Meets Brawn: Why Grid and Agents Need Each Other.* Proceeding of the 3rd International Conference on Autonomous Agents and Multi – Agent Systems, New York, USA, 2004.

[22] IBM Red Books. *Introduction to Grid Computing with Globus Toolkit*

[23] IBM Red Books. *Globus Toolkit 3.0 Quick Start*

[24] Isaac Chao, Ramon Sanguesa, Oscar Oardaiz. *Grid Resource Management using Software Agents: Grid Resource Management using Software Agents.* Ercim News No 59 Oct 2004.

[25] Jarek Nabrzyski, Jennifer M.Schopf. *Grid Resource Management*

[26] Jean Luc Lepesant, Sebastien Fibra. Submit jobs to grid resources using OGCE

[27] Jennifer M. Schopf, Mathematics and Computer Science Division, Argonne National Laboratory. *Ten actions when grid scheduling*

[28] Joshy Joseph, Craig Fellenstein. *Grid Computing.* Prentice Hall PTR

[29] Kai Reimers. *Automating Coordination Mechanism*

[30] KISTI Supercomputing Center Grid Technology Research Department Eunsung Kim. *GT 3 Base Services Inside- Information Services*

[31] MARK BAKER, RAJKUMAR BUYYA. *Cluster Computing: The Commodity Supercomputing*

[32] Nataraj Nagaratnam, Philippe Janson, John Dayka, Anthony Nadalin Frank Siebenlist, Von Welch, Ian Foster, Steve Tuecke. *The Security Architecture for Open Grid Service*

[33] Nguyễn Phương Lan, Hoàng Đức Hải. *Java lập trình mạng*

[34] Ola Redell. *Global Scheduling in Distributed Real-Time Computer Systems - An Automatic Control Perspective.*

[35] Otto Sievert. *A gentle introduction to Globus.*

[36] Paul Hyde. *Java Thread Programming*

[37] Rajkumar Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing.* Doctor thesis, 12 Apr 2002 Monash university Melbourne Australia.

[38] Rinat Khoussainov, Xin Zuo and Nicholas Kushmerick. *Grid-enabled Weka: A Toolkit for Machine Learning on the Grid.* Ercim News No 59 Oct 2004.

[39] Robert Tolksdorf . *Models of Coordination*

[40] Rui Min, Muthucumara Maheswaran. *Scheduling Advance Reservations with priorities in Grid Computing Systems*

[41] Srikumar Venugopal, Rajkumar Buyya, Lyle Winton. *A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids*

[42] Thomas Sandholm, Jarek Gawor. *Globus Toolkit 3 core – A Grid Service Container Framework*. 2003.

[43] University of Chicago. *Security for Grid Service*

[44] Viktors Berstis. *Fundamentals of Grid Computing*

[45] Vladimir Silva. Set up a grid job scheduler with the Globus Toolkit 3.2.x

[46] Vladimir Silva. The Master Managed Job Factory Service (MMJFS)

[47] Von Welch, Software Architect, Globus Project. *Globus Toolkit Firewall Requirements*

[48] William A. Ward, Jr., Carrie L. Mahood, John E. West Computer Sciences Corporation. *Scheduling Jobs on Parallel Systems Using a Relaxed Backfill Strategy.*

[49] Xiaoshan He, Xian-He Sun, Gregor Von Laszewski. *A QoS Guided Scheduling Algorithm for Grid Computing*

[50] Yannis Labrou, Tim Finin. *A Proposal for a new KQML Specification.* February 3, 1997

[51]. *Agent Communication Languages.*

http://www.ida.liu.se/labs/iislab/courses/Agents/paper/chapter3.html

[52]. *GT3 Grid Security Infrastructure Overview.*

[53]. *Grid Computing.* Prentice Hall, 2003.

[54]. *Jade Home Page.* http://jade.cselt.it/

[55] Website Dr. Rajkumar Buyya: http://www.buyya.com

[56] The GRIDS Lab and the Gridbus Project http://www.gridbus.org

[57] Website Globus Toolkits http://www.globus.org

[58] Website GridSphere http://www.gridsphere.org

[59] http://www-unix.globus.org/mail_archive/discuss/2004/02/threads.html

[60] http://www.webopedia.com/TERM

[61] http://www.cs.waikato.ac.nz/ml/weka/

[62] www.ibm.com/redbook

[63] www.ggf.org

[64] www.w3c.org/webservice

[65] www.computer.org

[66] www.acm.org.