Computación Paralela
Grado en Ingeniería Informática (ETSInf)
Curso 2016-17 ◇ Examen parcial 16/1/2017 ◇ Duración: 1h 30m

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Cuestión 1 (1 punto)

Se pretende distribuir con MPI los bloques cuadrados de la diagonal de una matriz cuadrada de dimensión $3 \cdot \text{DIM}$ entre 3 procesos. Si la matriz fuera de dimensión 6 (DIM=2), la distribución sería como se ejemplifica:

$$\begin{pmatrix} a_{00} & a_{01} & \dots & \dots & \dots & \dots \\ a_{10} & a_{11} & \dots & \dots & \dots & \dots \\ \dots & \dots & a_{22} & a_{23} & \dots & \dots \\ \dots & \dots & a_{32} & a_{33} & \dots & \dots \\ \dots & \dots & \dots & \dots & a_{44} & a_{45} \\ \dots & \dots & \dots & \dots & a_{54} & a_{55} \end{pmatrix} \rightarrow \begin{array}{ll} P_0 & \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \\ P_1 & \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix} \\ P_2 & \begin{bmatrix} a_{44} & a_{45} \\ a_{54} & a_{55} \end{bmatrix} \end{array}$$

Realiza una función que permita enviar los bloques con el mínimo número de mensajes. Se proporciona la definición de la cabecera de la función para facilitar la implementación. El proceso 0 tiene en `A` la matriz completa y tras la llamada a la función se debe tener en `Alcl` de cada proceso el bloque que le corresponde. Utiliza primitivas de comunicación punto a punto.

```
void SendBAD(double A[3*DIM][3*DIM], double Alcl[DIM][DIM]) {
```

Solución:

```
void SendBAD(double A[3*DIM][3*DIM], double Alcl[DIM][DIM]) {
  int i,rank,p,m,n;
  MPI_Datatype DiagBlock;

  MPI_Comm_size(MPI_COMM_WORLD, &p);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  n = 3*DIM;
  m = DIM;

  if (rank == 0) {
    MPI_Type_vector(m, m, n, MPI_DOUBLE, &DiagBlock);
    MPI_Type_commit(&DiagBlock);
    MPI_Sendrecv(A, 1, DiagBlock, 0, 0, Alcl, m*m, MPI_DOUBLE, 0, 0,
                 MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    for (i=1;i<p;i++)
      MPI_Send(&A[i*m][i*m], 1, DiagBlock, i, 0, MPI_COMM_WORLD);
    MPI_Type_free(&DiagBlock);
  } else {
    MPI_Recv(Alcl, m*m, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  }
}
```

Cuestión 2 (1 punto)

El siguiente programa lee una matriz cuadrada A de orden N y construye, a partir de ella, un vector v de dimensión N de manera que su componente i-ésima, $0 \leq i < N$, es igual a la suma de los elementos de la fila i-ésima de la matriz A. Finalmente, el programa imprime el vector v.

```
int main(int argc, char *argv[])
{
  int i,j;
  double A[N][N],v[N];
  read_mat(A);
  for (i=0;i<N;i++) {
    v[i] = 0.0;
    for (j=0;j<N;j++)
      v[i] += A[i][j];
  }
  write_vec(v);
  return 0;
}
```

0.75 p. (a) Utiliza comunicaciones colectivas para implementar un programa MPI que realice el mismo cálculo, de acuerdo con los siguientes pasos:

- El proceso P_0 lee la matriz A.
- P_0 reparte la matriz A entre todos los procesos.
- Cada proceso calcula la parte local de v.
- P_0 recoge el vector v a partir de las partes locales de todos los procesos.
- P_0 escribe el vector v.

Nota: Para simplificar, se puede asumir que N es divisible entre el número de procesos.

Solución:

```
int main(int argc, char *argv[])
{
  int i,j,id,p,tb;
  double A[N][N],Al[N][N],v[N],vl[N];
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &id);
  MPI_Comm_size(MPI_COMM_WORLD, &p);
  if (id==0) {
    read_mat(A);
  }
  tb = N/p;
  MPI_Scatter(A, tb*N, MPI_DOUBLE, Al, tb*N, MPI_DOUBLE, 0, MPI_COMM_WORLD);
  for (i=0;i<tb;i++) {
    vl[i] = 0.0;
    for (j=0;j<N;j++)
      vl[i] += Al[i][j];
  }
  MPI_Gather(vl, tb, MPI_DOUBLE, v, tb, MPI_DOUBLE, 0, MPI_COMM_WORLD);
  if (id==0) {
    write_vec(v);
  }
  MPI_Finalize();
```

```
    return 0;
  }
```

0.25 p. (b) Calcula los tiempos secuencial y paralelo, sin tener en cuenta las funciones de lectura y escritura. Indica el coste que has considerado para cada una de las operaciones colectivas realizadas.

