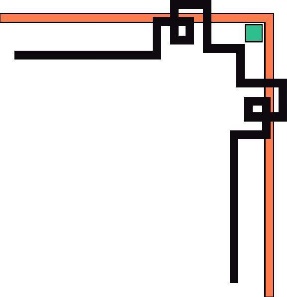
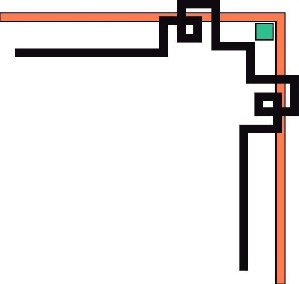
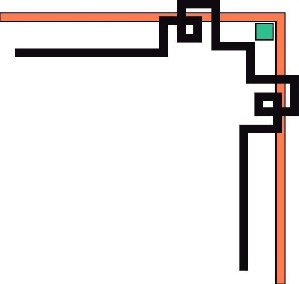
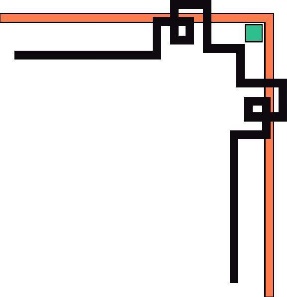
**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

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



**BÁO CÁO ASSIGNMENT**

**LẬP TRÌNH WEB**

**ĐỀ TÀI: Xây dựng website chơi game flash online**

**Giảng viên hướng dẫn:**

Nhóm thực hiện**:**

|  |  |
| --- | --- |
| Họ và tên | MSSV |
| Ngô Sỹ Khánh | 51001482 |
| Nguyễn Minh Tân |  |
| Nguyễn Lê Thanh Tùng |  |
| Nguyễn Xuân Thành | 51003010 |
| Phạm Văn Thành | 51003015 |

TP.Hồ Chí Minh, tháng 5 – 2014

*TP. Hồ Chí Minh, tháng 12 – 1013*

Nội dung

[I. Sơ lược về giải thuật 3](#_Toc385839595)

[II. Phân tích độ hiệu quả và speed up theo số lượng processor 4](#_Toc385839596)

[1. Lý thuyết 4](#_Toc385839597)

[2. Thực nghiệm 4](#_Toc385839598)

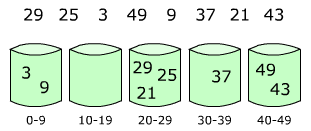
[III. Giải thích kết quả 5](#_Toc385839599)

[IV. Kết luận 6](#_Toc385839600)

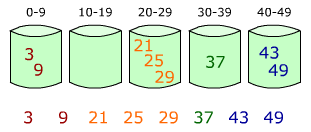
[V. Tài liệu tham khảo 7](#_Toc385839601)

# 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 processor 0 đóng vai trò phân phối dữ liệu đến các processors khác cũng như thu thập kết quả của từng processor gửi về. Các processor khác đóng vai trò là 1 bucket và chịu trách nhiệm sắp xếp dữ liệu mà nó nhận được và gửi dữ liệu về processor 0. Dữ liệu ban đầu được lưu xuống file và được processor 0 đọc lên. Từng giá trị một sẽ được xác định gửi đến processor 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 processor nhận.

Các bucket processor sử dụng giải thuật sắp xếp được hiện thực sẵn tương tự như quick sort. Các processor 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 processor. Giai đoạn 2 báo hiệu kết thúc việc nhận dữ liệu và thực hiện sắp xếp trên từng processor. Giai đoạn 3 là giai đoạn thu thập kết quả theo thứ tự của từng bucket processor. 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

Giả sử dữ liệu ban đầu có phân phối đều thì độ phức tạp của giải thuật bucket sort song song sẽ là:

*O(log n) với n / log n processors.*

Trong mỗi processor sẽ sử dụng giải thuật sắp sếp tuần tự hiệu quả nhất để sắp xếp dãy con. Trong trường hợp này nhóm sử dụng giải thuật tương tự quick sort với độ phức tạp là:

*O(k\*log k)*

*với k = N / M, N là chiều dài dãy ban đầu, M là chiều dài dãy con.*

### Thực nghiệm

Efficient

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

Speed up

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

(cái này là bảng tạm ko cần dung tới)

4 processors 1 triệu số

|  |  |  |
| --- | --- | --- |
|  | Send data time | Sort and gather result time |
| 1 | 0.6895s | 1.2070s |
| 2 | 0.6550s | 1.1949s |
| 3 | 0.6527s | 1.1927s |
| Trung bình | 0.6657s | 1.1982s |

4 processors 10 triệu số

|  |  |  |
| --- | --- | --- |
|  | Send data time | Sort and gather result time |
| 1 | 6.6026s | 12.3935s |
| 2 | 6.5690s | 12.2569s |
| 3 | 6.4541s | 16.5983 s |
| Trung bình | 6.5419s | 13.7496s |

4 processors 100 triệu số

|  |  |  |
| --- | --- | --- |
|  | Send data time | Sort and gather result time |
| 1 | 69.0429s | 329.7320s |
| 2 |  |  |
| 3 |  |  |
| Trung bình | 6.5419s | 13.7496s |

# 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 và xử lý IO (đọc ghi file). Thời gian gửi nhận dữ liệu phụ thuộc rất lớn vào đường truyền mạng. Trong khi đó thời gian đọc ghi file tuy đáng kể nhưng vẫn nhỏ hơn thời gian truyền nhận dữ liệu. Ngoài ra còn phải kể tới chi phí đồng bộ. Trong quá trình thu thập kết quả, processor chính phải nhận kết quả một cách tuần tự từ các processor khác nên dẫn tới kết quả là thời gian chạy giải thuật phụ thuộc vào bucket processor chạy chậm nhất do thời gian chạy các bucket processor này không bằng nhau. Giải thuật đạt thời gian chạy tốt nhất trong khoảng từ 4-6 processor.

Ngoài vấn đề thời gian chạy thì vấn đề lưu trữ dữ liệu cũng rất đáng quan tâm. Khi triển khai giải thuật này 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.

# 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: