Relación de Ejercicios

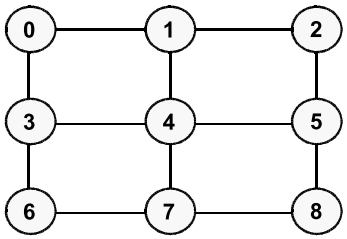
Programación Distribuida y Paralela.
4º de Ingeniería Superior en Informática.
Departamento de Lenguajes y Sistemas Informáticos

TEMA 1. INTRODUCCIÓN A LA PROGR. DISTRIBUIDA Y PARALELA

1. Describir una solución eficiente para realizar una operación de reducción global con 2^n tareas (suma, máximo, producto, etc.) suponiendo dos casos: a) que la solución debe obtenerse sólo en una de las tareas, b) que la solución debe obtenerse en todas las tareas. Describir la estructura de comunicación así como las operaciones que realiza cada tarea y evaluar el tiempo de comunicación y computación que requiere el algoritmo.

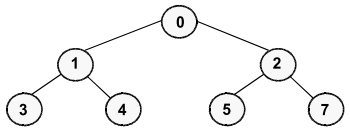
2. Describir gráficamente cómo se podría implementar una operación colectiva `MPI_Gather` siendo raíz el proceso 0 en un hipercubo de dimensión 2 (con 4 procesadores). Obtener una fórmula de tiempo de ejecución para esta operación en función del tamaño del bloque que aporta cada procesador N y los parámetros de comunicación t_s (latencia) y t_w (ancho de banda) de la arquitectura. Intentar extender la fórmula para un hipercubo de cualquier dimensión.

3. Describir gráficamente cómo se podría implementar de forma eficiente una operación tipo `MPI_Broadcast` de un valor entero sobre 9 procesadores conectados entre sí con una topología de interconexión malla cuadrada 3×3, asumiendo que el procesador 0 es el procesador raíz del broadcast. (véase la figura de abajo). Para ello, mostrar en qué instante de tiempo se realizan los diferentes pasos de comunicación, numerando con instantes de tiempo $(1, 2, 3, \ldots)$ los enlaces donde se produce la comunicación. Obtener una fórmula de tiempo de ejecución para esta operación, sobre 9 procesadores, en función de los parámetros de comunicación t_s (latencia) y t_w (ancho de banda para envío de enteros) de la arquitectura. Extender la fórmula para el caso en que P es un entero al cuadrado ($P = k^2$, con k entero positivo) y la malla tiene k filas y k columnas.



4. Describir gráficamente cómo se podría implementar de forma eficiente una operación colectiva `MPI_Gather` siendo raíz el proceso 0 sobre $P = 7$ procesadores conectados con una red de interconexión con topología de árbol binario (véase la figura de abajo). Se asume que cada procesador aporta un bloque de N elementos enteros. Para ello, mostrar en qué instante de tiempo se realizan los diferentes pasos de comunicación, numerando con instantes de tiempo los enlaces donde se produce la comunicación $(1, 2, 3, \ldots)$ e indicando también el tamaño de

los mensajes enviados en cada instante. Obtener una fórmula de tiempo de ejecución para esta operación, sobre 7 procesadores, en función del tamaño del bloque que aporta cada procesador N y los parámetros de comunicación t_s (latencia) y t_w (ancho de banda para envío de enteros) de la arquitectura. Extender la fórmula para el caso en que P es una potencia de 2 menos 1 ($P = 2^k - 1$, con k entero positivo).



5. Dada una lista de n números $x_0, \ldots, x_{n-1}$, se desea implementar la operación de prefijo que calcula todas las sumas parciales repartiendo los resultados en n procesos: $(x_0, P_0), (x_1 + x_0, P_1), \ldots, (x_{n-1} + \ldots + x_1 + x_0, P_{n-1})$. Derivar una solución paralela a este problema basada en una estructura de hipercubo. Evaluar el tiempo de computación y comunicación del algoritmo.

6. Describir un algoritmo basado en hipercubo para resolver el problema de encontrar todos los ceros de una lista de enteros. Ilustrar el algoritmo para el caso de $P = 4$, $P = 8$ y para 32 números. Evaluar el tiempo de computación y comunicación del algoritmo en base a parámetros de la arquitectura y el tamaño del problema.

7. Describir un algoritmo para resolver el problema de calcular si el número de ceros y de unos de una secuencia binaria es par o impar (tanto para el número de unos como para el número de ceros). Ilustrarlo con una secuencia de $N = 21$ bits para $P = 4$ y $P = 8$. Suponer que se desea que el resultado se obtenga en todos los procesadores que intervienen en la computación. Modelar el tiempo de ejecución del algoritmo, suponiendo que el número de procesadores es una potencia de 2, en función del tamaño de la secuencia N y los parámetros de comunicación t_s (latencia) y t_w (ancho de banda) de la arquitectura. ¿Cómo se podría realizar esta operación en MPI? Esbozar el código MPI que resolvería este problema.

8. Se dispone de un algoritmo paralelo para calcular una aproximación a la integral de una función f en un intervalo $[a, b]$ sobre $P = 2^k$ procesadores (k es un número entero positivo). Se asume que todos los valores numéricos, sean enteros o reales, ocupan una palabra de memoria. Se desea modelar el tiempo de ejecución de cada una de las fases del algoritmo. El algoritmo consta de las siguientes fases que se ejecutan en orden:

   a) Un procesador distinguido raíz selecciona un entero N, múltiplo de P y difunde el valor N a todos los demás procesadores utilizando un esquema de difusión en árbol binario. Obtener una fórmula del tiempo de ejecución paralelo de esta fase en función de N, P, t_s y t_w. Obtener inicialmente la fórmula para $P = 4$, describiendo los pasos de comunicación gráficamente y extender después la fórmula para $P = 2^k$.

   b) Cada procesador P_j selecciona N/P subintervalos disjuntos del intervalo $[a, b]$ y calcula una aproximación a la integral en dichos subintervalos utilizando la regla del trapecio y calcula la suma local de las N/P aproximaciones calculadas. Se supone que el costo asociado al cálculo de la integral en un subintervalo es tc_1 y el costo de la suma aritmética de dos números reales es tc_2. Obtener el tiempo de ejecución paralelo de este paso en función de N, P, tc_1 y tc_2.

c) La suma de las aproximaciones calculadas por cada procesador individual es obtenida en todos los procesadores mediante una operación de reducción todos a todos que sigue un esquema de hipercubo. Obtener una fórmula de tiempo de ejecución paralelo de este paso en función de N, P, t_s , t_w y tc_2. Ilustrar cómo se realizaría la operación de suma distribuida para el caso de $P = 8$.

9. Disponemos de un algoritmo paralelo para multiplicar una matriz real A de tamaño $N \times N$ por un vector x de dimensión N obteniendo un vector $y = A * x$. Supondremos que:

   - Disponemos de $P = 7$ procesadores (P_0, P_1,...,P_6) conectados de acuerdo a una topología de árbol binario con 3 niveles y la dimensión de la matriz N es un múltiplo de 7.
   - Tanto la matriz A como el vector x se encuentran inicialmente en uno de los procesadores P_0.

   El algoritmo procede de acuerdo a los siguientes pasos:

   (a) Desde P_0 se difunde la dimensión de la matriz (N) a todos los procesadores siguiendo un esquema de difusión en árbol binario.
   (b) El procesador P_0 reparte bloques de $\frac{N}{P}$ filas consecutivas de la matriz A entre todos los procesadores. Para ello, se sigue un esquema de reparto en árbol, en el que el procesador P_0 se queda las primeras $\frac{N}{P}$ filas de la matriz, parte el resto de la matriz en dos trozos iguales y los envía a sus dos hijos en el árbol, que harán exactamente lo mismo con la porción de matriz que les corresponde. Los nodos hojas se limitan a recibir un bloque de $\frac{N}{P}$ filas de la matriz. Finalmente, como resultado de este proceso, cada procesador P_i mantendrá $\frac{N}{P}$ filas consecutivas de la matriz A.
   (c) El procesador P_0 reparte equitativamente el vector x entre los procesadores, de tal forma que a cada procesador le corresponde un subvector de x con $\frac{N}{P}$ elementos consecutivos de dicho vector. Este reparto sigue el mismo esquema que se utiliza para repartir bloques de la matriz A.
   (d) Cada procesador multiplica localmente su bloque de filas por su subvector de x, y obtiene un subvector con $\frac{N}{P}$ elementos consecutivos del vector resultado y.
   (e) P_0 recolecta los subvectores de y siguiendo un procedimiento inverso al seguido en 3.

   Se desea lo siguiente:

   - Obtener una expresión para el tiempo de computación del algoritmo paralelo en función de N y t_c. (t_c = tiempo necesario para una suma y una multiplicación simple).
   - Obtener expresiones para las siguientes fracciones del tiempo de comunicación del algoritmo paralelo en función de N, t_s y t_w:
     (a) Difusión de la dimensión de la matriz (N) entre los procesadores.
     (b) Reparto equitativo de los bloques de filas de la matriz A entre los procesadores.
     (c) Reparto equitativo de los elementos del vector x entre los procesadores.
     (d) Recolección de los subvectores del vector y.
   - Cuál sería el tiempo de ocio de este algoritmo. Razonar la respuesta.
   - Obtener el tiempo de ejecución del algoritmo en función de N, t_s, t_w y t_c Obtener también expresiones para el speedup y la eficiencia.

TEMA 2. METODOLOGÍA DE PROGRAMACIÓN PARALELA

1. Se desea paralelizar una función vectorial que tiene como entrada un vector de reales y de dimensión N ($y = (y_1, y_2, ..., y_N)$) y devuelve como salida otro vector real dy también de dimensión N que se calcula en como:

   $$dy_i = (y_{i-1} + y_i * y_{i+1} - yi + 2)/8$$

   donde los valores del lado derecho que entran fuera del rango se determinan cíclicamente como:

   $$y_0 = y_N, y_{N+1} = y_1, y_{N+2} = y_2$$

   Establecer la descomposición de tareas así como la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas. Definir una estrategia de asignación eficiente sobre 4 y 8 procesadores. Cómo se distribuyen el vector de entrada y el vector solución entre los procesadores con la solución que se propone.

2. Se desea paralelizar un programa secuencial que procesa una malla bidimensional X de 1000×1000 puntos, actualizándola de acuerdo al siguiente esquema iterativo:

   $X_{i,j}^{t+1} = (4X_{i,j}^t + Xi + 1, j + 1^t + X_{i-1,j-1}^t + X_{i+1,j-1}^t + X_{i-1,j+1}^t)/8$

   Además, en cada iteración se calcula el máximo de los puntos situados en cada diagonal de la malla para chequear convergencia. Establecer la descomposición en tareas, definiendo la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas. Establecer una estrategia de asignación adecuada para resolver el problema sobre 5 procesadores.

3. Dada una matriz bidimensional de tamaño 12×7 indicar gráficamente, cómo se distribuiría dicha matriz cuando en cada caso:

   a) Cuando se utiliza un distribución por bloques de columnas y se dispone de tres procesos. ¿Qué parámetros definen esta distribución cíclica por bloques?

   b) Cuando se utiliza una distribución cíclica por bloques de tamaño 4×2 entre una malla con 2×3 de procesos.

4. Se desea resolver en paralelo un problema de exploración exhaustiva de un árbol de búsqueda. Se supone la existencia de las siguientes funciones:

   Función $Ramifica(nodo, hijo1, hijo2)$ para generar dos nuevos nodos del árbol, $hijo1$ e $hijo2$ que cuelgan de nodo en el árbol.

   Función lógica $Solucion(nodo, peso)$ que devuelve Verdadero si un nodo del árbol es una solución y, Falso en caso contrario. Si el nodo es solución, también devuelve un valor entero $peso$.

   El objetivo es obtener la suma de los pesos de todos los nodos solución del árbol.

   Estudiar las opciones más apropiadas para la descomposición y asignación de tareas del número de procesadores P y del coste de evaluación de las funciones $Ramifica$ y $Solucion$.

5. Supongamos que deseamos derivar un programa paralelo para ordenar N enteros $N = 2^k, k > 20$ sobre una arquitectura paralela de memoria distribuida con $P = 8$ procesadores, teniendo

en cuenta que el resultado de la ordenación debe replicarse en todos los procesadores. Supongamos también que en la máquina destino se dispone de un procedimiento secuencial $mezcla(lista1, lista2, listamezclada)$ que permite mezclar eficientemente dos secuencias ordenadas de enteros de cualquier longitud y de un procedimiento secuencial $ordena(lista)$ que permite ordenar una secuencia de enteros arbitraria.

Diseñar el programa paralelo de ordenación, justificando de forma concisa las decisiones tomadas. Suponer que la descomposición del problema en tareas ya ha sido realizado dando lugar a una tarea por cada par de elementos disjuntos de la lista a ordenar. Cada una de las tareas se encarga de ordenar el par que se le ha asignado. Establecer la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas. Definir una estrategia de asignación eficiente sobre los 8 procesadores.

6. Se desea encontrar todas las ocurrencias de una subcadena particular con m caracteres, llamado el patrón, dentro de otra cadena, llamada el texto con caracteres (N es mucho mayor que m y N es múltiplo de m). Tanto el patrón como el texto se almacenan internamente como arrays de caracteres. Describe un algoritmo paralelo basado en paso de mensajes para resolver el problema, en cada una de estas situaciones:

   - Que el algoritmo se ejecutará sobre un multicomputadorcon P procesadores ($P > 2$) de idéntica potencia y características.
   - Que el algoritmo se ejecutará en obre un multicomputador con P procesadores ($P > 2$) de distinta potencia y características.
   - Esbozar un enfoque paralelo para evaluar un polinomio de la forma

     $$a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

     para cualquier grado n, donde los valores a_i, n y el valor de x donde se ha de evaluar el polinomio son las entradas al problema.

     Razonar la solución considerando los diferentes enfoques de descomposición y asignación que son aplicables.