Solución: Tiempo secuencial:

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

Tiempo paralelo aritmético con p procesos:

$$t_{arit}(N,p) = \sum_{i=0}^{N/p-1} \sum_{j=0}^{N-1} 1 = \frac{N^2}{p} \text{ flops}$$

Tiempo paralelo de comunicaciones con p procesos:

- Reparto de la matriz A:
$$(p-1)\left(t_s + \frac{N^2}{p} t_w\right)$$
- Recogida del vector v:
$$(p-1)\left(t_s + \frac{N}{p} t_w\right)$$

Por lo tanto, el tiempo paralelo es

$$t(N,p) = \frac{N^2}{p} \text{ flops} + (p-1)\left(2t_s + \frac{N}{p}(N+1)t_w\right) \approx \frac{N^2}{p} \text{ flops} + 2pt_s + N^2 t_w$$

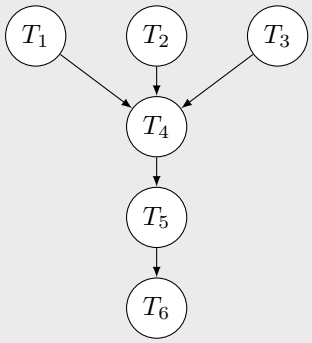
Cuestión 3 (1 punto)

Dada la siguiente función, donde suponemos que las funciones `T1`, `T2` y `T3` tienen un coste de $7n$ y las funciones `T5` y `T6` de n, siendo n un valor constante.

```
double ejemplo(int val[3])
{
  double a,b,c,d,e,f;
  a = T1(val[0]);
  b = T2(val[1]);
  c = T3(val[2]);
  d = a+b+c;       /* T4 */
  e = T5(val[2],d);
  f = T6(val[0],val[1],e);
  return f;
}
```

0.25 p. (a) Dibuja el grafo de dependencias y calcula el coste secuencial.

Solución: El grafo de dependencias se muestra en la siguiente figura:



El coste secuencial es: $t(n) = 7n + 7n + 7n + 2 + n + n \approx 23n$

0.5 p.

(b) Paralelízala usando MPI, suponiendo que hay tres procesos. Todos los procesos invocan la función con el mismo valor del argumento `val` (no es necesario comunicarlo). El valor de retorno de la función debe ser correcto en el proceso 0 (no es necesario que lo sea en los demás procesos).
Nota: para las comunicaciones deben utilizarse únicamente operaciones de comunicación colectiva.

Solución: Las tres primeras tareas se deben asignar a procesos distintos, para repartir la carga. Las otras tres tareas se han de ejecutar en el orden secuencial (debido a las dependencias), por lo que las asignamos a P_0, ya que este proceso es el que debe almacenar el valor correcto de retorno de la función.

```
double ejemplo(int val[3])
{
  double a,b,c,d,e,f,operand;
  int rank;
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);

  switch (rank) {
    case 0: a = T1(val[0]); operand = a; break;
    case 1: b = T2(val[1]); operand = b; break;
    case 2: c = T3(val[2]); operand = c; break;
  }
  MPI_Reduce(&operand, &d, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
  if (rank==0) {
    e = T5(val[2],d);
    f = T6(val[0],val[1],e);
  }
  return f;
}
```

0.25 p.

(c) Calcula el tiempo de ejecución paralelo (cálculo y comunicaciones) y el speedup con tres procesos. Obtén también el speedup asintótico, es decir, el límite cuando n tiene a infinito.

Solución: El tiempo paralelo se calcula a partir del coste asociado al camino crítico del grafo de dependencias, correspondiente a $T_1 - T_4 - T_5 - T_6$, por ejemplo. Para el coste de comunicación, consideramos que la operación de reducción internamente enviará únicamente dos mensajes, cada uno de ellos de longitud igual a 1 `double` (en la reducción también se realizarán 2 flops, que no se han incluido en las expresiones).

$$t(n,3) = t_{arit}(n,3) + t_{comm}(n,3)$$
$$t_{arit}(n,3) = 7n + n + n \approx 9n$$

$$t_{comm}(n,3) = 2 \cdot (t_s + t_w)$$

$$t(n,3) \approx 9n + 2t_s + 2t_w$$

El speedup será:

$$S(n,3) = \frac{t(n)}{t(n,3)} \approx \frac{23n}{9n + 2t_s + 2t_w}$$

Como en este caso el coste de comunicación es muy bajo, asintóticamente (cuando n crece) el speedup tiende al valor $S(n,3) \to \frac{23n}{9n} = 2,56$.