Parallel Computing

Degree in Computer Science Engineering (ETSINF)

Year 2022-23 ◇ Partial exam 18/1/23 ◇ Block MPI ◇ Duration: 1h 45m

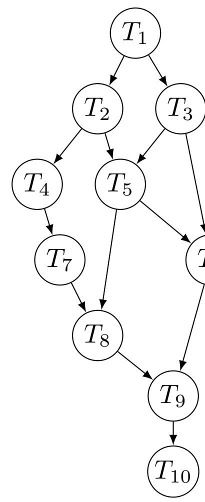
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Question 1 (1.2 points)

We want to parallelize the following code fragment using MPI with 2 processes. We assume that `n` is a predefined constant. In all calls, the second argument (the one following `n`) is input-output, which induces the task dependencies. The associated task dependency graph is provided. The computational cost of functions `f1`, `f2` and `f3` is $\frac{1}{3}n^3$ flops, n^3 flops and $2n^3$ flops, respectively.

```
double A[n][n], B[n][n], C[n][n], D[n][n],
       E[n][n], F[n][n];

f1(n,A);        /* Task T1 */
f2(n,D,A);      /* Task T2 */
f2(n,F,A);      /* Task T3 */
f2(n,B,D);      /* Task T4 */
f3(n,E,F,D);    /* Task T5 */
f3(n,C,E,F);    /* Task T6 */
f1(n,B);        /* Task T7 */
f2(n,E,B);      /* Task T8 */
f3(n,C,E,E);    /* Task T9 */
f1(n,C);        /* Task T10 */
```



0.9 p. (a) Implement a parallel version with MPI, using only point-to-point communication primitives. It can be assumed that initially all matrices contain equal valid values in all processes. The calculated matrices need not be collected in any of the processes. The chosen assignment should ensure, firstly, that the calculation is reasonably distributed between the two processes and, secondly, that communications between them are minimized.

Solution: To divide the calculation between the two processes, we will assign tasks T_2 and T_3 to different processes. We will also assign T_4 and T_7 to one process and T_5 to another (since the cost of T_4 and T_7 together is lower than that of T_5). In the same way, we will assign T_6 and T_8 to different processes.

To minimize communications, tasks T_2, T_4, T_7 and T_8 should be assigned to the same process, while the other would do tasks T_3, T_5 and T_6. In addition, task T_1 is replicated to avoid sending a message.

Accordingly, the assignment could be:

P_0: $T_1, T_2, T_4, T_7, T_8, T_9, T_{10}$.

P_1: T_1, T_3, T_5, T_6.

```
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

if (rank==0) {
  f1(n,A);     /* Task T1 */
  f2(n,D,A);   /* Task T2 */
  MPI_Send(D, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD);
  f2(n,B,D);   /* Task T4 */
  f1(n,B);     /* Task T7 */
  MPI_Recv(E, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  f2(n,E,B);   /* Task T8 */
  MPI_Recv(C, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  f3(n,C,E,E); /* Task T9 */
  f1(n,C);     /* Task T10 */
} else if (rank==1) {
  f1(n,A);     /* Task T1 */
  f2(n,F,A);   /* Task T3 */
  MPI_Recv(D, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

```
    f3(n,E,F,D); /* Task T5 */
    MPI_Send(E, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
    f3(n,C,F,F); /* Task T6 */
    MPI_Send(C, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
  }
```

0.3 p. (b) Calculate the sequential cost and parallel cost.

Solution: Sequential cost

$$t(n) = 3 \cdot \frac{1}{3}n^3 + 4 \cdot n^3 + 3 \cdot 2n^3 = 11n^3 \text{ flops}$$

Parallel cost

$$\begin{aligned} t_a(n,2) &= \tfrac{1}{3}n^3 + n^3 + 2n^3 + 2n^3 + 2n^3 + \tfrac{1}{3}n^3 = (7 + \tfrac{2}{3})n^3 \text{ flops} \\ t_c(n,2) &= 3(t_s + n^2 t_w) \\ t(n,2) &= t_a(n,2) + t_c(n,2) = (7 + \tfrac{2}{3})n^3 \text{ flops} + 3(t_s + n^2 t_w) \end{aligned}$$

Question 2 (1.2 points)

The following function implements the following expression $y = A * (A * x)$, where A is a square matrix of dimension N and x is a column vector of the same dimension.

```
void fun1(double A[N][N], double x[], double y[]) {
    int i,j;
    double z[N];

    for (i=0;i<N;i++) {
       z[i]=0.0;
       for (j=0;j<N;j++)
          z[i]+=A[i][j]*x[j];
    }
    for (i=0;i<N;i++) {
       y[i]=0.0;
       for (j=0;j<N;j++)
          y[i]+=A[i][j]*z[j];
    }
}
```

0.8 p. (a) Implement a parallel version using MPI, assuming that the input data is in process 0 and that the results must be complete in process 0 at the end of the execution. The problem size can be assumed to be a multiple of the number of processes.

Solution:

```
void fun1_par(double A[N][N], double x[], double y[]) {
    int p, np;
    int i,j;
    double zlcl[N];
    double Alcl[N][N];

    MPI_Comm_size(MPI_COMM_WORLD, &p);

    np = N/p;
    MPI_Scatter(A, np*N, MPI_DOUBLE, Alcl, np*N, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Bcast(x, N, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    for (i=0;i<np;i++) {
       zlcl[i]=0.0;
       for (j=0;j<N;j++)
          zlcl[i]+=Alcl[i][j]*x[j];
    }
    MPI_Allgather(zlcl, np, MPI_DOUBLE, y, np, MPI_DOUBLE, MPI_COMM_WORLD);
    for (i=0;i<np;i++) {
       zlcl[i]=0.0;
```

```
            for (j=0;j<N;j++)
               zlcl[i]+=Alcl[i][j]*y[j];
         }
         MPI_Gather(zlcl, np, MPI_DOUBLE, y, np, MPI_DOUBLE, 0, MPI_COMM_WORLD);
      }
```

0.4 p. (b) Calculate the parallel time expression, as well as the Speed Up and Efficiency. Calculate also the values of Speed Up and efficiency when the problem size (N) tends to infinity. Indicate separately the cost of each communication operation performed, as well as the arithmetic time.

Solution: An implementation of Allgather is assumed in which a Gather is performed on a process and then a Broadcast of length N is performed.

$$t(N) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 2 + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 2 = 2N^2 + 2N^2 = 4N^2$$

$$t(N,p) = t_{Scatter} + t_{Bcast} + t_{a1}(N,p) + t_{Allgather} + t_{Gather} + t_{a2}(N,p)$$

$$t_{Scatter} = (p-1)(t_s + \frac{N}{p}Nt_w)$$

$$t_{Bcast} = (p-1)(t_s + Nt_w)$$

$$t_{Allgather} = (p-1)(t_s + \frac{N}{p}t_w) + (p-1)(t_s + Nt_w)$$

$$t_{Gather} = (p-1)(t_s + \frac{N}{p}t_w)$$

$$t_a(N,p) = 2\sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 = \frac{4N^2}{p}$$

$$t(N,p) = (p-1)(t_s + \frac{N}{p}Nt_w) + (p-1)(t_s + Nt_w) + \sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 + (p-1)(t_s + \frac{N}{p}t_w) +$$

$$+(p-1)(t_s + Nt_w) + \sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 + (p-1)(t_s + \frac{N}{p}t_w)$$

$$t(N,p) \approx 5pt_s + (N^2 + 2pN + 2N)t_w + 4\frac{N^2}{p} = 5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p}$$

$$S(N,p) = \frac{t(N)}{t(N,p)} = \frac{4N^2}{5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p}}$$

$$\lim_{N\to\infty} S(N,p) = \lim_{N\to\infty} \frac{t(n)}{t(n,p)} = \frac{4}{t_w + \frac{4}{p}}$$

$$E(N,p) = \frac{S(N,p)}{p} = \frac{4N^2}{p(5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p})}$$

$$\lim_{N\to\infty} E(N,p) = \frac{4}{pt_w + 4}$$

Question 3 (1.1 points)

Let A be a matrix $A \in R^{n^2\times n^2}$ with blocks $A_i \in R^{n\times n}$, $i = 0, 1, \ldots, n-1$, located along the main diagonal of the matrix A, as shown in the following figure, all other elements being equal to 0:

$$A = \begin{pmatrix} A_0 & 0_{n\times n} & \cdots & 0_{n\times n} \\ 0_{n\times n} & A_1 & \cdots & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} & \ddots & \vdots \\ 0_{n\times n} & 0_{n\times n} & \cdots & A_{n-1} \end{pmatrix}.$$

0.9 p. (a) Implement an MPI function `copy_blocks`, using derived datatypes to reduce the number of messages sent. Assume that the array A is stored in process P_0 and must be partitioned among all processes, so that the local array B of process P_i must contain the array A_i, $i = 0, 1, \ldots, n-1$. Assume that the number of processes is greater than or equal to n.

```
void copy_blocks(double A[n*n][n*n], double B[n][n]){
        .........
}
```

Example of a matrix A of dimension 9×9 with 3 processes:

$$A = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 10 & 11 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & 13 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 16 & 17 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 18 & 19 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 21 & 22 & 23 \\ 0 & 0 & 0 & 0 & 0 & 0 & 24 & 25 & 26 \end{pmatrix}$$

$$B(P_0) = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}, B(P_1) = \begin{pmatrix} 9 & 10 & 11 \\ 12 & 13 & 14 \\ 15 & 16 & 17 \end{pmatrix}, B(P_2) = \begin{pmatrix} 18 & 19 & 20 \\ 21 & 22 & 23 \\ 24 & 25 & 26 \end{pmatrix}.$$

Solution:

```
void copy_blocks(double A[n*n][n*n], double B[n][n]){
  int i, rank;
  MPI_Status stat;
  MPI_Datatype ntype;
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Type_vector(n,n,n*n,MPI_DOUBLE,&ntype);
  MPI_Type_commit(&ntype);
  if (rank==0){
    for(i=1; i<n; i++)
      MPI_Send(&A[i*n][i*n], 1, ntype, i, 100, MPI_COMM_WORLD);
    MPI_Sendrecv(&A[0][0], 1, ntype, 0, 100, B, n*n, MPI_DOUBLE,
         0, 100, MPI_COMM_WORLD, &stat);
  }
  else if(rank<n)
    MPI_Recv(B, n*n, MPI_DOUBLE, 0, 100, MPI_COMM_WORLD, &stat);
  MPI_Type_free (&ntype);
}
```

0.1 p. (b) Calculate the communications time of the function `copy_blocks`.

Solution: Since process P_0 has to send block A_i to process P_i, $i = 1, \ldots, n-1$, the total number of messages will be $n-1$, and since $n \times n$ data are sent in each message, the communication time is

$$t_c = (n-1)(t_s + n^2 t_w).$$

0.1 p. (c) Calculate the communications time, assuming now that no derived datatypes have been used in point-to-point communications.

Solution: In this case, process P_0 has to send each of the n rows of A_i in separate messages to process P_i, $i = 1, \ldots, n-1$, thus the total number of messages will be $(n-1)n$, and since n data is sent in each message, the communication time is

$$t_c = (n-1)n(t_s + n t_w).$$