Universidad De Granada

Grado ingeniería informática (2017 – 2018)

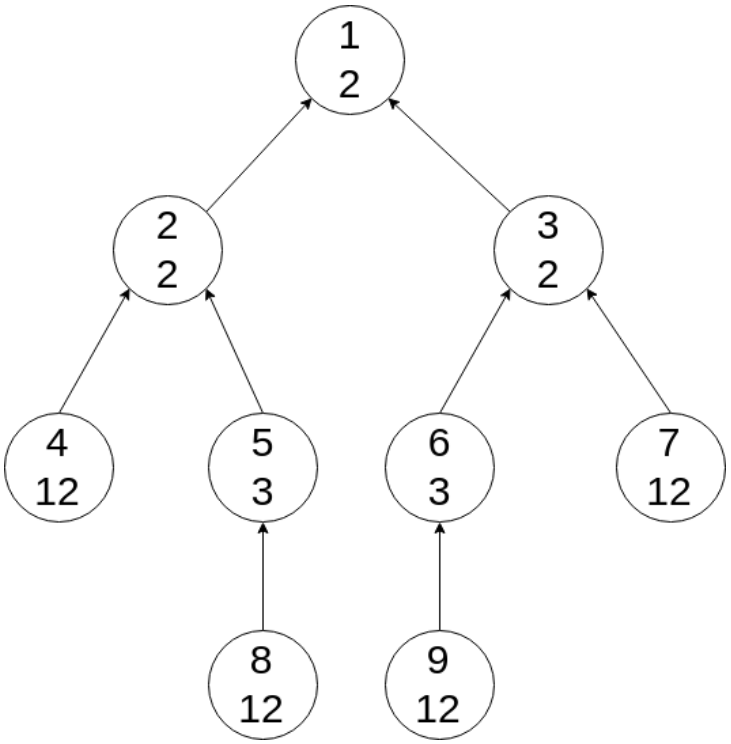
Programación Paralela

Entrega 4 de ejercicios

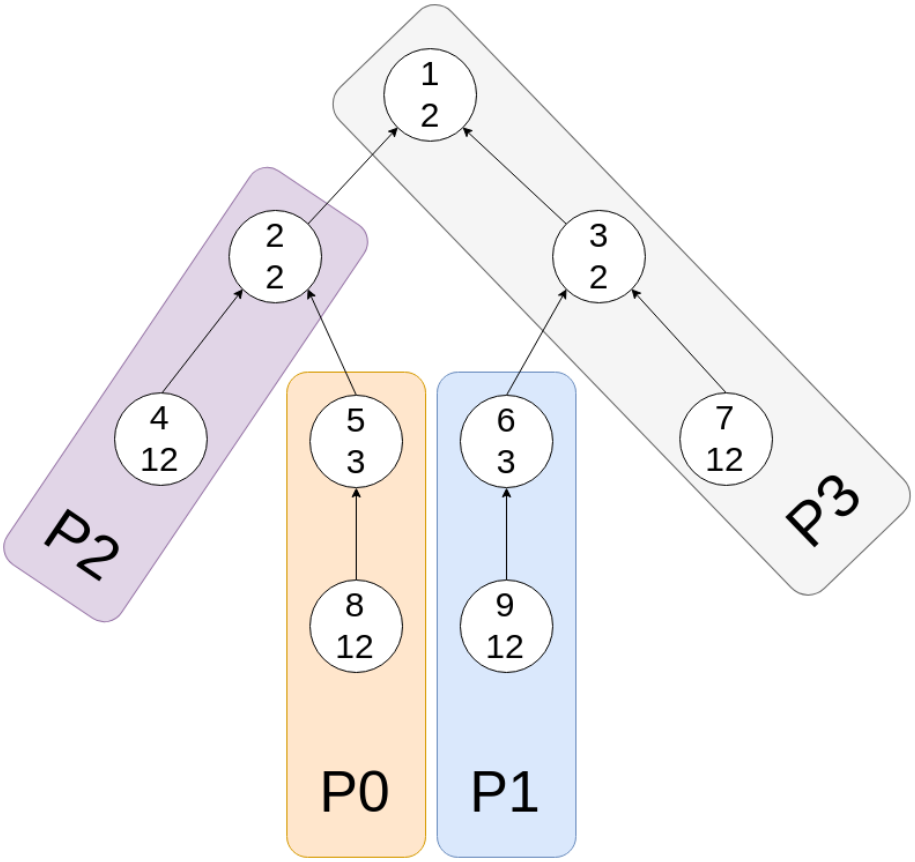
Trabajo realizado por Antonio Miguel Morillo Chica.

