**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**BÀI TẬP LỚN**

**MÔN HỌC: TÍNH TOÁN SONG SONG**

**HỌC KỲ 202/ NĂM HỌC 2021**

LỚP: L02

GIẢNG VIÊN HƯỚNG DẪN: THOẠI NAM

**Tp. Hồ Chí Minh – 2021**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ĐỀ TÀI: NHÂN MA TRẬN DÙNG GIẢI THUẬT STRASSEN**

**NHÓM SINH VIÊN:**

NGUYỄN VĂN KHANG – 1812554

NGUYỄN DUY KIÊN – 1812704

NGUYỄN ANH KHOA – 1812649

DƯƠNG MINH NHỊ LONG – 1812847

**Tp. Hồ Chí Minh – 2021**

*MỤC LỤC*

NỘI DUNG ………………………………………………………………….2

1. Bài toán …………………………………………………………………………...2

2. Giải pháp ………………………………………………………………………….2

2.1. Phương pháp chia để trị …………………………………………………2

2.2. Giải thuật nhân ma trận Strassen ………………………………………..3

2.3. Lập trình OpenMP ………………………………………………………4

3. Môi trường thử nghiệm và kết quả ……………………………………………….4

3.1. Nguồn dữ liệu tính toán …………………………………………………4

3.2. Kết quả …………………………………………………………………..4

4. Đánh giá ………………………………………………………………………….8

TÀI LIỆU THAM KHẢO …………………………………………………...9

**NỘI DUNG**

***1. Bài toán***

- Viết chương trình thực hiện load hai ma trận A và B và tiến hành nhân hai ma trận trên bằng giải pháp Divide-and-Conquer với các kích thước là 100x100, 1000x1000, 10000x10000 và số lượng core tính toán trong máy vật lý tối đa là 4, 8 &16 cores.

***2. Giải pháp***

- Trong bài tập lớn này, chúng em sử dụng phương pháp “chia để trị” (Devide-and-Conquer) để chia nhỏ bài toán, giải quyết từng bài toán nhân ma trận con bằng “thuật toán nhân ma trận Strassen” để tối ưu cho phép toán nhân ma trận.

- Chúng em sử dụng ngôn ngữ C/C++ và thư viện lập trình “OpenMP” để lập trình xử lý song song ở các luồng khác nhau nhằm tăng tốc độ xử lý cho chương trình.

***2.1. Phương pháp chia để trị (Devide-and-Conquer)***

- Trong cách tiếp cận theo phương pháp chia để trị, bài toán lớn ban đầu được chia thành các bài toán con nhỏ hơn và sau đó bài toán con được giải quyết một cách độc lập. Chúng ta có thể tiếp tục phân chia bài toán con thành các bài toán con nhỏ hơn, cho đến khi không thể tiến hành phân chia được nữa hoặc đủ để giải quyết. Bài toán con nhỏ nhất được gọi là “nguyên tử”. Lời giải của tất cả các bài toán con cuối cùng được hợp nhất để có được lời giải của bài toán ban đầu. Quá trình chung của phương pháp chia để trị có thể được tiếp cận như sau:

+ Bước chia: Chia nhỏ bài toán thành các bài toán nhỏ hơn. Bước này thường thực hiện bằng cách tiếp cận đệ quy để chia bài toán cho đến khi không có bài toán con nào có thể chia được nữa. Ở giai đoạn này, các bài toán con trở thành nguyên tử về bản chất nhưng vẫn thể hiện một phần nào đó của bài toán ban đầu.

+ Bước giải quyết: Bước này nhận được rất nhiều bài toán con nhỏ hơn cần giải quyết. Nói chung, ở cấp độ này, các bài toán con được coi là “tự giải quyết”.

+ Bước kết hợp: Khi các bài toán con nhỏ hơn được giải quyết, giai đoạn này kết hợp chúng một cách đệ quy cho đến khi chúng hình thành một giải pháp của bài toán ban đầu.

***2.2. Giải thuật nhân ma trận Strassen***

- Giải thuật nhân ma trận Strassen là giải thuật nhân ma trận áp dụng phương pháp chia để trị. Với mỗi ma trận vuông A, B, C có kích thước n×n, chúng ta chia chúng thành 4 ma trận con, và biểu diễn tích A×B = C, theo các ma trận con đó:

C1,1 = A1,1B1,1 + A1,2B2,1

C1,2 = A1,1B1,2 + A1,2B2,2

C2,1 = A2,1B1,1 + A2,2B2,1

C2,2 = A2,1B1,2 + A2,2B2,2

- Chúng ta định nghĩa ra các ma trận M mới như sau:

M1 = (A1,1 + A2,2)(B1,1 + B2,2)

M2 = (A2,1 + A2,2)B1,1

M3 = A1,1(B1,2 - B2,2)

M4 = A2,2(B2,1 - B1,1)

M5 = (A1,1 + A1,2)B2,2

M6 = (A2,1 - A1,1)(B1,1 + B1,2)

M7 = (A1,2 - A2,2)(B2,1 + B2,2)

- Biểu diễn lại các phần tử của C theo M như sau:

C1,1 = M1 + M4 – M5 + M7

C1,2 = M3 + M5

C2,1 = M2 + M4

C2,2 = M1 – M2 + M3 + M6

- Bằng cách này, chúng ta chỉ cần 7 phép nhân thay vì 8 phép nhân như phương pháp nhân ma trận thông thường. Nếu xem phép toán nhân là phép toán trừu tượng, độ phức tạp của giải thuật Strassen cho bài toán nhân ma trận kích thước n\*n là O(nlog7) nhanh hơn cách tiếp cận thông thường là O(n3), tuy nhiên với ma trận nhỏ thì điều này không đúng do chi phí phép cộng có thể nhiều hơn chi phí phép nhân tiết kiệm được.

***2.3. Lập trình OpenMP***

- OpenMP là giao diện lập trình ứng dụng (API), cung cấp một mô hình di động, có thể được sử dụng để định hướng sự song song đa luồng, chia sẻ bộ nhớ dùng chung. - Bao gồm ba thành phần API chính:

+ Chỉ thị trình biên dịch

+ Quy định thư viện thời gian chạy

+ Các biến môi trường

- Trong bài toán nhân ma trận trên, OpenMP được sử dụng để song song hóa quá trình tính toán bằng cái tạo ra các luồng khác nhau thực hiện các phép toán độc lập trên cơ chế chia sẻ tài nguyên bộ nhớ. Lập trình song song sử dụng OpenMP giúp tăng quá trình tính toán đáng kể so với tính toán tuần tự.

***3. Môi trường thử nghiệm và kết quả***

- Sử dụng lập trình OpenMP với ngôn ngữ C/C++ để song song hóa tính toán.

- Tính toán trên các thiết bị xử lý vật lý với số lượng core cho phép (4, 8 & 16 core).

***3.1. Nguồn dữ liệu tính toán***

- Sử dụng các ma trận có kích thước NxN khác nhau 100x100, 1000x1000, 10000x10000.

- Các phần tử trong ma trận là các số nguyên có giá trị trong khoảng (-100, 100) và được khởi tạo ngẫu nhiên bằng cách sử dụng hàm rand() trong C/C++.

***3.2. Kết quả***

- Kết quả là thời gian thực thi tính toán của giải thuật được đo bằng hàm clock trong thư viện có sẵn của C/C++ với các kích thước ma trận khác nhau và số lượng core vật lý cho phép.

- Trong báo cáo này, chúng em so sánh thời gian thực thi của các giải thuật Naïve (nhân ma trận theo phương pháp thông thường), giải thuật Strassen tuần tự với các base case (kích thước ma trận nguyên tử) khác nhau, giải thuật Strassen song song với các base case khác nhau, ở đây thuật toán strassen tính toán trên ma trận vuông có kích thước ứng với 2k.

- Ma trận với kích thước: 128x128 (biểu diễn cho ma trận 100x100)

+ Cores: 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Giải thuật Naïve | 0.010 | 0.020 | 0.020 | 0.020 | 0.010 | 0.016 |
| Strassen tuần tự  Base = 1 | 1.185 | 1.054 | 1.050 | 1.095 | 1.145 | 1.105 |
| Strassen tuần tự  Base = 16 | 0.016 | 0.017 | 0.016 | 0.016 | 0.017 | 0.082 |
| Strassen song song  Base = 1 | 4.017 | 4.829 | 4.405 | 4.418 | 4.321 | 4.398 |
| Strassen song song  Base = 16 | 0.015 | 0.008 | 0.012 | 0.008 | 0.009 | 0.010 |

=> Speed up: Time Naïve / Time Strassen song song = 0.016 / 0.010 = 1.6

+ Cores: 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 1 | 0.295 | 0.405 | 0.302 | 0.299 | 0.292 | 0.318 |
| Strassen song song  Base = 16 | 0.040 | 0.034 | 0.035 | 0.038 | 0.037 | 0.036 |

+ Core 16:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 1 | 0.344 | 0.298 | 0.295 | 0.417 | 0.420 | 0.345 |
| Strassen song song  Base = 16 | 0.040 | 0.042 | 0.034 | 0.039 | 0.037 | 0.038 |

- Ma trận với kích thước: 1024x1024 (biểu diễn cho kích thước 1000x1000)

+ Cores: 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Giải thuật Naïve | 6.294 | 6.377 | 6.634 | 6.602 | 6.228 | 6.427 |
| Strassen tuần tự  Base = 1 | 401.478 | 407.479 | 404.436 | 406.575 | 407.973 | 405.588 |
| Strassen tuần tự  Base = 16 | 6.294 | 6.377 | 6.634 | 6.602 | 6.228 | 6.427 |
| Strassen song song  Base = 1 | 90.197 | 90.507 | 90.297 | 90.184 | 90.086 | 90.254 |
| Strassen song song  Base = 16 | 4.061 | 3.828 | 3.943 | 3.927 | 3.988 | 3.949 |

=> Speed up: Time Naïve / Time Strassen song song = 6.427 / 3.949 = 1.627

+ Cores: 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 1 | 88.232 | 73.254 | 76.314 | 80.119 | 80.086 | 79.600 |
| Strassen song song  Base = 16 | 1.377 | 1.206 | 1.191 | 1.192 | 1.018 | 1.197 |

+ Core 16:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 1 | 84.338 | 70.885 | 90.297 | 90.027 | 90.178 | 85.145 |
| Strassen song song  Base = 16 | 1.124 | 1.240 | 0.958 | 1.257 | 1.258 | 1.167 |

- Ma trận với kích thước: 2048x2048

+ Cores: 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Giải thuật Naïve | 96.983 | 96.169 | 96.711 | 96.846 | 97.719 | 96.886 |
| Strassen tuần tự  Base = 16 | 27.740 | 27.600 | 27.630 | 28.160 | 27.580 | 27.742 |
| Strassen tuần tự  Base = 32 | 25.930 | 24.470 | 25.590 | 25.650 | 24.340 | 25.196 |
| Strassen song song  Base = 16 | 6.617 | 6.730 | 7.857 | 6.986 | 7.719 | 7.182 |
| Strassen song song  Base = 32 | 5.897 | 6.446 | 5.903 | 5.818 | 5.758 | 5.964 |

=> Speed up: Time Naïve / Time Strassen song song = 96.886 / 5.964 = 16.245

+ Cores: 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 16 | 8.912 | 8.967 | 8.954 | 7.986 | 8.522 | 8.668 |
| Strassen song song  Base = 32 | 7.636 | 8.059 | 7.967 | 7.555 | 7.522 | 7.748 |

+ Core 16:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Giải thuật | Lần 1 | Lần 2 | Lần 3 | Lần 4 | Lần 5 | Trung bình |
| Strassen song song  Base = 16 | 6.517 | 6.636 | 6.783 | 7.233 | 6.822 | 6.796 |
| Strassen song song  Base = 32 | 5.516 | 4.110 | 4.308 | 4.238 | 4.438 | 4.522 |

***4. Đánh giá***

- Đối với kích thước ma trận nhỏ (< 1000), thuật toán Strassen có thể có thời gian thực thi cao hơn thuật toán nhân ma trận thông thường.

- Đối với ma trận có kích thước vừa và lớn, thuật toán Strassen cho thời gian thực thi tốt hơn.

- Tuy nhiên, cần lưu ý thông số base case, vì với base case không thích hợp, thuật toán có thể phải tiêu tốn tài nguyên cho việc chia nhỏ, dẫn đến thời gian thực thi tốn nhiều hơn thuật toán nhân ma trận thông thường. Việc chọn được base case ngoài phụ thuộc vào kích thước ma trận, còn cần phải xem xét thêm về thiết bị xử lý tính toán.

- Kết hợp tính toán song song, thời gian thực thi được rút ngắn đáng kể khi ma trận có kích thước càng lớn, tính toán song song phù hợp với bài toán có kích thước lớn và nhiều bộ xử lý.

- Đối với ma trận có kích thước 10000x10000, trong khả năng cho phép của thiết bị vật lý cá nhân và hệ thống được cung cấp, không đủ đáp ứng nên không có kết quả ở bài báo cáo này.

**TÀI LIỆU THAM KHẢO**

1. Barry Wilkinson and MiChael Allen (2005), *Parallel Programming: Techniques*

*and Applications Using Networked Workstations and Parallel Computers*.

2. OpenMP, “OpenMP Application Programming Interface ”, truy cập ngày

15/6/2021.

<https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf>

3. Tuan Nguyen, Alex Adamson, Andreas Santucci (2016), *Distributed Algorithms*

*and Optimization.*

<https://stanford.edu/~rezab/classes/cme323/S16/notes/Lecture03/cme323_lec3.pdf>