Lab 5 Parallel Programming with MPI

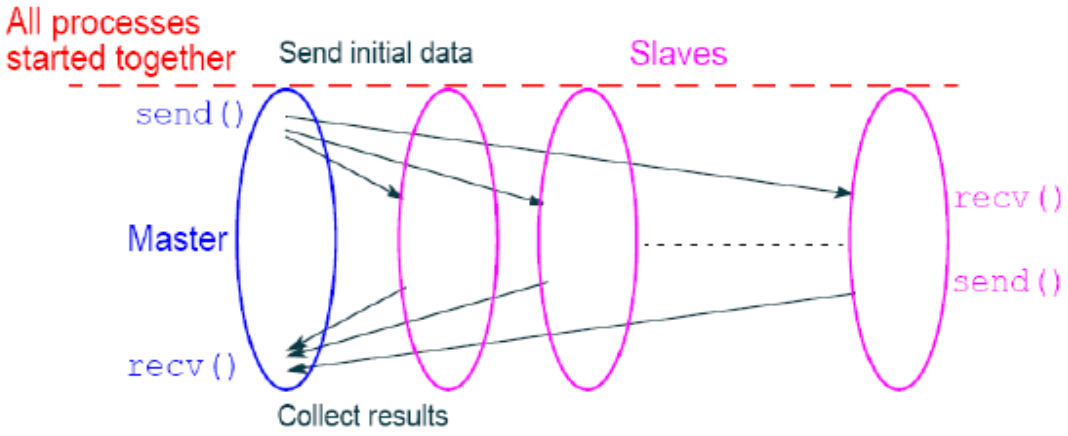
Master-Worker model

1. Mục tiêu

- SV tìm hiểu cách song song hóa bài toán theo mô hình master-worker
- SV phát triển chương trình đã song song hóa theo mô hình workpool (processor farms).
- Nhận xét về kết quả và ứng dụng của cả 2 mô hình.

2. Nội dung

2.1 Master-Worker Model



Usual MPI approach

2.2. Chương trình minh họa

2.2.1 Khung lập trình mẫu cho mô hình Master-Worker

```
#include <mpi.h>
int main(int argc, char ** argv){
    int rank,size;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    if(rank == 0)
            master(size);
    else
            worker();
    MPI_Finalize();
    return 0;
}
```

☺ **Lưu ý**: mô hình này đã được sử dụng trong bài lab 4 cho bài toán tính tổng !

2.2.2 Chương trình nhân ma trận và vector theo kiểu mẫu Master-Woker

```
#include <mpi.h>
#include <stdio.h>
#define N 100000

long matrixA[N][N];

int main(int argc, char ** argv){
    int rank,size;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
```

```
    if(rank == 0)
      master(size);
    else
      worker();

    MPI_Finalize();
    return 0;
}

long minFunc(int x, int y)
{
    return (long)(x<y?x:y);
}

//////////////////////////////////////////////////////
int master(int procs){
    long vectorC[N];
    long i,j,dotp, sender, row, numsent=0;
     double starttime = 0, stoptime = 0;
    MPI_Status status;
    /* Initialize data */
    for(i=0; i < N; i++){
       vectorC[i] = i;        //vector
      for(j=0; j < N; j++)
          matrixA[i][j] = 1;  //matrix
     }
     starttime = MPI_Wtime();
    /* distribute data to worker */
    for(i=1; i < minFunc(procs, N); i++)
```

```
        MPI_Send (&vectorC, N, MPI_LONG, i, N+1 ,MPI_COMM_WORLD); //vector
    for(i=1; i < minFunc(procs, N); i++)
   {
      MPI_Send(&matrixA[i-1][0], N, MPI_LONG, i, i, MPI_COMM_WORLD );  //matrix
      numsent++;
   }
   /* receive result and distribute data */
   for(i=0; i < N; i++)
   {
      MPI_Recv(&dotp, 1, MPI_LONG, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD,
&status);
      /* SV xac dinh process gui ket qua ve va gui tiep du lieu cho no ??? */
      sender = ….; // TODO
      row      = ….; // TODO
      vectorC[row] = dotp;
      if(numsent < N) {
           MPI_Send(&matrixA[numsent][0], N, MPI_LONG, sender, numsent+1, MPI_COMM_WORLD);
           numsent++;
        }
      else {
            /* SV gui thong diep thong bao ket thuc cong viec */
           MPI_Send(MPI_BOTTOM, 0, MPI_LONG, sender, 0, MPI_COMM_WORLD);
      }
   }
   /* In ket qua de xac dinh tinh dung dan cua chuong trinh */
   for(i = 0; i < N; i++)
       fprintf(stdout,"%ld ",vectorC[i]);
    stoptime = MPI_Wtime();
    fprintf(stdout,"\n\nDuration time: %lf \n", (stoptime - starttime));
   return 0;
}
```

/* SV tìm hiểu mã nguồn chương trình và hoàn tất hàm worker */

```
int woker(){
    /* Cong viec cua worker */
    - Nhận dữ liệu từ master
    - Nhân vector dữ liệu vừa nhận với vector của nó
    - Gửi kết quả trả về
    - Đợi nhận thêm dữ liệu
    - Nếu nhận được MPI_BOTTOM thì kết thúc.
    /* Kết thúc */
    return 0;
}
```

☺ Lưu ý, bài trên có thể phát triển thành bài toán nhân hai ma trận!

2.2.3 Chương trình nhân ma trận và vector theo kiểu mẫu Workpool

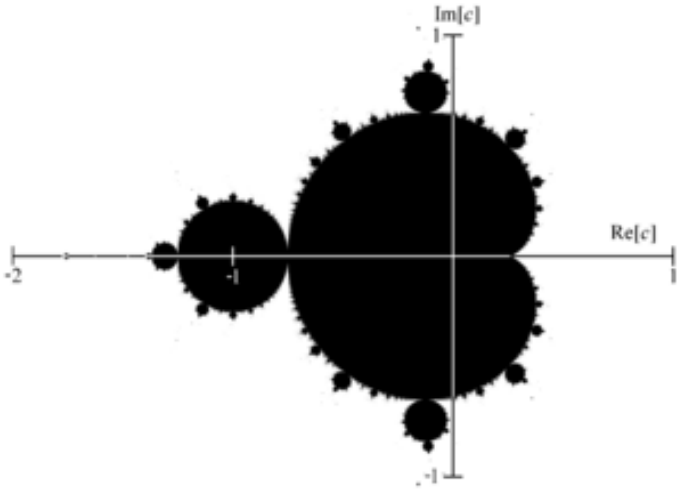
- Sinh viên hoàn tất source cho chương trình nhân ma trận với vector theo kiểu mẫu Workpool

3. Bài tập

3.1 Viết chương trình tính số π theo mô hình master/slave

3.2 Viết chương trình nhân hai ma trận theo mô hình workpool

3.3 SV tìm hiểu về hình Mandelbrot Set, viết chương trình MPI minh họa.


Im[c]
1
Re[c]
-2
-1
1
-1