TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO:

BÁO CÁO BÀI TẬP LỚN
MÔN XỬ LÝ SONG SONG VÀ HỆ PHÂN BỐ

GVHD: **Nguyễn Quang Hùng**

Nhóm: **7**

Đề bài: Hiện thực song song hóa giải thuật Dijkstra và đánh giá Speed Up

Sinh viên báo cáo:

Trần Lý Minh Tuấn **MSSV: 51304584**

Lê Tuấn Anh **MSSV: 51300080**

TP.HCM, NGÀY 30 THÁNG 04 NĂM 2017

MỤC LỤC

I. Giới thiệu đề tài:

1. Đề tài:

Hiện thực song song hóa giải thuật tìm đường đi ngắn nhất Dijkstra và phân tích kết quả thu được.

2. Các yêu cầu:

STT	Yêu cầu
1	Hiện thực giải thuật tuần tự hiệu quả
2	Hiện thực giải thuật song song hiệu quả
3	Đo speed up
4	Giải thích kết quả đo được

II. Giải thuật tuần tự:

1. Mã giả:

```
1  nv // number of vertices
2  notdone[] //list of vertices not check yet
3  ohd[] // 1-hop distances between vertices
4  parent[] // parent[i] =j mean: j is parent of i
5  mind[] // min distances found so far
6
7  sequenceDijkstra()
8      init notdone, ohd, parent;
9      read ohd from input file;
10
11     for loop = 0 to nv-1
12         min = infinity
13         minNode = 0
14
15         for i = 0 to nv-1
16             find min and minNode, which in notdone[] and mind[minNode] is minimum
17         end
18         update notdone[]
19
20         for j=0 to nv-1
21             update mind[] and parent[]
22         end
23     end
24 end
```

2. Phân tích giải thuật:

Theo như mã giả đã nêu ở trên ta thấy, giải thuật chạy gồm 2 vòng for chứa trong một vòng for lớn. Độ phức tạp của giải thuật là $O(V^2)$.

Vòng lặp ngoài cùng (dòng 11) để duyệt qua tất cả các node của đồ thị. Vòng lặp thứ 2 nhằm tìm ra node nào đang có đường đi tới root nhỏ nhất (root ở đây là node 0). Vòng lặp thứ 3 nhằm cập nhật lại khoảng cách nhỏ nhất từ một node trong đồ thị tới node gốc và cập nhật node cha của nó.

Việc sử dụng vòng lặp như vậy khiến cho thời gian chạy của giải thuật rất chậm. Ta có thể cải thiện được thời gian chạy của giải thuật bằng việc song song hóa nó.

III. Giải thuật song song:

1. Cơ sở lý thuyết:

Việc song song hóa bài toán tìm đường đi ngắn nhất bằng giải thuật Dijkstra được thực hiện dựa trên các hàm của MPI. MPI (Message Passing Interface) là một chuẩn mô tả các đặc điểm và cú pháp của một thư viện lập trình song song. Có rất nhiều thư viện dựa trên chuẩn MPI như MPICH, OpenMPI, LAM/MPI.

Trong bài toán này, ta sử dụng các hàm giao tiếp nhóm để hiện thực việc giao tiếp giữa các processes. Các processes chạy cùng một chương trình nhưng trên những phần dữ liệu khác nhau. Đó là mô hình SPMD (Single Program Multiple Data).

Để hiện thực việc giao tiếp giữa các processes ta sử dụng các hàm giao tiếp nhóm của MPI.

- Int MPI_Reduce(const void*sendbuf, void*recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm);
- Int MPI_Bcast(void*buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm);
- Int MPI_Gather(const void*sendbuf, int sendcount, MPI_Datatype datatype, void*recvbuf, int recvcount, MPI_Datatype datatype, int root, MPI_Comm comm);
- Ngoài ra còn có một số hàm cơ bản khác.

