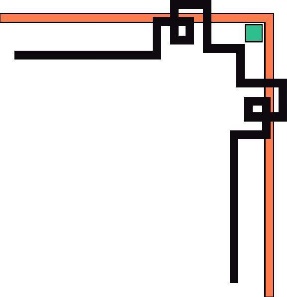
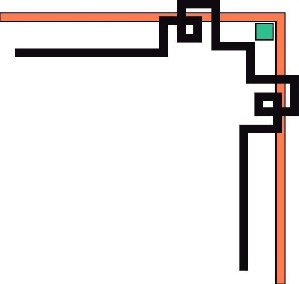
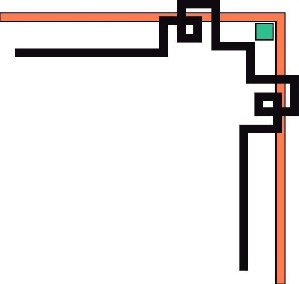
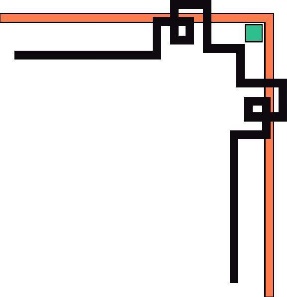
**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**Khoa Khoa học và Kỹ thuật Máy tính**



**BÁO CÁO ASSIGNMENT**

**Xử lý song song & hệ phân bố**

**ĐỀ TÀI: Hiện thực giải thuật bucket sort**

**Giảng viên hướng dẫn: Nguyễn Quang Hùng**

Nhóm thực hiện**:**

|  |  |
| --- | --- |
| Họ và tên | MSSV |
| Lê Đức Anh | 51000050 |
| Lâm Chí Hải | 51000843 |
| Nguyễn Xuân Thành | 51003010 |

TP.Hồ Chí Minh, tháng 4 – 2014

*TP. Hồ Chí Minh, tháng 12 – 1013*

Nội dung

[I. Sơ lược về giải thuật 2](#_Toc388458024)

[II. Phân tích độ hiệu quả và speed up theo số lượng processor 4](#_Toc388458025)

[1. Lý thuyết 4](#_Toc388458026)

[2. Thực nghiệm 5](#_Toc388458027)

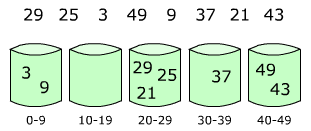
[III. Giải thích kết quả 7](#_Toc388458028)

[IV. Kết luận 9](#_Toc388458029)

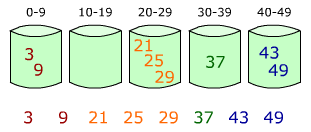
[V. Tài liệu tham khảo 10](#_Toc388458030)

# I. Sơ lược về giải thuật

Giải thuật bucket sort là giải thuật sắp sếp trên dữ liệu lớn và tận dụng được kiến trúc tính toán song song. Đây là một dạng giải thuật song song dữ liệu (data parallel). Bước đầu, dữ liệu được chia thành từng bucket. Sau đó từng processor sẽ thực hiện sắp xếp từng bucket. Lấy kết quả trong từng bucket sẽ có được dãy số đã sắp xếp. Quá trình được minh họa bởi hình sau:



*Dãy chưa được sắp xếp*



*Dãy đã được sắp xếp*

Giải thuật bucket sort trong bài tập lớn này được nhóm hiện thực bằng thư viện mpi c++. Giải thuật hiện thực theo mô hình master – wokers. Trong đó process 0 đóng vai trò phân phối dữ liệu đến các process khác cũng như thu thập kết quả của từng process gửi về. Các process khác đóng vai trò là 1 bucket và chịu trách nhiệm sắp xếp dữ liệu mà nó nhận được và gửi dữ liệu về process 0. Dữ liệu ban đầu được lưu xuống file và được process 0 đọc lên. Từng giá trị một sẽ được xác định gửi đến process nào qua công thức sau:

offset = ceil((maxVal – minVal) / numBucket)

procID = (value – minVal) / offset

Trong đó:

maxVal: giá trị lớn nhất trong tập dữ liệu.

minVal: giá trị nhỏ nhất trong tập dữ liệu.

numBucket: số lượng bucket.

value: giá trị hiện tại.

offset: khoảng giá trị của 1 bucket (được làm tròn lên).

proID: id của process nhận.

Các bucket process (khác process 0) sử dụng giải thuật sắp xếp được hiện thực sẵn tương tự như quick sort. Các process này cũng được giả định là có đủ bộ nhớ để lưu trữ dữ liệu nhận được. Quá trình gửi nhận dữ liệu chỉ sử dụng 2 hàm MPI\_Send(), MPI\_Recv() và được đồng bộ qua các giá trị: INIT\_PHRASE, SORT\_PHRASE, GATHER\_PHRASE và END\_PHRASE. Các giá trị trên đánh dấu từng giai đoạn của thuật toán. Giai đoạn 1 là qua trình phân phối dữ liệu đến từng process. Giai đoạn 2 báo hiệu kết thúc việc nhận dữ liệu và thực hiện sắp xếp trên từng process. Giai đoạn 3 là giai đoạn thu thập kết quả theo thứ tự của từng bucket process. Giai đoạn 4 là ghi kết quả cuối cùng xuống file và kết thúc giải thuật.

# II. Phân tích độ hiệu quả và speed up theo số lượng processor

### Lý thuyết

Một cách tổng quát, thời gian thực thi của giải thuật bucket sort bao gồm thời gian phân chia dữ liệu vào các bucket và thời gian sắp xếp các bucket đó.

Giải thuật quick sort được sử dụng để sắp xếp từng bucket có độ phức tạp là:

*O(n/p log2(n/p)).*

*với p = số bucket, n là chiều dài dãy ban đầu.*

Vậy nên sắp xếp p bucket sẽ có độ phức tạp là:

*O(n log2(n/p)).*

Độ phức tạp của việc chia dữ liệu vào từng bucket :

*O(n log2(p)).*

Do đó:

Độ phức tạp của giải thuật bucket sort tuần tự sẽ là:

*O(n(log2(p) + log2(n/p))).*

Độ phức tạp của giải thuật bucket sort song song sẽ là:

*O(n(log2(p) + log2(n/p)/p)).*

Trong giải thuật song song, mỗi processor sẽ tương ứng với 1 bucket và sẽ thực thi việc sắp xếp đồng thời với nhau.

Công thức tính speed up:

S =

*Ts: thời gian thực thi của giải thuật tuần tự tốt nhất.*

*Tp: thời gian thực thi của giải thuật song song.*

Speed up giữa 2 giải thuật bucket sort tuần tự và song song:

*S = = (1+L)*

*với L =*

Speed up giữa giải thuật quick sort và bucket sort song song:

*S = =*

Công thức tính độ hiệu quả (efficiency):

E =

*S: speed up.*

*N: số processor.*

### Thực nghiệm

a. Cấu hình máy tiến hành test

Intel core i5: 4 processor

RAM: 4Gb

Dữ liệu để test bao gồm 4 file chứa các giá trị từ 0 đến 99 và có kích thước dữ liệu lần lượt là: 1 triệu số, 10 triệu số, 100 triệu số và 1 tỷ số. Các dãy số trong từng file có thứ tự ngẫu nhiên.

|  |  |  |
| --- | --- | --- |
| Tên file | Chiều dài dãy số | Kích thước lưu trữ |
| 1M.txt | 1 triệu | 2.76 Mb |
| 10M.txt | 10 triệu | 27.6 Mb |
| 100M.txt | 100 triệu | 276 Mb |
| 1B.txt | 1 tỷ | 2.7 Gb |

b. Kết quả các lần thử nghiệm:

4 processors 1 triệu số

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Gửi dữ liệu | Sắp xếp và thu thập | Sắp sếp tuần tự |
| 1 | 0.63953 | 1.07379 | 0.89 |
| 2 | 0.64550 | 1.06954 | 0.88 |
| 3 | 0.63964 | 1.07023 | 0.89 |
| Trung bình | 0.64156 | 1.07119 | 0.8867 |
| Tổng thời gian | 1.71275 | | 0.8867 |

4 processors 10 triệu số

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Gửi dữ liệu | Sắp xếp và thu thập | Sắp xếp tuần tự |
| 1 | 6.38744 | 10.5835 | 9.187 |
| 2 | 6.39025 | 10.5924 | 9.188 |
| 3 | 6.38623 | 10.5833 | 9.185 |
| Trung bình | 6.38797 | 10.5864 | 9.1867 |
| Tổng thời gian | 16.97437 | | 9.1867 |

4 processors 100 triệu số

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Gửi dữ liệu | Sắp xếp và thu thập | Sắp xếp tuần tự |
| 1 | 63.0986 | 114.683 | 101.64 |
| 2 | 63.1277 | 114.676 | 101.65 |
| 3 | 63.0875 | 115.023 | 101.64 |
| Trung bình | 63.1046 | 114.794 | 101.6433 |
| Tổng thời gian | 117.8986 | | 101.6433 |

4 processors 1 tỷ số

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Gửi dữ liệu | Sắp xếp và thu thập | Sắp xếp tuần tự |
| 1 | 695.365 | 1696.91 | 4321.62 |
| 2 | 700.235 | 1695.32 | 4322.05 |
| 3 | 694.223 | 1693.95 | 4321.92 |
| Trung bình | 696.607 | 1695.393 | 4321.86 |
| Tổng thời gian | 2392 | | 4321.86 |

c. Đồ thị biểu diển sự thay đổi

*Đồ thị thời gian thực thi của giải thuật tuần tự và song song trên tập dữ liệu test*

d. Đánh giá các chỉ số hiệu quả

Speed up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước dữ liệu \ số processors | 2 | 4 | 6 | 8 |
| 1 triệu |  | 0.5177 |  |  |
| 10 triệu |  | 0.5412 |  |  |
| 100 triệu |  | 0.8621 |  |  |
| 1 tỉ |  | 1.8068 |  |  |

Efficiency

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước dữ liệu \ số processors | 2 | 4 | 6 | 8 |
| 1 triệu |  | 0.1294 |  |  |
| 10 triệu |  | 0.1353 |  |  |
| 100 triệu |  | 0.2155 |  |  |
| 1 tỉ |  | 0.4517 |  |  |

# III. Giải thích kết quả

Từ kết quả thực nghiệm thu được có thể nhận thấy thời gian thực thi giải thuật tăng gần như tuyến tính với lượng dữ liệu. Chiếm phần lớn thời gian chạy là thời gian gửi nhận dữ liệu. Trên hệ thống multi-computer, thời gian gửi nhận dữ liệu phụ thuộc rất lớn vào đường truyền mạng. Nguyên nhân là do process chính gửi và nhận dữ liệu từ các process con theo từng giá trị một . Ngoài ra còn phải kể tới chi phí đồng bộ. Trong quá trình thu thập kết quả, process chính phải nhận kết quả một cách tuần tự từ các process khác nên dẫn tới kết quả là thời gian chạy giải thuật phụ thuộc vào bucket process chạy chậm nhất do thời gian chạy các bucket process này không bằng nhau. Giải thuật đạt thời gian chạy tốt nhất khi tập dữ liệu lớn hơn 109.

Ngoài vấn đề thời gian chạy thì một số vấn đề sau cũng rất đáng quan tâm:

* Khi giải thuật chạy trên hệ thống nhiều processor, nếu số lượng processor lớn hơn hoặc bằng với số lượng process thì sẽ đảm bảo hiệu suất tốt hơn. Trong trường hợp số processor nhỏ hơn số process, khi đó 1 processor sẽ có nhiều hơn 2 process, khi đó vì phụ thuộc vào cơ chế lập lịch của hệ điều hành mà hiệu suất thực thi sẽ thấp hơn so với trường hợp trên. Trong một vài trường hợp khi không còn bộ nhớ cho 1 process (thông thường 1 process có kích thước nhỏ hơn 4Gb), hệ điều hành sẽ sử dụng pagefile lưu tạm trạng thái của process xuống đĩa cứng và do đó làm chương trình bị gián đoạn và chạy chậm rất nhiều so với mong đợi.
* Khi giải thuật này được triển khai trên hệ thống multi – computer thì cần chú ý đến dung lượng bộ nhớ trong của từng máy. Vì mỗi máy phải thực hiện sắp xếp toàn bộ dữ liệu mà nó nhận được nên nếu bộ nhớ không đủ sẽ dẫn tới không thể thực thi bucket đó và chương trình có thể chạy vô tận.

# IV. Kết luận

Giải thuật bucket sort là một giải thuật tiêu biểu cho việc kết hợp tính toán song song với các giải thuật tuần tự đã biết. Đây là một giải thuật đơn giản nhưng hiệu quả và có thể dễ dàng triển khai trên thực tế. Bên cạnh đó cũng còn nhiều giải thuật sắp xếp song song hiệu quả hơn tương ứng với các kiến trúc phần cứng khác nhau mà trong tài liệu không đề cập tới.

# V. Tài liệu tham khảo

1. Sơ lược về giải thuật bucket sort: <http://en.wikipedia.org/wiki/Bucket_sort>.

2. Phân tích độ phức tạp của giải thuật bucket sort song song: slide bài giảng từ trang [http://homepages.math.uic.edu/~jan/mcs572/**parallelsorting**.pdf](http://homepages.math.uic.edu/~jan/mcs572/parallelsorting.pdf).