PROPUESTA DE RESOLUCIÓN DE LOS EJERCICIOS 8, 10 Y 12 DEL TEMA 3

8. Se pretende paralelizar un algoritmo iterativo que actúa sobre una matriz cuadrada **A=(aij)** con 1000 x 1000 enteros usando un modelo programación basado en paso de mensajes (distribuido). Inicialmente se asume que la matriz A tiene un valor inicial $\mathbf{A^{(0)}}$ (sería el valor de A para la iteración inicial, t=0) y se desea realizar la actualización de A en cada iteración (es decir, pasar del valor en la iteración t al valor en t+1). La actualización de cada elemento aij de A en la iteración t+1 se lleva acabo a partir de los valores de A en la iteración anterior t, usando la siguiente expresión:

Si (t+1 es par)	Si (t+1 es impar)
Si (i+j es par) $a_{i,j}^{(t+1)} = a_{i,j}^{(t)} - 2\,a_{i+1,j}^{(t)} - a_{i,j+1}^{(t)} - a_{i,j-1}^{(t)} + 2\,a_{i-1,j}^{(t)}$ en caso contrario $a_{i,j}^{(t+1)} = a_{i,j}^{(t)}$	Si (i+j es impar) $a_{i,j}^{(t+1)} = a_{i,j}^{(t)} - 2\,a_{i+1,j}^{(t)} - a_{i,j+1}^{(t)} - a_{i,j-1}^{(t)} + 2\,a_{i-1,j}^{(t)}$ en caso contrario $a_{i,j}^{(t+1)} = a_{i,j}^{(t)}$

En definitiva, mientras en una iteración se utiliza una fórmula de diferencias finitas que sólo se aplica a los elementos que ocupan posiciones pares (a_{ij}, con i+j par), en la siguiente iteración, la fórmula se utiliza sólo para actualizar aquellas entradas con posición impar (i+j impar). Sería algo similar a un tablero de ajedrez en el que primero se actualizan las casillas negras, después las blancas, y así sucesivamente. La fórmula de diferencias finitas se aplica a cada elemento a_{ij} de A mientras a_{ij} cumpla una condición que tiene un costo computacional constante.

Se asume que en la fase de descomposición se asigna una tarea a la gestión de cada elemento de A.

a) Establecer qué distribución de la matriz A sobre 4 procesos resulta más apropiada cuando se conoce de antemano que todos los elementos de A requieren el mismo número de iteraciones para converger al resultado. Describir el código de cada proceso en estas circunstancias.

Si todos los $a_{i,j}$ requieren el mismo número de iteraciones, el costo computacional es prácticamente uniforme por elemento de la matriz (considerando varios pasos de tiempo el efecto par-impar no debería causas desequilibrio en el costo por elemento). Por tanto, para conseguir una distribución eficiente solo hay que centrarse en repartir equitativamente los elementos de A entre los 4 procesos de forma que se aproveche la localidad espacial que exhibe el cálculo. Por ello, lo suyo es optar por una **distribución por bloques bidimensionales estándar** de la matriz entre los procesos, donde cada proceso actualizaría una submatriz de 500 x 500 elementos contiguos de la matriz A y las comunicaciones se centrarían en enviar los elementos del contorno de cada submatriz.

Asumo condiciones periódicas (este dato no se da pero uno puede asumirlo), es decir, que:

$a_{-1,j}=a_{999,j}$ $a_{i,-1}=a_{i,999}$ $a_{1000,j}=a_{0,j}$ $a_{i,1000}=a_{i,0}$ con i=0,…,999 , j=0,…,999

Un pseudogódigo muy resumido del proceso[I,J] con I=0, 1 y J=0,1 sería, asumiendo que ya cada proceso tiene su submatriz A_local de la matriz de entrada A en su memoria local:

Proceso [I , J]

```
For t=0,…,t_final
    If (t mod 2==0)        // t es par
        Envía (datos con i+j par de filas superior e inferior, Proceso[ (I+1) mod 2 , J] );
        Envía (datos con i+j par de columna inferior, Proceso[ I , (J+1) mod 2] );
        Recibe (Vector_Filas, Proceso[ (I+1) mod 2 , J] );
        Envía (Vector_Columnas, Proceso[ I , (J+1) mod 2] );
        Actualiza A_local usando valores de A_local, Vector_Filas y Vector_Columnas
    Else // t es impar
        … Código similar cambiando par por impar
```

Como solo hay dos procesos en cada dimensión (filas/columnas) de la malla de procesos, cada proceso envía/recibe de un solo proceso (Proceso((I+1) mod 2,J) o) en cada dimensión de la malla.

b) Establecer la distribución más apropiada sobre 4 procesos cuando se sabe que el número de iteraciones que requiere cada elemento de A no es el mismo para todos los elementos de A, sino que es proporcional al número de columna que ocupa. Justificar la respuesta con concisión.