2. Mã giả:

```
nv // number of vertices
notdone[] //list of vertices not check yet
ohd[] // 1-hop distances between vertices
parent[] // parent[i] =j mean: j is parent of i
mind[] // min distances found so far
mymin[], // mymin[0] is min for my chunk,
            // mymin[1] is vertex which achieves that min
overallmin[], // overallmin[0] is current min over all nodes,
        // overallmin[1] is vertex which achieves that min
me // my process index
chunk // number of vertices handled by each node
nnodes // number of MPI nodes in the computation
startv,endv, // start, end vertices for this node

 // finds closest to 0 among notdone, among startv through endv
findmymin()
    mymin[0] = largeint;
    for i= startv to endv-1
        if (notdone[i] and mind[i] < mymin[0])
            mymin[0] = mind[i];
            mymin[1] = i;
        end
    end
end
```

```
updatemymind(){ // update my mind segment
            // for each i in [startv,endv], ask whether a shorter path to i
            // exists, through mv
    mv = overallmin[1];
    md = overallmin[0];
    for i= startv to endv-1
        if (md + ohd[mv*nv+i] < mind[i])
            mind[i] = md + ohd[mv*nv+i];
            parent[i] = mv;
        end
    end
end
```

```
parallelDijkstra()
    init notdone[], ohd[], parent[], mind[];
    read ohd from input file;
    mind[0] =0;
    parent[0] =0;

    for step =0 to nv-1
        findmymin();
        MPI_Reduce(mymin,overallmin,1,MPI_2INT,MPI_MINLOC,0,comm);
        MPI_Bcast(overallmin,1,MPI_2INT,0,comm);
        notdone[overallmin[1]] = 0;
        updatemymind(startv,endv);
    end
    MPI_Gather(mind+startv,chunk,MPI_INT,mind,chunk,MPI_INT,0,comm);
    MPI_Gather(parent+startv,chunk,MPI_INT,parent,chunk,MPI_INT,0,comm);
end
```

3. Phân tích giải thuật:

Ý tưởng của giải thuật dựa trên việc phân chia công việc cho các processes. Mỗi process sẽ đảm nhận nv/nnodes đỉnh với nv là số đỉnh của đồ thị và nnodes là số process được sử dụng.

Các process sẽ tìm ra đỉnh nào có khoảng cách tới đỉnh gốc nhỏ nhất. Sau đó gửi thông tin đỉnh đó về cho process 0(root). Process 0 sẽ tìm trong các đỉnh được gửi về đỉnh có khoảng cách tới đỉnh gốc nhỏ nhất. Các process sẽ cập nhật lại khoảng cách từ các đỉnh mình quản lý tới đỉnh gốc. Cuối cùng sau khi thực hiện duyệt qua tất cả các đỉnh của đồ thị, các processes sẽ gửi toàn bộ thông tin các đỉnh mà nó quản lý về cho process 0, thông tin bao gồm khoảng cách nhỏ nhất từ 1 đỉnh tới đỉnh gốc và đỉnh cha của đỉnh đó.

Theo lý tưởng, nếu bỏ qua thời gian giao tiếp giữa các processes, thời gian chạy của giải thuật sẽ là $\frac{n}{p}$ với n là số đỉnh của đồ thị và p là số processes. Khi đó speed up là p và độ hiệu quả là 1.