Ejercicio 1.

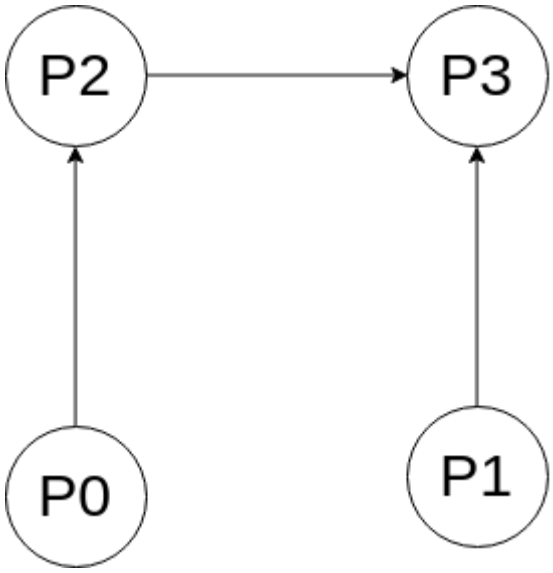
¿Cuál es el grado medio de concurrencia de los siguientes grafos de dependencias entretareas? en el que cada nodo está etiquetado con su identificación en la parte superior y con su coste en la parte inferior. Asumiendo que el costo de comunicación entre tareas fuese despreciable, establecer cuál sería la asignación óptima de las tareas del grafo a cuatro procesos en el caso a) Ttres procesos en el caso b). Dibujar un diagrama de ejecución.

