Computació Paral·lela
Grau en Enginyeria Informàtica (ETSInf)
Curs 2021-22 ◇ 28/1/22 ◇ Bloc MPI ◇ Duració: 1h 45m

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Qüestió 1 (1.3 punts)

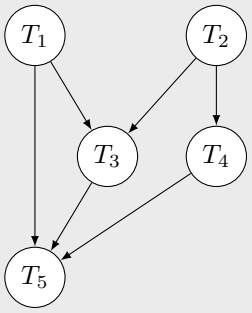
Donada la següent funció, on `NR` i `NC` són constants enteres:

```
void calcular(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);
}
```

i tenint en compte que la funció `task1` modifica els seus dos primers arguments, mentres que la resta de funcions (`task2` a `task5`) modifiquen només el seu primer argument.

0.3 p. (a) Dibuixa el graf de dependències de les diferents tasques.

Solució: El graf de dependències és el següent:



1 p. (b) Escriu una versió paral·lela MPI de la funció anterior per a dos processos, suposant que els vectors `x` i `y` estan inicialment disponibles en tots els processos. El contingut final correcte del vector `x` deu quedar en el procés 0 i el del vector `y` en el procés 1.

La versió paral·lela deurà:

- Usar una assignació que maximitze el paral·lelisme i minimitze el cost de comunicacions.
- Evitar possibles interbloquejos.

Solució:

El resultat final de `x` es calcula en la tasca T_5 i el de `y` en T_3. Donat que `x` deu quedar en P_0 i `y` en P_1, intentarem que P_0 faça T_5 i P_1 faça T_3. A més, intentarem que les tasques T_1 i T_5 les faça el mateix procés, per a no tindre que comunicar la matriu `A`. El mateix ocorre amb les tasques T_2 i T_3, en aquest cas per la matriu `B`.

D'acord amb açò, l'assignació òptima seria $P_0 : T_1, T_4, T_5$; $P_1 : T_2, T_3$.

```
void calcular_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);
      /* Per a evitar interbloquejos, primer rebem i després enviem
         (al revés que en el procés 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);
    }
  }
```

Qüestió 2 (1.2 punts)

La següent funció calcula el producte $A \times x$ en y i després escala y de forma que l'element de major valor absolut siga 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) Paral·lelitza-la amb MPI de forma que el treball dels bucles quede repartit entre tots els processos disponibles. Utilitza operacions de comunicació col·lectiva on siga possible. Els paràmetres d'entrada a la funció sols tenen valors vàlids en el procés 0. Volem el resultat (el vector y) en el procés 0. Assumim que `M` i `N` són un múltiple exacte del nombre de processos.

Solució:

```
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) Indica el cost de comunicacions de cada operació de comunicació que hages utilitzat, suposant una implementació senzilla de les comunicacions.

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

Qüestió 3 (1 punt)

Tenim un programa MPI en el que dos processos, P_0 i P_1, deuen comunicar-se determinats elements de una matriu de nombres reals de doble precisió, representada per un array bidimensional `A` en el procés emisor i `B` en el receptor. L'algoritme matricial requerix que s'envien els elements de la antidiagonal principal (excepte el primer) junt amb els elements adjacents mostrats en la figura, marcats com a i b, respectivament, en la matriu `A`, de forma que el procés receptor deu emmagatzemar estos elements desplaçats una filera cap amunt, és a dir, amb els valors b ocupant la antidiagonal principal i els valors a ocupant els elements adjacents mostrats en la figura, com s'indica en la matriu `B`.

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

0.9 p. (a) Escriu el codi necessari per a realitzar la comunicació (enviament des de P_0 i recepció en P_1) utilitzant un únic missatge. És obligatori l'ús de tipus derivats. Es deurà usar la següent capçalera de funció:

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

Solució: Per a realitzar la comunicació en un únic missatge, és necessari definir un nou tipus de dades derivat de MPI. Una volta hem fet açò, simplement hi ha que enviar un element d'aquest tipus, usant els punters apropiats per al buffer de enviament i recepció, de forma que es lligen i escriguen els valors en les posicions requerides.

```
void comunica (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) Indica quin és el cost de comunicació.

Solució: S'envia un únic missatge, la longitud del qual és 2∗(N-1). Per tant, el cost serà

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