BÀI TOÁN DÂY RUNG TRÊN MÔI TRƯỜNG SONG SONG

Nguyễn Mậu Hân
Trường Đại học Khoa học, Đại học Huế
Trần Anh Nam
Trường CĐSP Kỹ thuật Gia Lai

TÓM TẮT

Bài toán dây rung được biết đến một cách rộng rãi trong khoa học kỹ thuật, đặc biệt trong lĩnh vực xử lý âm thanh. Bài báo này giới thiệu một phương pháp để song song hóa giải thuật bài toán dây rung. Bằng cách sử dụng chuẩn lập trình song song LAM/MPI trên cụm máy tính sử dụng hệ điều hành Linux, giải thuật song song cho bài toán dây rung đã được cài đặt thành công và có ý nghĩa thực tế.

I. Giới thiệu

Hầu hết các nhạc cụ thường sử dụng dây rung để tạo nên các âm thanh trong âm nhạc như đàn guitar, đàn cello và piano. Bất kỳ ai đã chơi các loại nhạc cụ này đều biết rằng sự thay đổi độ căng của dây sẽ làm thay đổi tần số âm thanh của dây. Ở đây, chúng tôi xem xét một dây rung bằng cách giữ cố định ở hai đầu và mô tả hành vi của nó. Sau đó, sử dụng công cụ toán học để mô tả các hiện tượng sóng. Trong trường hợp tổng quát, sự chuyển động sóng được biểu diễn bởi một hàm theo thời gian. Xét hàm u(x,t) biểu diễn biên độ của dây rung giữa hai điểm, nếu biên độ này là nhỏ thì nó được xác định bởi phương trình đạo hàm riêng dạng hyperbolic thể hiện bởi phương trình sóng [1]:

$$\frac{c^2\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial^2 t} \qquad (1.1)$$

Trong phần 2, chúng tôi trình bày phương pháp giải bài toán dây rung bằng phương pháp tính dựa vào xấp xỉ tỉ sai phân của đạo hàm riêng cấp 2. Sau đó, chúng tôi áp dụng phương pháp luận của Ian Foster để thiết kế giải thuật song song cho bài toán trong phần 3. Trong phần 4, chúng tôi thực hiện đánh giá độ phức tạp của giải thuật song song. Kết quả thực nghiệm được trình bày trong phần 5 và cuối cùng là kết luận của bài báo được trình bày trong phần 6.

II. Phương pháp giải bài toán dây rung

Để có thể giải bài toán dây rung bằng phương pháp tính, chúng tôi sử dụng phương pháp sai phân hữu hạn biến đổi phương trình đạo hàm riêng (1.1) thành phương trình sai phân nhờ xấp xỉ tỉ sai phân của đạo hàm riêng cấp hai:

$$\frac{\partial^2 u(x,t)}{\partial^2 x} \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2} \qquad (2.1)$$

$$\frac{\partial^2 u(x,t)}{\partial^2 t} \approx \frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{(\Delta t)^2} \qquad (2.2)$$

Thay (2.1) và (2.2) vào (1.1), ta có:

$$\frac{c^2 u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2} = \frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{(\Delta t)^2} \quad \text{hay}$$

$$L^2[u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)] = u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t) \quad (2.3)$$

Trong đó, $L = \frac{c\Delta t}{\Delta x}$.

Chúng ta sẽ xây dựng lưới tính toán gồm các điểm lưới $x_i = (i-1)\Delta x$, $t_j = (j-1)\Delta t$. Ta có: $u(x_i, t_j) = u[(i-1)\Delta x, (j-1)\Delta t]$, $i, j = 1, 2, 3, \ldots$. Nên có thể viết gọn (2.3) lại như sau:

$$L^2(u_{i+1,j} - 2u_{i,i} + u_{i-1,j}) = u_{i,j+1} - u_{i,j} + u_{i,j-1} \qquad (2.4)$$

hay $$u_{i,j+1} = L^2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + 2u_{i,j} - u_{i,j-1} \qquad (2.5)$$

III. Thiết kế giải thuật song song

Khảo sát (2.5) ta thấy phần tử $u_{i,j+1}$ phụ thuộc vào các phần tử $u_{i,j}$, $u_{i,j-1}$, $u_{i+1,j}$ và $u_{i-1,j}$. Do đó, áp dụng phương pháp luận của Foster để thiết kế giải thuật song song cho bài toán dây rung [1]. Phương pháp này bao gồm bốn bước sau:

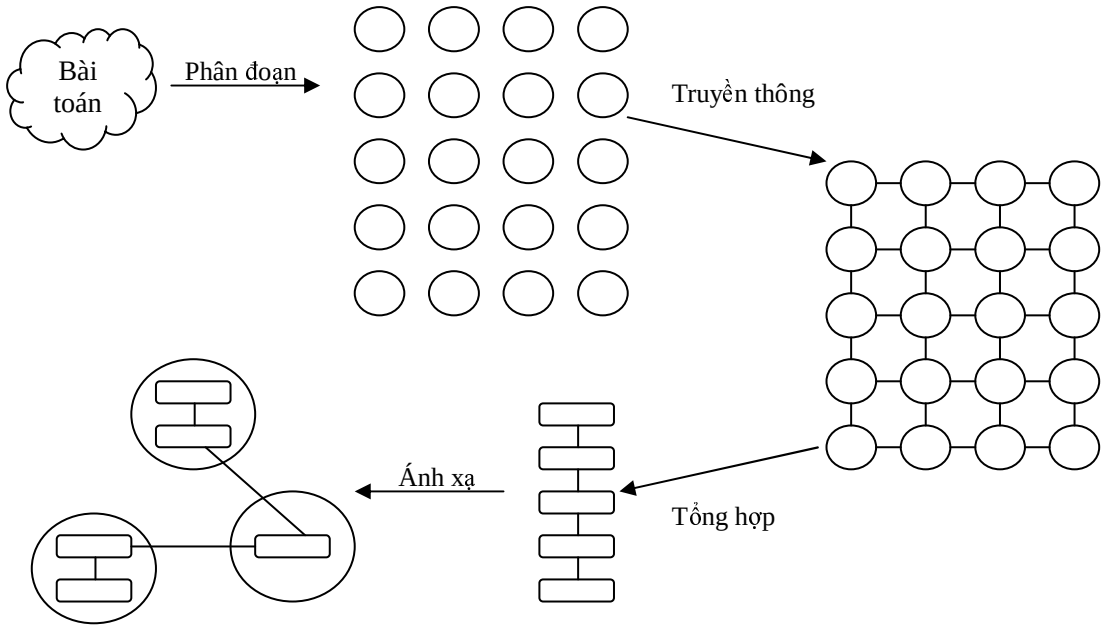
Phân đoạn

Truyền thông

Tổng hợp

Ánh xạ

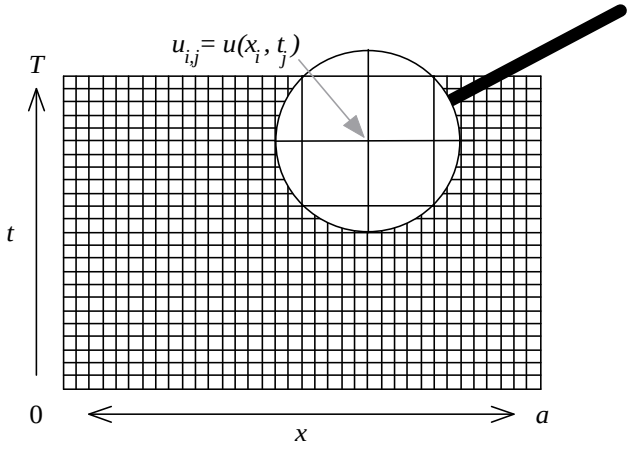
Phân đoạn là quá trình chia việc tính toán và dữ liệu thành các công việc nhỏ hơn, chấp nhận sự tính toán dư thừa và dữ liệu được lưu trữ tối ưu nhất. Mô hình truyền thông giữa các tiến trình được thực hiện trong bước thứ hai của phương pháp luận, nó có khả năng cân bằng tải giữa các tiến trình, các tiến trình có thể trao đổi dữ liệu cho nhau trong quá trình tính toán. Quá trình tổng hợp được tiến hành ở bước thứ ba, nó thực hiện nhóm các công việc nhỏ thành các công việc lớn hơn. Bước cuối cùng là ánh xạ, đó là quá trình gán các tiến trình cho các bộ xử lý. Mục đích của việc ánh xạ là cực đại hóa khả năng của bộ xử lý và cực tiểu hóa sự truyền thông giữa các bộ xử lý. Chúng ta sẽ mô tả chi tiết bốn giai đoạn này như sau.



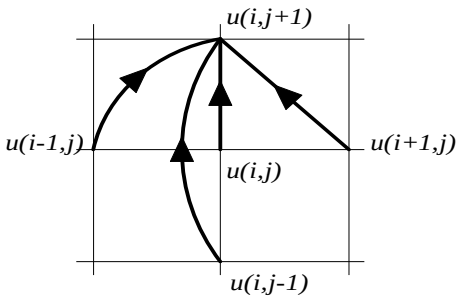
Hình 1. *Thiết kế giải thuật song song theo Ian Foster*

3.1. Phân đoạn

Để song song hóa chúng ta cần chọn loại dữ liệu để phân tách. Sự phân tách dữ liệu được chọn thông qua sự phân tách hàm. Chúng ta cần sử dụng cùng một hàm trên tất cả các tiến trình nhưng với những dữ liệu khác nhau. Theo (2.5), $u_{i,j+1}$ phụ thuộc vào $u_{i,j}$ và $u_{i,j-1}$. Mặt khác, $u_{i,j+1}$ yêu cầu các giá trị $u_{i+1,j}$ và $u_{i-1,j}$ phải được tính toán trước. Để xây dựng lưới tính toán cho bài toán, chúng ta tiến hành rời rạc hóa miền không gian và thời gian thành các khoảng bởi ma trận hai chiều như Hình 2. Do đó, việc tính giá trị của mỗi phần tử phụ thuộc vào các phần tử lân cận được minh họa trong Hình 3.



Hình 2. *Lưới tính toán là ma trận hai chiều*



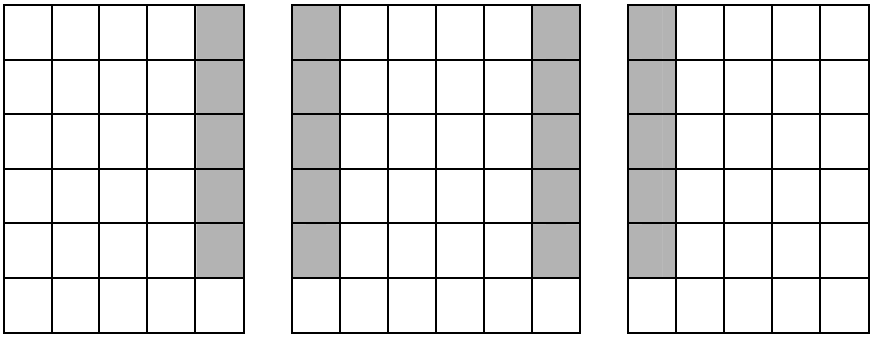
Hình 3. *Sự phụ thuộc dữ liệu của phần tử cần tính*

3.2. Truyền thông

Sự truyền thông giữa các điểm lưới được chỉ ra trong Hình 3, nó được suy ra từ công thức (2.5). Giá trị của phần tử cần tính $u_{i,j+1}$ trong lưới tính toán phụ thuộc vào giá trị của các phần tử $u_{i-1,j}$, $u_{i,j}$, $u_{i+1,j}$, $u_{i,j-1}$. Để tính giá trị của các phần tử trên dòng $j+1$ chúng ta cần biết giá trị của các phần tử trên dòng j và $j-1$. Hai dòng này được khởi tạo giá trị nhờ vào điều kiện đầu của bài toán mà không tốn bất kỳ chi phí truyền thông nào.

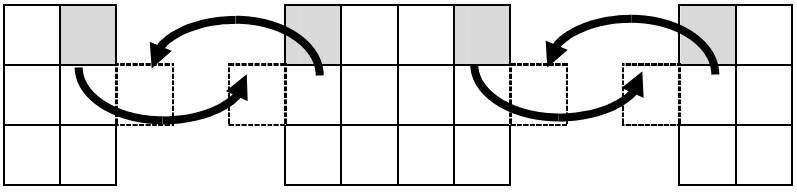
3.3. Tổng hợp

Tổng hợp là quá trình nhóm các công việc lại thành các khối lớn hơn để loại bỏ sự truyền thông giữa các tiến trình. Mỗi tiến trình là một khối cột có số phần tử bằng nhau.



Hình 4. *Cách bố trí các điểm ảo*

Để dễ dàng hơn trong việc lập trình song song chúng tôi sử dụng các điểm ảo [1], đó chính là các ô nhớ được sử dụng để lưu trữ dữ liệu của các tiến trình bên cạnh. Điều này cho phép chương trình song song cập nhật giá trị của các phần tử trên cùng một bước lặp.