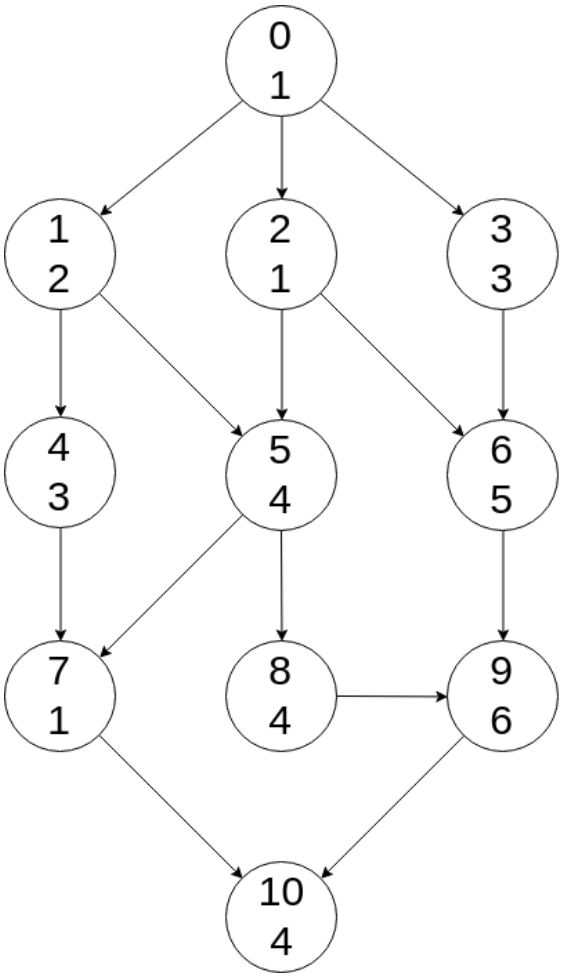


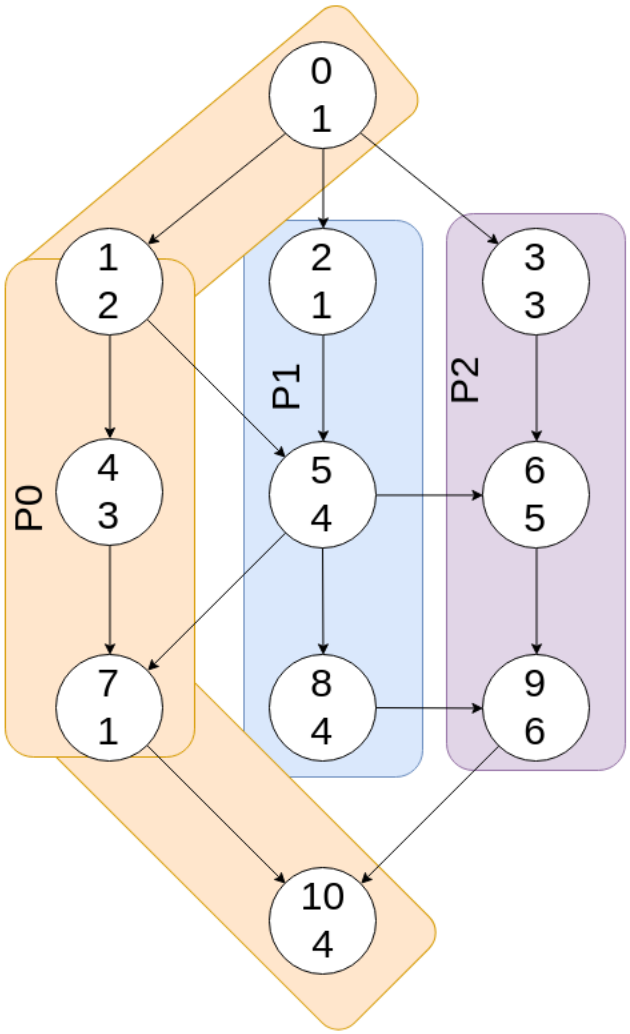
Las tareas de las hojas del grafo son las mismas tareas mas pesadas. Por ello buscamos paralelizar cada una de ellas de forma que la ejecute un proceso. Las ramas internas de los procesos 5 y 6 se pueden hacer con un solo poceso con los respectivos 8 y 9. Tras esto un tercer proceso se encargaría del del nodo 4 y 2 que tendía que esperar al primer proceso que se encargue de 5 y 8. Finalmente el último porceso se encarga de 7, 3 y 1. De forma que:

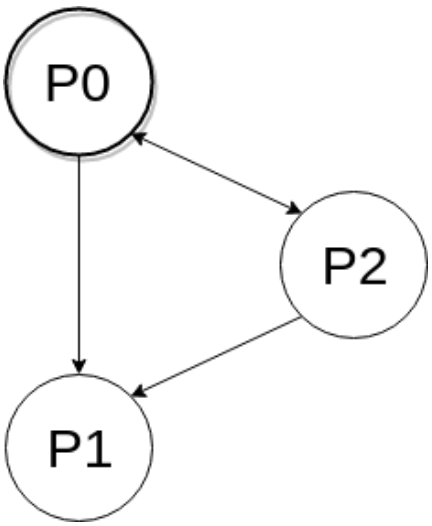


Cuyo diagrama de ejecución sería el siguiente:




0
1
1
2
2
1
3
3
4
3
5
4
6
5
7
1
8
4
9
6
10
4


0
1
1
2
2
1
3
3
4
3
5
4
6
5
7
1
8
4
9
6
10
4
P0
P1
P2


P0
P2
P1

Ejercicio 3.

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

$$dy_i = \frac{y_{i-1} + y_i * y_{i+1}}{8}$$

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

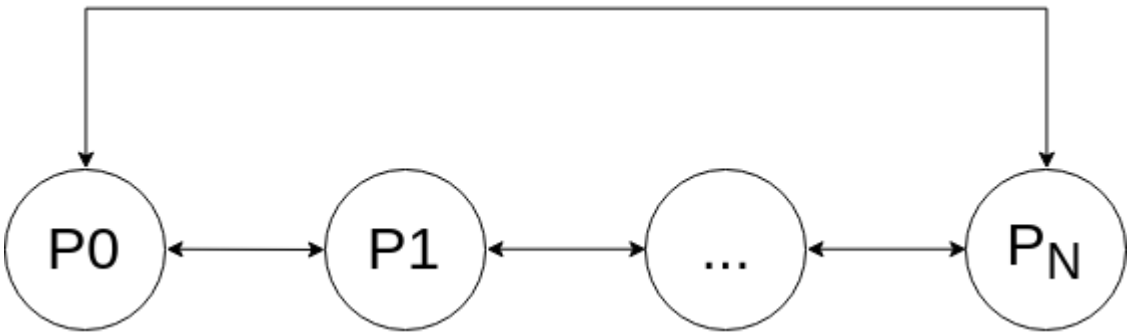
$$y_0 = y_N, y_{N+1} = y_1, y_{N+2} = y_2$$

Se supone que la función se evaluará repetidamente durante un cierto número de iteraciones y en cada iteración actual.

a) **Establecer la descomposición de tareas así como la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas.**

En el caso general cada tarea sería el calculo de una componente si tuvienemos N componentes y P procesadores P=N de forma que, cada proceso tendría una componente y necesitaría una del anterior y dos de los dos procesos sigientes.

