Lab 6 Parallel Programming with MPI

Speedup

1 Mục tiêu

- SV viết chương trình tuần tự và song song cho ứng dụng nhân ma trận với vector
- SV thử nghiệm và đánh giá thời gian thực hiện của giải thuật tuần tự và song song.
- Nhận xét về kết quả và vẽ đồ thị minh họa.

2 Nội dung

Khái niệm về speedup liên quan đến một câu hỏi thường gặp khi phát triển chương trình song song trên hệ thống nhiều bộ xử lý (multiprocessors) là chương trình song song thực thi nhanh hơn bao nhiêu lần. Để thực hiện so sánh, giải pháp được đề ra là so sánh thời gian thực hiện của giải thuật tuần tự tốt nhất trên hệ thống đơn xử lý (single-processor) với giải thuật song song trên hệ thống multiprocessor.

2.1 Speedup

Hệ số speedup (S_p) là một đo đạc của hiệu suất liên quan và được định nghĩa như sau:

$$S_p = \frac{t_s}{t_p}$$

Trong đó:

t_s : Thời gian thực thi sử dụng hệ thống một bộ xử lý (giải thuật tuần tự tốt nhất)

t_p : Thời gian thực thi sử dụng hệ thống p bộ xử lý

☺ Lưu ý giá trị của S_p có thể lớn hơn p hay không ?

2.2 Efficiency

Trong một số trường hợp người phát triển ứng dụng song song cần đánh giá bao lâu các bộ xử lý được sử dụng cho quá trình tính toán, đại lượng đo đạc này được gọi là độ hiệu quả (efficiency - E). E được định nghĩa như sau:

$$E = \frac{t_s}{t_p * p}$$

Hoặc:

$$E = \frac{S(p)}{p} * 100\%$$

Vd: Độ hiệu quả E = 50% có nghĩa là các bộ xử lý được sử dụng một nữa thời gian cho quá trình tính toán trên tổng thời gian thực thi của chương trình.

2.3 Chương trình minh họa

2.3.1 SV viết chương trình tuần tự

- Ma trận nhân vector hoặc ma trận nhân ma trận
- Đo đạc thời gian của chương trình tuần tự.

2.3.2 Chương trình nhân ma trận và vector:

```
/*   SV su dung chuong trinh hoan chinh da thuc hanh trong bai lab 5, kết
 *   hợp đo thời gian thực thi bằng hàm: MPI_Wtime()
 */

#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;
}

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;

}
```

☺ SV thực hiện đo đạc với số lượng processor tăng dần, ghi nhận số liệu và vẽ đồ thị minh họa cho kết quả đo đạc speedup !

3 Bài tập

SV kết hợp đo đạc thời gian và tính speedup cho mỗi chương trình sau:

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.