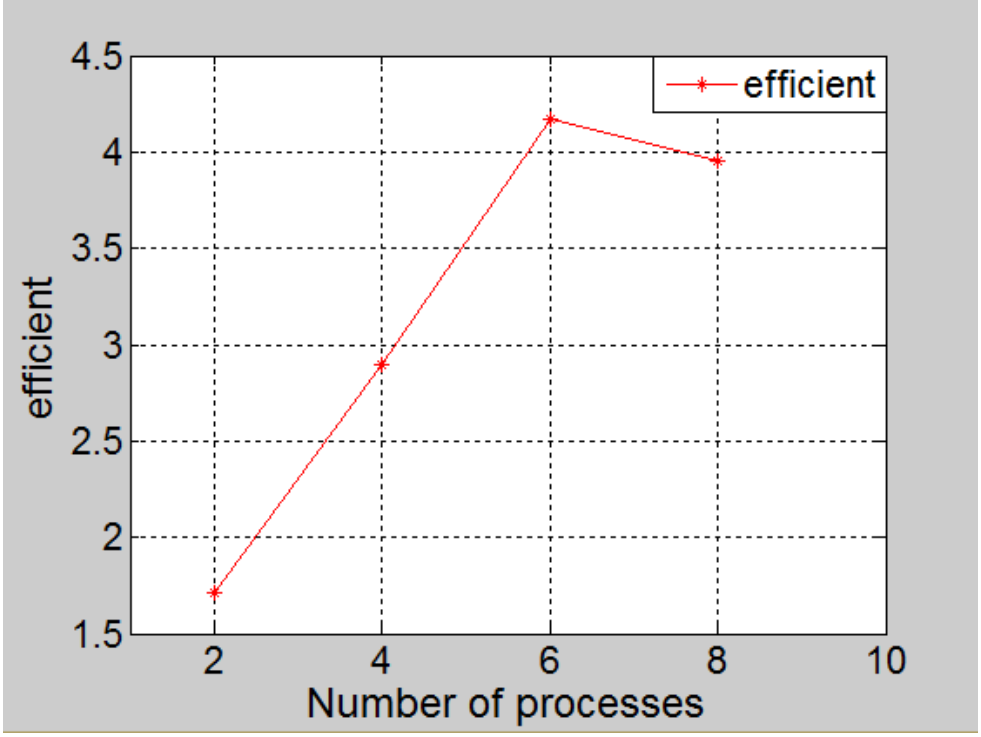
IV: Kết quả:

1. Kết quả chay giải thuật:

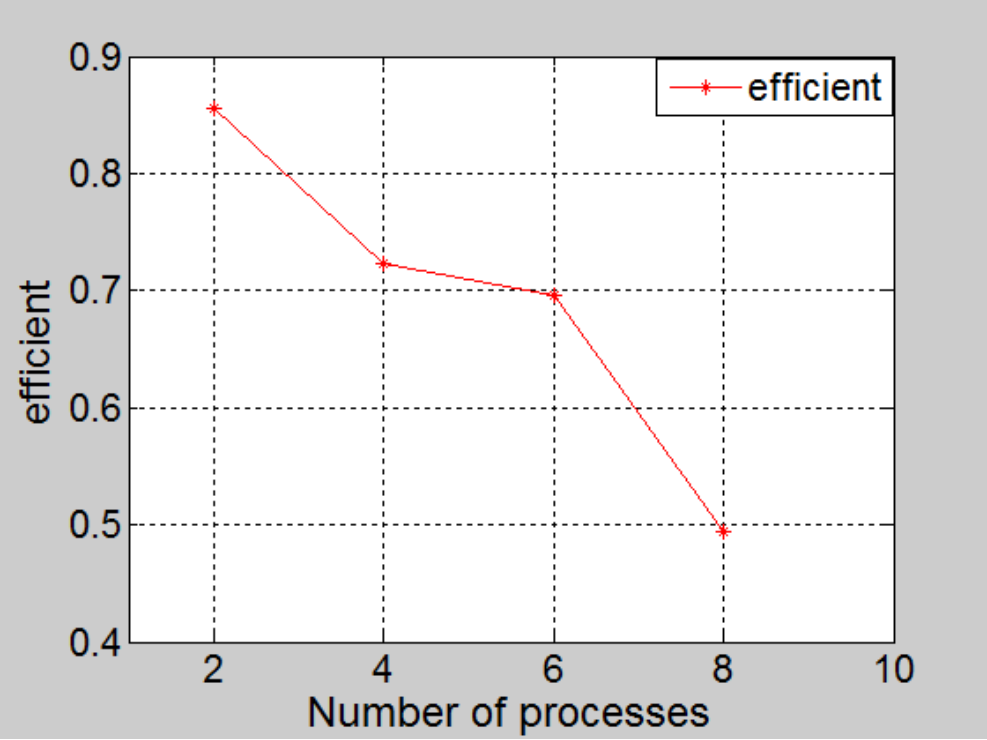
Việc chạy giải thuật được thực hiện trên phương pháp tuần tự và song song với 2,4,6,8 processes, đồ thị có 24000 đỉnh. Với mỗi trường hợp chạy 5 lần và lấy kết quả trung bình. Thời gian chạy tình bằng giây(s)

Lần chạy	Tuần tự	2 process	4 process	6 process	8 process
1	3.58	2.119289	1.186781	0.851713	0.65631
2	3.58	2.08492	1.131365	0.918454	0.984692
3	3.57	2.071674	1.137166	0.797883	1.064709
4	3.57	2.088134	1.384562	0.837526	0.837212
5	3.57	2.085246	1.335901	0.87519	0.977047
Trung bình	3.574	2.089853	1.235155	0.856273	0.903994
Speed up		1.7102	2.8936	4.1739	3.9536
efficient		0.8551	0.7234	0.6957	0.4942

a. Phân tích kết quả:



ĐỒ THỊ BIỂU DIỄN SPEED UP TƯƠNG ỨNG VỚI SỐ PROCESSES



ĐỒ THỊ BIỂU DIỄN ĐỘ HIỆU QUẢ TƯƠNG ỨNG VỚI SỐ PROCESSES

Theo như đồ thị ta thấy, speed up tăng tuyến tính khi tăng từ 2 processes lên 6 processes nhưng lại có dấu hiệu giảm khi tăng từ 6 processes lên 8 processes và độ hiệu quả có chiều hướng giảm khi ta tăng số processes lên.

Sự thay đổi của speed up và độ hiệu quả có thể được giải thích như sau: Khi ta tăng số lượng processes lên thì đồng thời thời gian giao tiếp giữa các processes cũng tăng lên.

Khi chạy giải thuật với 2 processes ta thấy speedup còn thấp nhưng độ hiệu quả là rất cao (sắp xỉ 1) là do lúc này số lượng processes còn thấp nên speedup chỉ đạt tối đa là 2 (trường hợp lý tưởng) và thời gian giao tiếp giữa các processes là rất nhỏ nên độ hiệu quả đạt rất cao. Nhưng khi tăng từ 2 processes lên 6 processes, thời gian giao tiếp giữa các processes cũng tăng lên nhưng vẫn còn chưa đáng kể so với thời gian chạy giải thuật. Do đó ta vẫn thấy speedup tăng liên tục theo đường gần như tuyến tính. Nhưng khi ta tăng lên 8 processes thì thời gian chạy giải thuật lại bị ảnh hưởng khá nhiều bởi thời gian giao tiếp của các processes nên speedup đã có dấu hiệu giảm xuống và độ hiệu quả cũng đạt rất thấp(sấp xỉ 0.5).

Do đó ta thấy không phải cứ tăng số lượng processes thì speedup sẽ tăng như là trường hợp lý tưởng đã phân tích ở phần trên mà việc sử dụng bao nhiêu processes còn phụ thuộc vào kích thước của bài toán.

2. Phụ lục:

a. Phân công của nhóm:

Họ tên	Mssv	Phân công
Trần Lý Minh Tuấn	51304584	Code giải thuật song song
Lê Tuấn Anh	51300080	Viết báo cáo, code giải thuật tuần tự

b. Tài liệu tham khảo:

http://www.mpich.org/static/docs/v3.1/www3/

SONG SONG HÓA THUẬT TOÁN DIJKSTRA TÌM ĐƯỜNG ĐI NGẮN NHẤT TỪ MỘT ĐỈNH ĐẾN TẤT CẢ CÁC ĐỈNH – Nguyễn Đình Lầu, Trần Ngọc Việt – Trường cao đẳng giao thông vận tải II

An Implementation Of Parallelizing Dijkstra's Algorithm – CSE Course Project- Zilong Ye