Lab 5 Parallel Programming with MPI

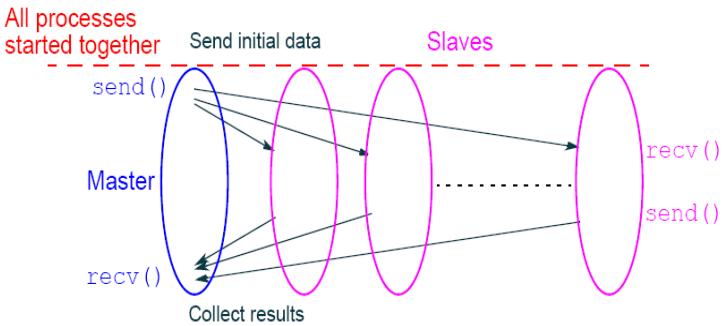
Master slave model

1 Mục tiêu

- SV tìm hiểu cách song song hóa bài toán theo mô hình master/slave
- 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

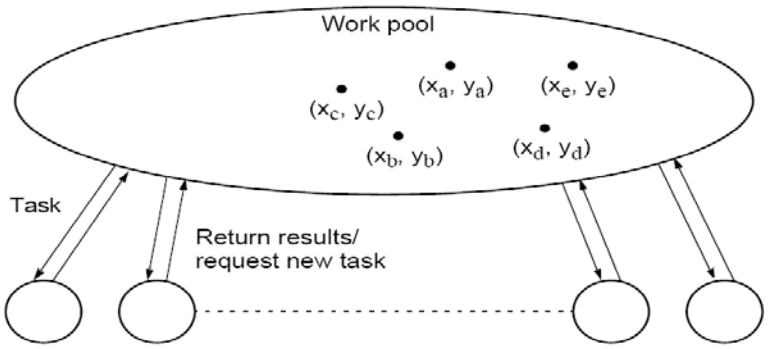
2.1 Master/slave model



Usual MPI approach

- Một process sẽ đóng vai trò master để phân phối công việc
- Các process còn lại sẽ giữ vai trò như slave và chỉ thực hiện công việc. Kết quả được trả về cho master như hình minh họa.
- Công việc thường được phân chia đều giữa các slave
- SV thảo luận về ưu và khuyết điểm của mô hình này.

2.2 Workpool model



- Một process đóng vai trò master để phân phối công việc
- Các process khác sẽ thực thi công việc và trả kết quả về. Process nào trả kết quả về trước sẽ được gán công việc tiếp theo.
- Khối lượng công việc được phân chia giữa các process là tùy thuộc sức mạnh của processor chạy process đó.
- SV thảo luận ưu khuyết điểm của mô hình này.

2.3 Chương trình minh họa

2.2.1 Một mô hình lập trình mẫu:

```
#include <mpi.h>
#include <stdio.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_code();
    else
      slave_code();

    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 một ma trận và một vector:

```
#include <mpi.h>
#include <stdio.h>

int master(int procs){
    long matrixA[N][N], vectorC[N];
    long i,j,dotp, sender, row, numsent=0;
    MPI_Status status;

      /* Initialize data */
    for(i=0; i < N; i++)
      for(j=0; j < N; j++)
          matrixA[i][j] = 1;
```

```
    /* distribute data to slave */
    for(i=1; i < minFunc(procs, N); i++)
    {
      MPI_Send(&matrixA[i-1][0], N, MPI_LONG, i, i, MPI_COMM_WORLD );
      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 = status.MPI_SOURCE;
      row    = status.MPI_TAG - 1;
      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 < 10; i++)
      fprintf(stdout,"%ld ",vectorC[i]);
    return 0;
}

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

int slave(){


    /* Cong viec cua slave */

      -  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 không còn dữ liệu thì kết thúc


    /* Kết thúc */


    return 0;

}
```

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

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.