Parallel Computing
Degree in Computer Science Engineering (ETSInf)
Year 2021-22 ◇ 28/1/22 ◇ Block MPI ◇ Duration: 1h 45m

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Question 1 (1.3 points)

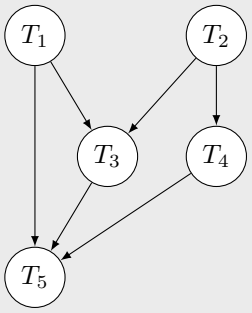
Given the following function, where `NR` and `NC` are integer constants:

```
void calculate(double x[NR], double y[NR]) {
  double A[NR][NC], B[NR][NC], v1[NR];
  double s,t;
  task1(A,y,x);
  s=task2(B);
  t=task3(y,B,x);
  task4(v1,s);
  task5(x,A,v1,t);
}
```

and taking into accout that function `task1` modifies its two first arguments, while the rest of functions (`task2` to `task5`) modifiy only its first argument.

0.3 p. (a) Draw the task dependency graph.

Solution: The task dependency graph is the following:



1 p. (b) Write a parallel version in MPI for two processes, assuming that the vectors `x` and `y` are initially available in all processes. The final correct content of vector `x` must be stored in process 0 and that of vector `y` in process 1.

The parallel version should:

- Use an assignment that maximizes the parallelism and minimizes the communication cost.
- Avoid possible deadlocks.

Solution:

The final result of `x` is computed at task T_5 and that of `y` in T_3. Given that `x` must be stored in P_0 and `y` in P_1, we will try that P_0 runs T_5 and P_1 runs T_3. Furthermore, we will try that tasks T_1 and T_5 are done by the same process, so that matrix `A` need not be communicated. The same happens with tasks T_2 and T_3, in this case for matrix `B`.

According to this, the optimal assignment would be $P_0 : T_1, T_4, T_5$; $P_1 : T_2, T_3$.

```
void calculate_mpi(double x[NR], double y[NR]) {
  double s, t;
```

```
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    if (rank==0) {
      double A[NR][NC], v1[NR];
      task1(A,y,x);
      MPI_Send(y,NR,MPI_DOUBLE,1,33,MPI_COMM_WORLD);
      MPI_Recv(&s,1,MPI_DOUBLE,1,33,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
      task4(v1,s);
      MPI_Recv(&t,1,MPI_DOUBLE,1,33,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
      task5(x,A,v1,t);
    }
    else if (rank==1) {
      double B[NR][NC];
      s=task2(B);
      /* To avoid deadlocks, we first receive and the send
         (the opposite to process 0) */
      MPI_Recv(y,NR,MPI_DOUBLE,0,33,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
      MPI_Send(&s,1,MPI_DOUBLE,0,33,MPI_COMM_WORLD);
      t=task3(y,B,x);
      MPI_Send(&t,1,MPI_DOUBLE,0,33,MPI_COMM_WORLD);
    }
  }
```

Question 2 (1.2 points)

The following function computes the product $A \times x$ in y and then scales y in such a way that the largest element in absolute value becomes 1.

```
void vector( double y[M], double A[M][N], double x[N] )
{ int i,j;
  double a,ma;

  ma = 0;
  for ( i = 0 ; i < M ; i++ ) {
    a = 0;
    for ( j = 0 ; j < N ; j++ )
      a += A[i][j] * x[j];
    y[i] = a;
    if ( a < 0 ) a = -a;
    if ( a > ma ) ma = a;
  }
  for ( i = 0 ; i < M ; i++ )
    y[i] /= ma;
}
```

1 p. (a) Parallelize it with MPI in a way that the workload of the loops gets distributed among all available processes. Use collective communication operations whenever possible. The input arguments of the function only have valid values in process 0. We want the result (vector y) to be in process 0. We assume that `M` and `N` are an exact multiple of the number of processes.

Solution:

```
  void vector( double y[M], double A[M][N], double x[N] )
  { int i,j, np,nb;
```

```
  double a,ma, Al[M][N],yl[M],mal;

  MPI_Comm_size(MPI_COMM_WORLD,&np);

  nb = M / np;
  MPI_Scatter(A,nb*N,MPI_DOUBLE,Al,nb*N,MPI_DOUBLE,0,MPI_COMM_WORLD);
  MPI_Bcast(x,N,MPI_DOUBLE,0,MPI_COMM_WORLD);

  mal = 0;
  for ( i = 0 ; i < nb ; i++ ) {
    a = 0;
    for ( j = 0 ; j < N ; j++ )
      a += Al[i][j] * x[j];
    yl[i] = a;
    if ( a < 0 ) a = -a;
    if ( a > mal ) mal = a;
  }

  MPI_Allreduce(&mal,&ma,1,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD);

  for ( i = 0 ; i < nb ; i++ )
    yl[i] /= ma;

  MPI_Gather(yl,nb,MPI_DOUBLE,y,nb,MPI_DOUBLE,0,MPI_COMM_WORLD);
}
```

0.2 p. (b) Indicate the communication cost of each communication operation that you have used, assuming a simple implementation of the communications.

Solution: $t_{scat} = (p-1) * (t_s + M/pNt_w)$
$t_{bcast} = (p-1) * (t_s + Nt_w)$
$t_{allred} = 2 * (p-1) * (t_s + t_w)$
$t_{gat} = (p-1) * (t_s + M/pt_w)$

Question 3 (1 point)

We have an MPI program in which two processes, P_0 and P_1, must communicate certain elements of a matrix of double precision real numbers, represented by a bidimensional array `A` in the sending process and `B` in the receiving process. The matrix algorithm requires sending the elements of the main antidiagonal (except the first one) together with the adjacent elements shown in the figure, marked as a and b, respectively, in matrix `A`, in such a way that the receiving process must store these elements shifted one row upwards, that is, with values b occupying the main antidiagonal and the values a occypying the adjacent elements shown in the figure, as indicated in matrix `B`.

$$\mathtt{A} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & a & b \\ \cdot & \cdot & \cdot & a & b & \cdot \\ \cdot & \cdot & a & b & \cdot & \cdot \\ \cdot & a & b & \cdot & \cdot & \cdot \\ a & b & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \qquad \mathtt{B} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & a & b \\ \cdot & \cdot & \cdot & a & b & \cdot \\ \cdot & \cdot & a & b & \cdot & \cdot \\ \cdot & a & b & \cdot & \cdot & \cdot \\ a & b & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

0.9 p. (a) Write the necessary code to carry out the communication (sending from P_0 and reception at P_1) using a single message. It is compulsory to use derived data types. The following function header should be used:

```
void comunicate (double A[N][N], double B[N][N])
```

Solution: To perform the communication with a single message, it is necessary to define a new MPI derived data type. Once this is done, we must simply send an element of this new type, using the appropriate pointers of the send and receive buffers, in such a way that values are read and written at the required locations.

```
void comunicate (double A[N][N], double B[N][N]) {
int rank;
MPI_Datatype diags;
MPI_Type_vector(N-1, 2, N-1, MPI_DOUBLE, &diags);
MPI_Type_commit(&diags);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank==0) {
  MPI_Send(&A[1][N-2], 1, diags, 1, 0, MPI_COMM_WORLD);
} else if (rank==1) {
  MPI_Recv(&B[0][N-2], 1, diags, 0, 0, MPI_COMM_WORLD,
           MPI_STATUS_IGNORE);
}
MPI_Type_free(&diags);
}
```

0.1 p.

(b) Indicate which is the communication cost.

Solution: Just one message is sent, whose length is `2*(N-1)`. Therefore, the cost will be

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