En este caso, no todos los $a_{i,j}$ requieren el mismo número de iteraciones aunque sí hay localidad espacial en el cálculo. Se podría optar directamente por usar una distribución cíclica por bloques usando un tamaño de bloque de tamaño medio (por ejemplo 64). De esa forma la carga se repartiría aceptablemente entre los procesos aunque el costo de comunicación entre los procesos se incrementaría bastante respecto a la solución del caso a).

No obstante, si uno se da cuenta de que, en este caso, todas las filas tendrían el mismo costo computacional, lo más eficiente es optar por una **distribución estándar por bloques de filas** de la matriz entre los procesos, donde cada proceso actualizaría una submatriz de 250 x 1000 elementos contiguos de la matriz A. De esta forma, la carga estaría perfectamente repartida y las comunicaciones se centrarían en enviar elementos de la primera y ultima fila de cada submatriz, lo que sería bastante más eficiente que en el caso en el que se sigue una distribución cíclica por bloques.

10. Se desea encontrar todas las ocurrencias de una subcadena particular con m caracteres, llamado **patrón**, dentro de otra cadena, llamada **texto** con N caracteres (N>>m y N es múltiplo de m). Tanto el patrón como el texto se almacenan internamente como arrays de caracteres. Describe un algoritmo paralelo para resolver el problema.

a) Establecer la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas, suponiendo suponiendo que el texto se reparte inicialmente por bloques de caracteres consecutivos entre las tareas.

Para ilustrar el algoritmo, supongamos que P=3 procesos, el patrón es “casa” y el texto es “La_casa donde_mi_bella_casandra_se_casa”.

En este caso, un reparto por bloques de caracteres consecutivos, nos llevaría a la siguiente distribución (el tamaño de bloque sería de 13 caracteres):

Subcadena	La_**casa**_donde	_mi bella_**cas**	**a**ndra_se_**casa**
Tarea	**P0**	**P1**	**P2**

Una primera aproximación al problema consistiría en que cada tarea buscara las ocurrencias del patrón en su subcadena y después todas las tareas se coordinaran para sumar las ocurrencias encontradas y obtener el total. No obstante, con dicha aproximación no se contarían las ocurrencias que están a caballo entre una tarea y la siguiente como la que existe en el ejemplo entre P0 y P1. Por ello, se requiere comunicación entre las tareas, no solo para conseguir la suma de las ocurrencias, sino para identificar ocurrencias del patrón entre tareas consecutivas. A continuación, se esboza una propuesta algorítmica que resuelve, con un patrón de comunicación en anillo bastante eficiente, la identificación de ocurrencias intercaladas entre tareas consecutivas. La idea es que cada tarea (exceptuando la última), además de contar las ocurrencias locales, pasa a la siguiente tarea el número de caracteres del comienzo del patrón identificados al final de su subcadena (en el ejemplo anterior, P0 pasaría un 0 a P1 y P1 pasaría un 3 a P2). Después cada tarea usa el número recibido para ver si mantiene una ocurrencia un patrón intercalado con la tarea anterior.

Tarea [i, i=0,…,P-1]

```
int N_ocurrencias;
int N_caract_final, N_caract_inicio, N_anterior;
// Se asume que el patrón y la subcadena local ya han sido proporcionados a cada tarea
char * patron;
char * Subcadena;
{
 Obtener (patron, Subcadena);
 N_ocurrencias=Buscar_ocurrencias(Subcadena, patron); // Cuenta ocurrencias en subcadena local
 N_caract_final=… Número caracteres del principio del patrón al final de Subcadena;
  If (i<P-1)
    Envia (N_caract_final ,Tarea[i+1]);
   If (i>0)
      {Recibe (N_anterior ,Tarea[i+1]);
        N_caract_final=… Número caracteres del final del patrón en comienzo de Subcadena;
        If ( N_anterior+N_caract_final==Tamaño(patron))  N_ocurrencias++;
       }
  // Habría que diseñar un algoritmo de reducción todos_a_todos para sumar las ocurrencias locales
  Reduccion_a_Todos(N_ocurrencias, suma) //El resultado se obtendría en todas las tareas
  }
```

b) Establecer una estrategia de aglomeración y asignación eficiente sobre 8 procesadores para dos casos diferentes:

- Todos los procesadores son del mismo tipo y potencia .

En este caso el algoritmo obtenido en la fase de descomposición (apartado a)es perfectamente válido con P=8, ya que una distribución por bloques repartiría la carga perfectamente.

- Los procesadores son de tipos y potencias muy dispares pero desconocidos a priori.

No se podría hacer un reparto estático por bloques de la carga ya que no se puede estimar a priori cómo repartir el texto entre los procesos para que todos tardaran aproximadamente el mismo tiempo para procesar su subcadena asignada.

Habría que optar por un reparto dinámico del texto entre los procesos. Para ello, se podría hacer que uno de los procesos mantuviera el texto (proceso maestro) y fuera repartiendo dinámicamente subcadenas de caracteres contiguos de un determinado tamaño al resto de procesos (esclavos). Cuando un proceso esclavo termina y devuelve sus resultados, el proceso maestro le asigna una nueva subcadena del texto. Cada proceso esclavo debe enviar al maestro, como resultados, lo siguiente:
a) El número de ocurrencias del patrón encontradas en la subcadena,
b) Número caracteres del comienzo del patrón al final de la subcadena
c) Número caracteres del final del patrón en comienzo de la subcadena.

El maestro va actualizando el número de ocurrencias encontradas usando los resultados que va recibiendo de los procesos esclavos (suma las ocurrencias locales recibidas y chequea si hay ocurrencias intercaladas entre subcadenas consecutivas).

Para el tamaño de bloque, el maestro podría empezar asignando una subcadena de tamaño N/5P a cada proceso esclavo e ir decrementando el tamaño de bloque conforme queda menos texto sin procesar.

12. Considérese una matriz bidimensional cuadrada con 1000 **x** 1000 enteros que se reparte entre 10 procesadores siguiendo una distribución cíclica por bloques. Especifica los 4 parámetros (P_x, P_y, mb_x, mb_y) que definen la distribución particular de la matriz entre los 10 procesadores en cada uno de estos casos:

a) Distribución cíclica por columnas.
Sería una distribución cíclica donde el tamaño de bloque corresponde con el tamaño de una columna, esto es, 1000 filas y 1 columna (1000x1). Como el elemento que se reparte cíclicamente es la columna, la malla de procesos tendría una sola fila y 10 columnas (1x10). Por tanto, los parámetros serían: **(P_x=1, P_y=10, mb_x=1000, mb_y=1)**.

b) Distribución cíclica por filas.
Sería una distribución cíclica donde el tamaño de bloque corresponde con el tamaño de una fila, esto es, 1 fila y 1000 columnas (1x1000). Como el elemento que se reparte cíclicamente es la fila, la malla de procesos tendría 10 filas y 1 columna (10x1).
Por tanto, los parámetros serían: **(P_x=1, P_y=10, mb_x=1, mb_y=1000)**.

c) Distribución cíclica por bloques de columnas con tamaño de bloque 5.
Sería una distribución cíclica donde el tamaño de bloque con el tamaño de 5 columnas, esto es, 1000 filas y 5 columnas (1000x5). Como el elemento que se reparte cíclicamente son bloques de 5 columnas, la malla de procesos tendría una sola fila y 10 columnas (1x10). Por tanto, los parámetros serían: **(P_x=1, P_y=10, mb_x=1000, mb_y=5)**.

d) Distribución cíclica por bloques de filas con tamaño de bloque 5.
Sería una distribución cíclica donde el tamaño de bloque corresponde con el tamaño de 5 filas, esto es, 5 filas y 1000 columnas (5x1000). La malla de procesos tendría 10 filas y 1 columna (10x1).
Por tanto, los parámetros serían: **(P_x=1, P_y=10, mb_x=5, mb_y=1000)**.