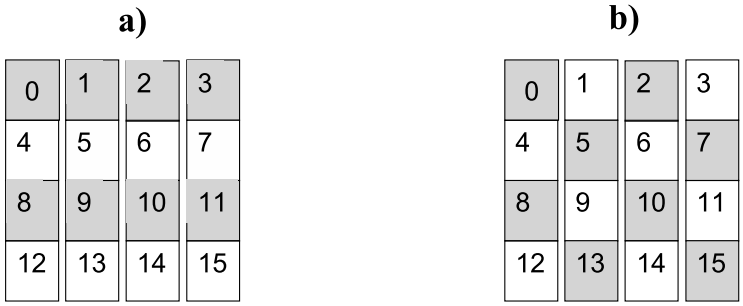
7. Describir cómo se podría realizar la descomposición de un cálculo para determinar si el número de 1's en una secuencia binaria de gran tamaño, almacenada en un array, es par o impar (chequeo de paridad).

8. Describir una solución paralela para encontrar el primer elemento en una lista de enteros almacenada en un vector que cumpla una determinada condición.

9. Esbozar un algoritmo paralelo basado en paso de mensajes para filtrar una secuencia de números naturales. El algoritmo recibe como entrada una secuencia de números naturales, finalizada en un número entero negativo, y sólo devuelve aquellos números de la secuencia que son múltiplos de 2, 3, 5, 7 y 9.

TEMA 3: NOTACIONES DE PROGRAMACIÓN PARALELA

1. Dada la malla de procesos que se muestra a continuación, en la que cada proceso aparece con su rango en el comunicador `MPI_COMM_WORLD`. Escribir una única llamada a `MPI_Comm_Split` que permita obtener un nuevo comunicador (`new_comm`) que incluya a los procesos que aparecen con fondo gris en la figura pero cuyos rangos se ordenen de forma inversamente proporcional al rango que tenían en comunicador original. Utilizar el menor número de órdenes posible.



2. Escribir un programa en MPI que implemente un "ping-pong", esto es, un programa en el que un proceso envía un mensaje a otro y éste último lo devuelve inmediatamente al primero. Utilizar la función `MPI_Wtime` para calcular cuánto tiempo se invierte en esta operación.

3. Utilizando las operaciones punto-a-punto de MPI, implementar la función `Bcast(int source, char *msg, int tam)` que ejecutarán todos los procesos de un anillo para realizar una operación de broadcast. ¿Cuántas iteraciones invierte la función?

4. Escribir un programa MPI que calcule el producto de dos matrices $C = A * B$ siguiendo una estrategia maestro/esclavo, esto es: el proceso raíz (el maestro) envía la matriz B mediante broadcast al resto de procesos (los esclavos); a continuación el maestro envía, utilizando operaciones punto-a-punto, bloques de filas de la matriz A a los esclavos para que éstos calculen los bloques de la matriz resultante y devuelvan el resultado al maestro. ¿Varía el tiempo de ejecución en función del tamaño del bloque?

5. Escribir un programa OpenMP que calcule la media de un vector de enteros en paralelo usando varias hebras.

6. Considera el siguiente bucle:

```
for (i = 0; i < n; i++)
    f(i);
```

   en el que el tiempo de ejecución de la función `f()` depende de la iteración i. Paraleliza el bucle anterior con una directiva `parallel for`. Realizar ejecuciones en las que se varíen el número de threads y en las que se modifiquen el tipo de schedule a `static`, `dynamic`, `guided`. Comprobar cómo se modifica el tiempo de ejecución en cada caso.

7. La siguiente sección de código muestra dos bucles anidados en los que hay dependencias de datos entre las distintas iteraciones. ¿Qué tipo de paralelización se prodría realizar en OpenMP?

```
for (i = 0; i < N; i++)
    for (j = 0; j < M; j++)
        f[i][j] = min(f[i-1][j], f[i-1][j-a[i]]);
```

8. Obsérvese la siguiente sección de código:

```
for (i = 1; i < n; i++) {
    x[i] = y[i-1] * 2;
    y[i] = y[i] + i;
}
```

   Se produce la siguiente dependencia de datos, en la iteración i se calculan $x[i]$ e $y[i]$, pero el valor de $x[i]$ depende del valor de $y[i-1]$. Si se paraleliza el bucle directamente en OpenMP se obtendrían resultados erróneos derivados de la dependencia de datos que hay entre iteraciones. Sin embargo, una pequeña transformación en el bucle y en el orden de evaluación, permite resolver el problema realizando la paralelización deseada. Realizar la transformación necesaria para poder paralelizar el bucle directamente de forma que se obtenga una ejecución correcta.

9. La siguiente sección de código muestra un bucle en el que el tiempo de ejecución de la función `f(i)` depende de la iteración. ¿Qué tipo de paralelización se podría realizar en OpenMP sin transformar el bucle `while` en un bucle `for`?

```
i = 0;
while (i < n) {
    f(i);
    i = i + 1;
}
```