La estructura de comunicación de los procesos sería de la siguiente forma:

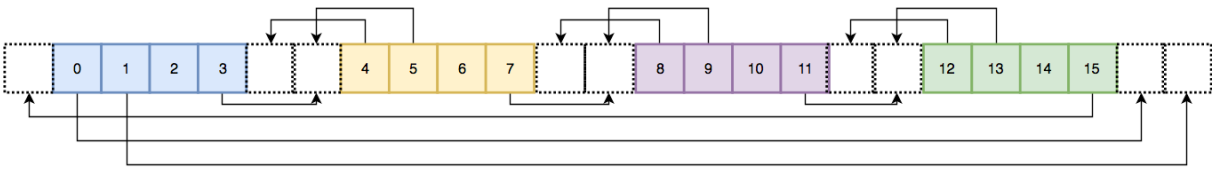


b) **Definir una estrategia de asignación eficiente sobre 4 y 8 procesadores. Cómo se distribuyen el vector de entrada y el vector solución entre los procesadores con la solución que se propone.**

Supongamos el número de componentes de "y" fuese un múltiplo del numeor de procesos o dicho de otra manera N | numP (N divide a numP). De tal forma para N=16 para 4 procesos el vector se distribuiría entre los 4 procesos de forma que cada uno tuviesen N/numP componentes encargandose de hacer N/numP tareas, es decir, 4 componentes y 4 tareas por proceso.

Con 8 sería de igual pero a cada proceso le coresponderían dos componentes del vector y realizaría 2 tateas.

En ambos casos el vector solución sería la recolección de las soluciones parciales aportadas por los 4/8 procesos. En el siguiente representación se puede observar para 4 procesadores el particionamiento del vector y la comunicación de las "celdas fantasmas" que son necesarias:



Ejercicio 8.

Se pretende paralelizar un algoritmo iterativo que actúa sobre una matriz cuadrada A=(aij) con 1000x1000 enteros. Inicialmente se asume que la matriz A tiene un valor inicial $A^{(0)}$ (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) $a_{i,j}^{(t+1)} = a_{i+1,j}^{(t)} - 2a_{i,j+1}^{(t)} - a_{i,j+1}^{(t)} - a_{i,j-1}^{(t)} + 2a_{i-1,j}^{(t)}$ en caso contrario $a_{i,j}^{(t+1)} = a_{i,j}^{(t)}$	Si (i+j es impar) $a_{i,j}^{(t+1)} = a_{i,j}^{(t)} - 2a_{i+1,j}^{(t)} - a_{i,j+1}^{(t)} - a_{i,j-1}^{(t)} + 2a_{i-1,j}^{(t)}$ 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.

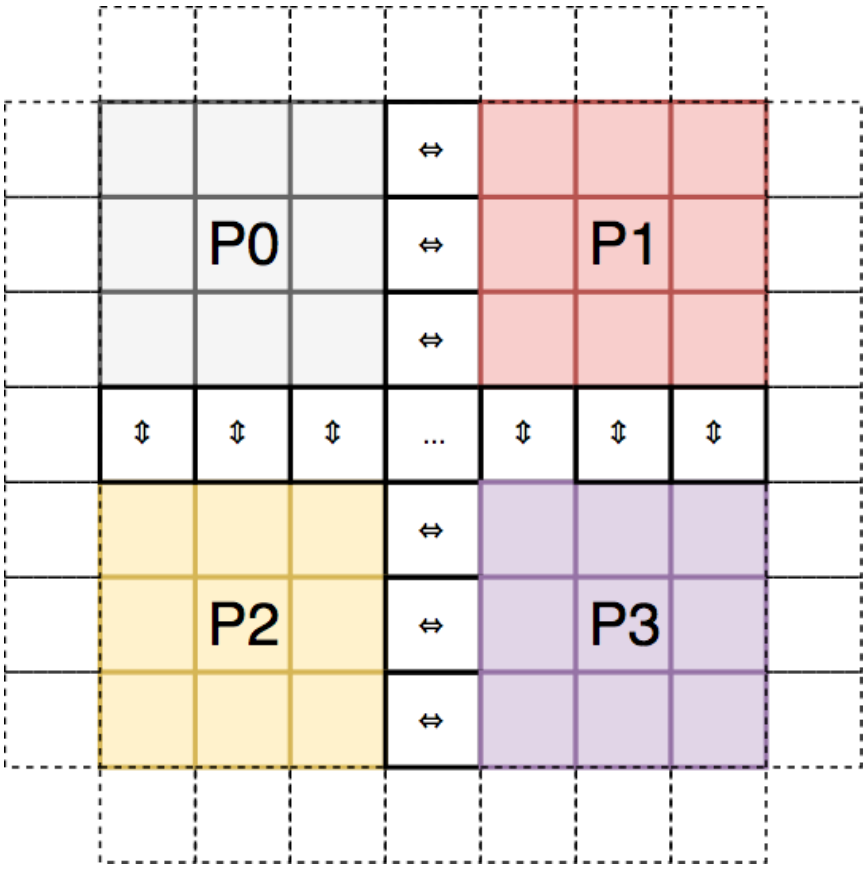
Se asume que en la fase de descomposición se asigna una tarea a la gestión de cada elemento de A.