Hình 5. *Sự truyền thông giữa các tiến trình*

3.4 Ánh xạ

Sau khi tổng hợp, chúng ta tiến hành gán mỗi tiến trình cho một bộ xử lý, việc ánh xạ cố gắng cực đại hóa khả năng của bộ xử lý và cực tiểu hóa sự truyền thông giữa các bộ xử lý.

IV. Độ phức tạp của giải thuật song song

Đối với giải thuật song song, nếu số phần tử trên mỗi tiến trình là như nhau thì độ phức tạp của nó là O(n/p) trên mỗi bước lặp (trong đó n là số điểm trên dây rung, p là số tiến trình). Thời gian truyền thông đối với việc gửi và nhận là O(1). Do đó, toàn bộ thời gian truyền thông qua giải thuật song song là O(p).

V. Kết quả thực nghiệm

Chúng tôi chọn chuẩn lập trình LAM/MPI trên môi trường Linux để thực nghiệm bài toán đặt ra vì những ưu điểm của nó so với các chuẩn lập trình song song khác như PVM, MPICH[2],… Sau đó, thực hiện cài đặt bài toán với ngôn ngữ lập trình C kết hợp với các thư viện của MPI [3], [4]. Bài toán được thực hiện trên cụm máy tính Linux với cấu hình của các nút như sau:

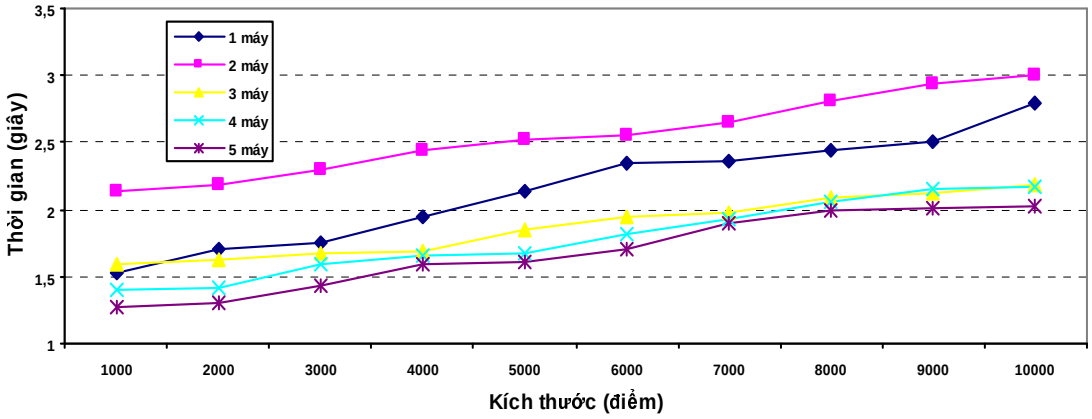
Mainboard	**Memory**	**Hard Disk**	**Ethernet card**
ASUS P4BGL-MX wLAN Celeron 1.8GHz	256 MB DDR PC226	40 GB HDD	Realtek RTL 8139 Family PCI Fast Ethernet 100Mbps

Cấu hình của các nút tham gia tính toán

Kết quả đo được thể hiện trong Bảng 1 và Bảng 2.

Bảng 1. *Thời gian đo được sau 2000 bước, 10 tiến trình và kích thước bài toán từ 1000 điểm đến 10000 điểm (thời gian được tính bằng giây)*

Điểm / Nút	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
1	01.53	01.71	01.75	01.94	02.13	02.35	02.37	02.44	02.50	02.79
2	02.13	02.19	02.30	02.44	02.53	02.56	02.65	02.81	02.94	03.00
3	01.60	01.63	01.67	01.69	01.85	01.94	01.97	02.09	02.12	02.19
4	01.40	01.42	01.59	01.66	01.67	01.82	01.93	02.06	02.15	02.17
5	01.28	01.30	01.44	01.59	01.61	01.70	01.90	02.00	02.01	02.03

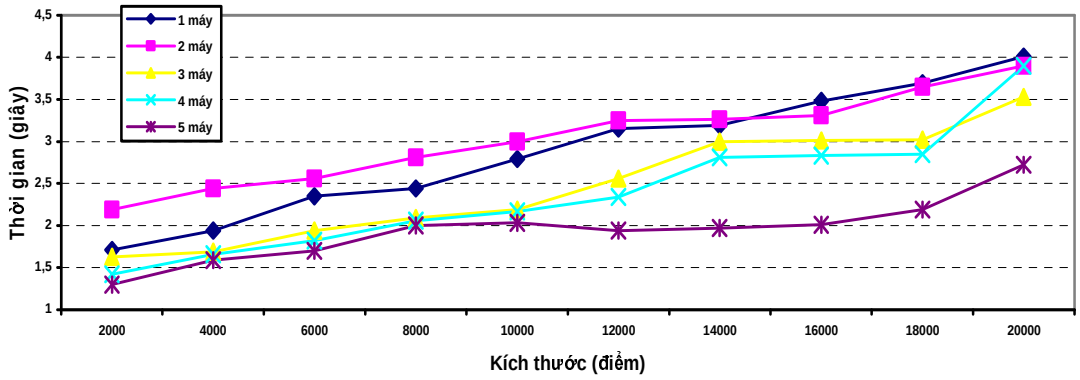


Hình 6. *Biểu đồ so sánh thời gian thực hiện kích thước từ 1000 đến 10000*

Bảng 2. *Thời gian đo được sau 2000 bước, 10 tiến trình và kích thước bài toán từ 2000 điểm đến 20000 điểm (thời gian được tính bằng giây)*

Điểm / Nút	2000	4000	6000	8000	10000	12000	14000	16000	18000	20000
1	01.71	01.94	02.35	02.44	02.79	03.15	03.19	03.48	03.69	04.01
2	02.19	02.44	02.56	02.81	03.00	03.25	03.26	03.31	03.65	03.90
3	01.63	01.69	01.94	02.09	02.19	02.56	03.00	03.01	03.02	03.53
4	01.42	01.66	01.82	02.06	02.17	02.34	02.81	02.83	02.85	03.90
5	01.30	01.59	01.70	02.00	02.03	01.94	01.97	02.01	02.19	02.72

Hình 6 cho thấy cùng một kích thước của bài toán (cùng số điểm, số tiến trình và số bước) nhưng thời gian thực hiện trên 1 nút nhanh hơn trên 2 nút. Sở dĩ có điều này là do kích thước của bài toán không đủ lớn nên việc thực hiện trên 2 máy ngoài chi phí tính toán còn có chi phí truyền thông giữa các tiến trình qua LAN và sự tăng lên về kích thước dữ liệu do sự mã hóa dữ liệu truyền thông theo phương pháp RSA. Tuy nhiên, với kích thước bài toán đủ lớn thì tốc độ tính toán sẽ tăng lên đáng kể khi số nút tăng như thể hiện trên biểu đồ ở Hình 7.



Hình 7. *Biểu đồ so sánh thời gian thực hiện kích thước từ 2000 đến 20000*

VI. Kết luận

Trong bài báo này, chúng tôi đã giới thiệu phương pháp giải bài toán dây rung nhờ xấp xỉ tỉ sai phân của đạo hàm riêng cấp 2 để xác định biên độ rung của dây. Sau đó chúng tôi sử luận phương pháp luận của Ian Foster để thiết kế giải thuật song song, thực hiện đánh giá độ phức tạp của thuật toán. Cuối cùng, cài đặt thuật toán dựa trên cụm máy tính Linux và đánh giá, so sánh các kết quả đo được.

TÀI LIỆU THAM KHẢO

1. Michael J. Quinn, *Parallel programming in C with MPI and OpenMP*, Mc Graw Hill, 2004.
2. P. H Carns, W. B. Ligon III, S. P. McMillan, R. B. Ross, *An evaluation of message passing implementations on Beowulf workstations*, Parallel Architecture Research Lab, 2001.
3. P. K. Jimack, N. Touheed, *Developing parallel finite element software using MPI*, University of Leeds, 2000.
4. P. K. Jimack, N. Touheed, *An introduction to MPI for computational mechanics*, University of Leeds, 1999.

VIBRATING STRING PROBLEM ON PARALLIZATION PROCESSING

Nguyen Mau Han
College of Sciences, Hue University
Tran Anh Nam
Gia Lai College of Technology Pedagogy

SUMMARY

Vibrating string problem is widely regconised in science and technology, specially in acoustic field. This paper suggests a method to parallise the vibrating string problem. By using LAM/MPI parallel programing standard in cluster computers we have sucessfully built parallel algorithm of vibrating string problem.