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.

La distribución de la matriz la he ajustado como hemos visto en las prácticas, dividimos la matriz en submatrices de tamaño 1000/4, es decir, cada proceso se encargaría de una submatriz de tamaño 250x250 donde en cada iteracción t, se actualizarían la mitad de sus valores.

De igual forma tenemos que tener en cuenta que cuando estemos en posiciones i,j que estén en el "borde" de las submatrices, tendremos que recoger los datos de otros procesos, hay que ver a la matriz como una especie de esfera, donde los bordes paralalelos están "unidos", en la siguiente imagen se puede observar la distribución de los procesos (submatrices) en la columna/fila centrales se indica que los datos de la izq. y der. han de ser compartidos entre procesos al igual que las celdas fantasma externas: (hay que tener en cuenta que la matriz que se observa es de 6x6 pero realmente actuariamos igual para 1000x1000.



El código que se va a usar sería algo parecido a lo siguiente, lo escribo en psudocódigo para que sea más general.

```
inicio_local_i = (idProceso/2) * tamBloque
fin_local_i = (idProceso/2 + 1) * tamBloque
inicio_local_j = (idProceso%2) * tamBloque
fin_local_j = (idProceso%2+1) * tamBloque

vector buff_fil_sup, buff_fil_inf, buff_col_izq, buff_col_der
vector mi_fil_sup, mi_fil_inf, mi_col_izq, mi_col_der

for k in A
      recibo(buff_fil_sup) , recibo(buff_fil_inf)
      recibo(buff_col_izq) , recibo(buff_fil_der)
      envio(mi_fil_sup)    , envio(mi_fil_inf)
      envio(mi_col_izq)    , envio(mi_col_izq)

   if (k%2) them
      for i in A
         for j in A
            if ( i+j % 2) them
               A[i,j] = A[i,j] - 2A[i+1,j] - A[i,j+1] - A[i,j-1] + 2A[i-1,j]
            else them
               A[i,j] = A[i.j]

   else them
      recibo(buff_fil_sup) , recibo(buff_fil_inf)
      recibo(buff_col_izq) , recibo(buff_fil_der)
      envio(mi_fil_sup)    , envio(mi_fil_inf)
      envio(mi_col_izq)    , envio(mi_col_izq)

      for i in A
         for j in A
            if ( i+j % 2) them
               A[i,j] = A[i,j] - 2A[i+1,j] - A[i,j+1] - A[i,j-1] + 2A[i-1,j]
            else them
               A[i,j] = A[i.j]
```

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 seguiríamos una distribución parecida a la anterior, dividiariamos la matriz subMatrices de igual tamaño dos a dos pero estas

parejas serían de distinto tamaño. Si nombramos a las columnas de una matriz del 1-10 representando que en la primera se hará una sola iteración por elemento y en la decima 10 iteracciones podemos ver que:

1 + 2 + 3 + 4 + 5 + 6 + 7 = 28 y 8 + 9 + 10 = 27, vemos como los elementos entre las columnas 1-7 harían 1*10 + 2*10 + 3*10 + 4*10 + 5*10 + 6*10 + 7*10 iteracciones que son un total de 280 iters y en el segundo caso: 8*10 + 9*10 + 10*10 = 270 iters. En la siguiente imagen se puede ver la distribución, pero, generalizando, el 70% de las columnas pertenecerán a un par de procesos y el 30% restante al otro par: