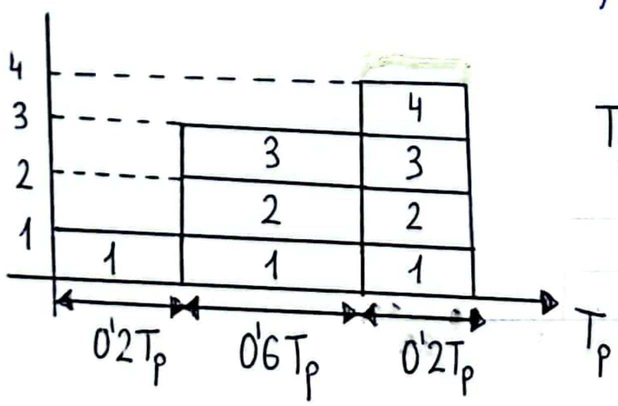
# Arquitectura de Computadores

# Relación de Ejercicios 2

## Ejercicios

1. Un programa tarda 40s en ejecutarse en un multiprocesador. Durante un 20% de ese tiempo se ha ejecutado en cuatro procesadores; durante un 60%, en tres; y durante el 20% restante, en un procesador (consideramos que se ha distribuido la carga de trabajo por igual entre los procesadores que colaboran en la ejecución en cada momento, despreciamos sobrecarga). ¿Cuánto tiempo tardaría en ejecutarse el programa en un único procesador? ¿Cuál es la ganancia en velocidad obtenida con respecto al tiempo de ejecución secuencial? ¿Y la eficiencia?




$T_p = 40s = T_p(4)$

$$T_s = 0'2T_p + 3\cdot 0'6T_p + 4\cdot 0'2T_p = 2'8T_p \Rightarrow \boxed{T_s = 2'8T_p = 112s}$$

$$S(4) = \frac{T_s}{T_p(4)} = \frac{2'8\cdot \cancel{T_p(4)}}{\cancel{T_p(4)}} \Rightarrow \boxed{S(4) = 2'8}$$

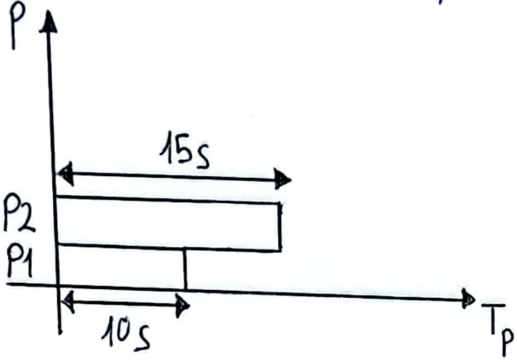
(Recordemos que:
- $S_{máx} = p$ cuando $T_p = \frac{T_s}{p}$
- $S_{mín} = 1$ cuando $T_p = T_s$ )

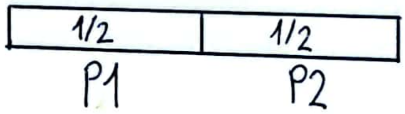
$$E(4) = \frac{S(4)}{p} = \frac{S(4)}{4} = \frac{2'8}{4} \Rightarrow \boxed{E(4) = 0'7}$$

(Recordemos que:
- $E_{máx} = 1$ cuando $S = S_{máx} = p$
- $E_{mín} = \frac{1}{p}$ cuando $S = S_{mín} = 1$ )

② Un programa tarda 20s en ejecutarse en un procesador P1, y requiere 30s en otro procesador P2. Si se dispone de los dos procesadores para la ejecución del programa (despreciamos sobrecarga):

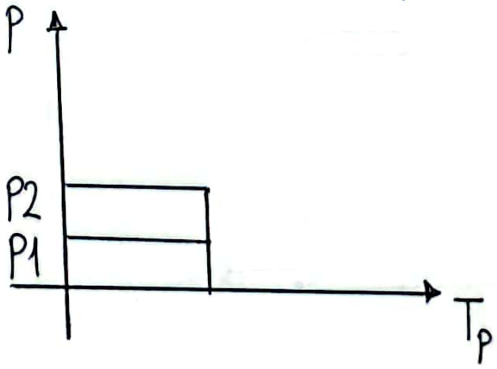
a) ¿Qué tiempo tarda en ejecutarse el programa si la carga de trabajo se distribuye por igual entre los procesadores P1 y P2?



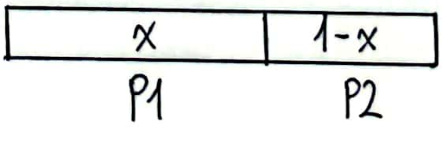




$$T_p^{P1,P2}\left(\tfrac{1}{2},\tfrac{1}{2}\right) = \text{máx}(10s, 15s) = 15s$$

b) ¿Qué distribución de carga entre los dos procesadores P1 y P2 permite el menor tiempo de ejecución utilizando los dos procesadores en paralelo? ¿Cuál es este tiempo?




El menor tiempo de ejecución en paralelo lo conseguiremos cuando todos los procesadores trabajen y terminen al mismo tiempo, de forma que ninguno se quede ocioso.




$$T_p^{P1}(x) = T_p^{P2}(1-x)$$

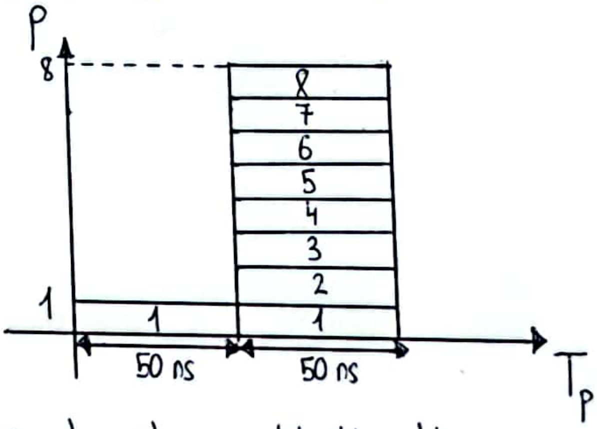
$$20s \cdot x = 30s \cdot (1-x)$$

$$50s \cdot x = 30s$$

$$x = \frac{3}{5} \Rightarrow (1-x) = 1 - \frac{3}{5} = \frac{2}{5}$$

$$T_p^{P1,P2}(x, 1-x) = T_p^{P1,P2}\left(\tfrac{3}{5},\tfrac{2}{5}\right) = 20s \cdot \frac{3}{5} = 12s$$

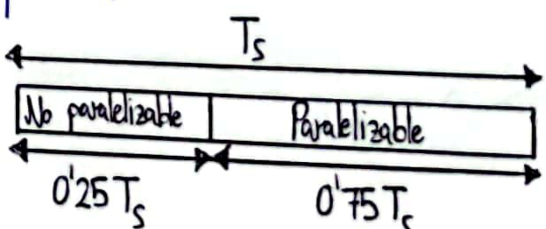
(3) ¿Cuál es fracción de código paralelo de un programa secuencial que, ejecutado en paralelo en 8 procesadores, tarda un tiempo de 100 ns, durante 50 ns utiliza un único procesador y durante otros 50 ns utiliza 8 procesadores (distribuyendo la carga de trabajo por igual entre los procesadores)?




Sea $f_p$ la fracción de código paralelizable del programa secuencial. Tenemos que:

$$f_p = \frac{8 \cdot 50\,ns}{T_s} = \frac{8 \cdot 50\,ns}{50\,ns + 8 \cdot 50\,ns} \Rightarrow f_p = \frac{8}{9}$$

(4) Un 25% de un programa no se puede paralelizar, el resto se puede distribuir por igual entre cualquier número de procesadores. ¿Cuál es el máximo valor de ganancia de velocidad que se podría conseguir al paralelizarlo en p procesadores, y con infinitos? ¿A partir de cuál número de procesadores se podrían conseguir ganancias mayores o iguales que 2?




El máximo valor de ganancia de velocidad que se podría conseguir al paralelizar el programa en p procesadores sería:

$$S(p) = \frac{T_s}{T_p(p)} = \frac{\cancel{T_s}}{0'25\cancel{T_s} + \frac{0'75\cancel{T_s}}{p}} \Rightarrow S(p) = \frac{1}{0'25 + \frac{0'75}{p}}$$

Para infinitos procesadores tendríamos que:

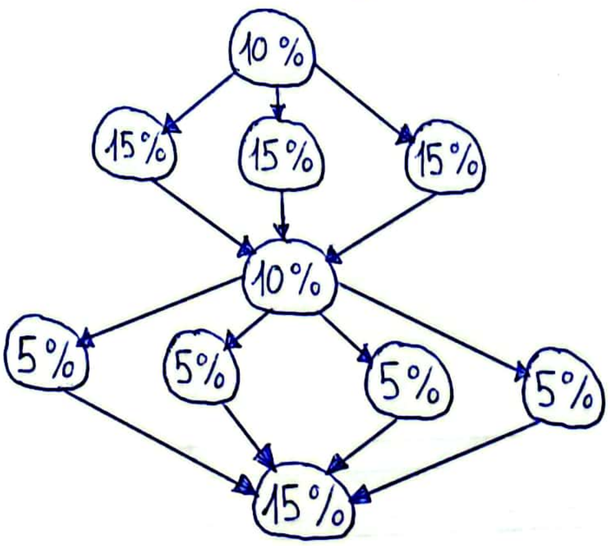
$$S(p) = \frac{1}{0'25 + \frac{0'75}{p}} \xrightarrow[p \to \infty]{} \boxed{S(p \to \infty) = \frac{1}{0'25} = 4}$$

Finalmente, calculemos el número de procesadores a partir del cual se podrían conseguir ganancias mayores o iguales que 2:

$$S(p) \geq 2 \iff \frac{1}{0'25 + \frac{0'75}{p}} \geq 2 \iff \frac{p}{0'25p + 0'75} \geq 2 \iff$$

$$\iff p \geq 0'5p + 1'5 \iff 0'5p \geq 1'5 \iff \boxed{p \geq 3}$$

5 En la Figura 1, se presenta el grafo de dependencia entre tareas para una aplicación.

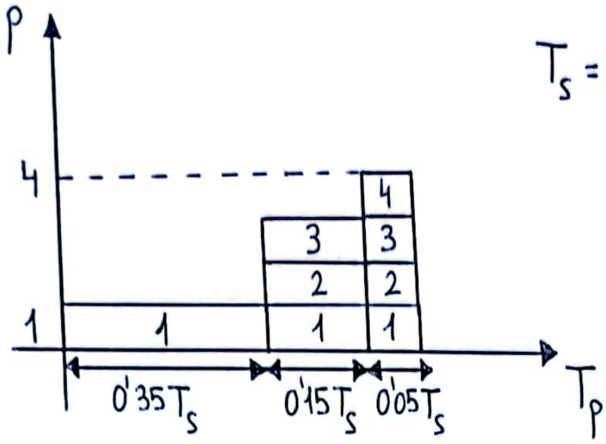



$\Rightarrow$ Grado de paralelismo = 4

Figura 1

La figura muestra la fracción del tiempo de ejecución secuencial que la aplicación tarda en ejecutar grupos de tareas del grafo. Suponiendo un tiempo de ejecución secuencial de 60 s, que las tareas no se pueden dividir en tareas de menor granularidad y que el tiempo de comunicación es despreciable, obtener el tiempo de ejecución en paralelo y la ganancia en velocidad en un computador con:

a) 4 procesadores.

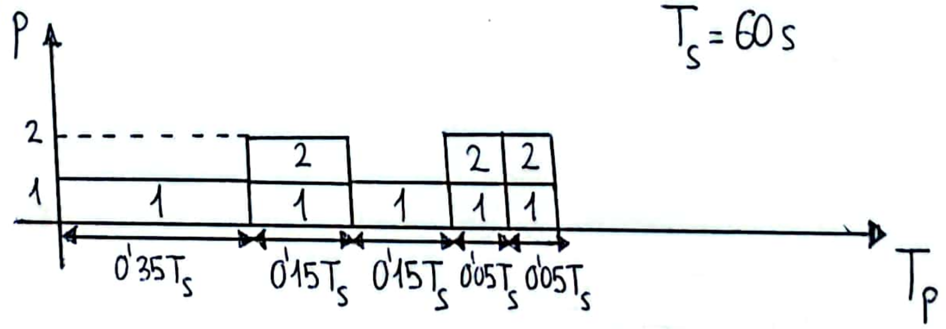



$T_s = 60\,s$

$$T_p(4) = 0'35T_s + 0'15T_s + 0'05T_s = 0'55T_s \Rightarrow T_p(4) = 0'55T_s = 0'55 \cdot 60s = 33s$$

$$S(4) = \frac{T_s}{T_p(4)} = \frac{\cancel{T_s}}{0'55\cancel{T_s}} = \frac{1}{0'55} \Rightarrow S(4) = 1'82$$

b) 2 procesadores.

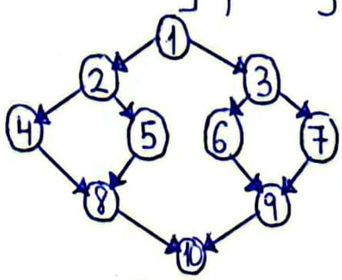



$T_s = 60\,s$

$$T_p(2) = 0'35T_s + 2 \cdot 0'15T_s + 2 \cdot 0'05T_s = 0'75T_s \Rightarrow T_p(2) = 0'75T_s = 0'75 \cdot 60s = 45\,s$$

$$S(2) = \frac{T_s}{T_p(2)} = \frac{\cancel{T_s}}{0'75\cancel{T_s}} = \frac{1}{0'75} \Rightarrow S(2) = 1'33$$

6 Un programa se ha conseguido dividir en 10 tareas. El orden de precedencia entre las tareas se muestra con el grafo dirigido de la Figura 2.

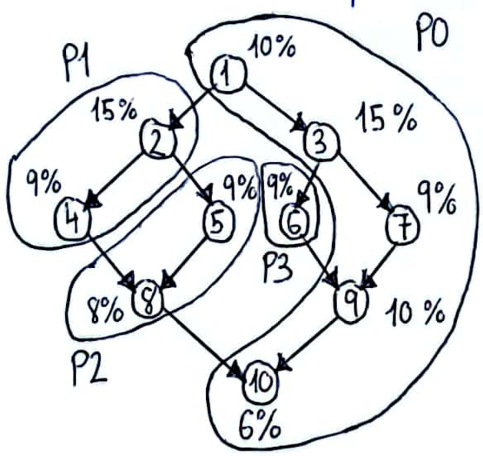



$\Rightarrow$ Grado de paralelismo = 4

Figura 2

La ejecución de estas tareas en un procesador supone un tiempo de 2 s. El 10 % de ese tiempo es debido a la ejecución de la tarea 1; el 15 % a la ejecución de la tarea 2; otro 15 % a la ejecución de 3; cada tarea 4, 5, 6 o 7 supone el 9 %; un 8 % supone la tarea 8; la tarea 9 un 10 %; por último, la tarea 10 supone un 6 %. Se dispone de una arquitectura con 8 procesadores para ejecutar la aplicación. Consideramos que el tiempo de comunicación se puede despreciar.

a) ¿Qué tiempo tarda en ejecutarse el programa en paralelo?




$$T_s = 2s$$

$$T_p(8) = T_p(4) = 0'1\,T_s + 0'15\,T_s + 0'09\,T_s + 0'1\,T_s + 0'06\,T_s =$$

$$= 0'5\,T_s \Rightarrow T_p(8) = 0'5\,T_s = 0'5 \cdot 2s = 1s$$

b) ¿Qué ganancia en velocidad se obtiene con respecto a su ejecución secuencial?

$$S(8) = \frac{T_s}{T_p(8)} = \frac{2s}{1s} \Rightarrow S(8) = 2$$

7 Se quiere paralelizar el siguiente trozo de código:

```
{Cálculos antes del bucle}
for (i=0; i<w; i++) {
    Código para i
}
{Cálculos después del bucle}
```

Los cálculos antes y después del bucle suponen un tiempo de $t_1$ y $t_2$, respectivamente. Una iteración del ciclo supone un tiempo $t_i$. En la ejecución paralela, la inicialización de $p$ procesos supone un tiempo $k_1 p$ ($k_1$ constante), los procesos se comunican y se sincronizan, lo que supone un tiempo $k_2 p$ ($k_2$ constante): $k_1 p + k_2 p$ constituyen la sobrecarga.

a) Obtener una expresión para el tiempo de ejecución paralela del trozo de código en $p$ procesadores ($T_p$).

Llamemos $t = t_1 + t_2$ y $k = k_1 + k_2$. Tenemos que:

$$\boxed{T_p(p,w) = T_c(p,w) + T_o(p) = t + \underbrace{\left\lceil \frac{w}{p} \right\rceil}_{\substack{\frac{w}{p} \text{ se redondea} \\ \text{al entero superior}}} \cdot t_i + k \cdot p} \quad (1)$$

b) Obtener una expresión para la ganancia en velocidad de la ejecución paralela con respecto a una ejecución secuencial ($S$).

$$\boxed{S(p,w) = \frac{T_s}{T_p(p,w)} = \frac{t + w \cdot t_i}{t + \left\lceil \frac{w}{p} \right\rceil \cdot t_i + k \cdot p}}$$

c) ¿Tiene el tiempo $T_p$ con respecto a $p$ una característica lineal o puede presentar algún mínimo? ¿Por qué? En caso de presentar un mínimo, ¿para qué número de procesadores $p$ se alcanza?

En la expresión $T_p(p,w) = t + \left\lceil \frac{w}{p} \right\rceil \cdot t_i + k \cdot p$, el término $t + \left\lceil \frac{w}{p} \right\rceil \cdot t_i$, o tiempo de cálculo paralelo, tiene tendencia a decrecer con pendiente que va disminuyendo conforme se incrementa $p$ (debido a que $p$ está en el divisor), y el término $k \cdot p$, o tiempo de sobrecarga, crece conforme se incrementa p con pendiente k constante. Habrá oscilaciones en el tiempo de cálculo paralelo debidas al redondeo al entero superior del cociente $\frac{w}{p}$, pero la tendencia es que $T_p$ va decreciendo. Dado que la pendiente de la sobrecarga es constante y que la pendiente del tiempo de cálculo decrece conforme se incrementa p, llega un momento en que el tiempo de ejecución en paralelo pasa de decrecer a crecer, es decir, $T_p$ tiene un mínimo.

Para calcular dicho mínimo:

$$T_p(p,w) = t + \frac{w}{p} \cdot t_i + k \cdot p$$ (Nótese que hemos eliminado el redondeo)

$$T'_p(p,w) = -\frac{w}{p^2} \cdot t_i + k$$

$$T'_p(p,w) = 0 \iff \frac{w}{p^2} \cdot t_i = k \iff \boxed{p = \sqrt{\frac{w \cdot t_i}{k}}} \quad (2)$$

El resultado negativo de la raíz se descarta. En cuanto al resultado positivo, debido al redondeo, habrá que comprobar para cuál de los naturales próximos al resultado obtenido (incluido el propio resultado si es un número natural) se obtiene un menor tiempo. Debido al redondeo hacia arriba de la expresión (1), se debería comprobar necesariamente que:

1) El natural p' menor o igual y más alejado al resultado p generado con (2) para el que $\left\lceil \frac{w}{p} \right\rceil = \left\lceil \frac{w}{p'} \right\rceil$ y
2) El natural p' mayor y más próximo al p generado con (2) para el que $\left\lceil \frac{w}{p'} \right\rceil = \left\lceil \frac{w}{p} \right\rceil - 1$,

En cualquier caso, el número de procesadores que obtengamos debe ser menor que w (que es el grado de paralelismo del código).

Finalmente, comprobemos que $p = \sqrt{\frac{w \cdot t_i}{k}}$ se trata de un mínimo:

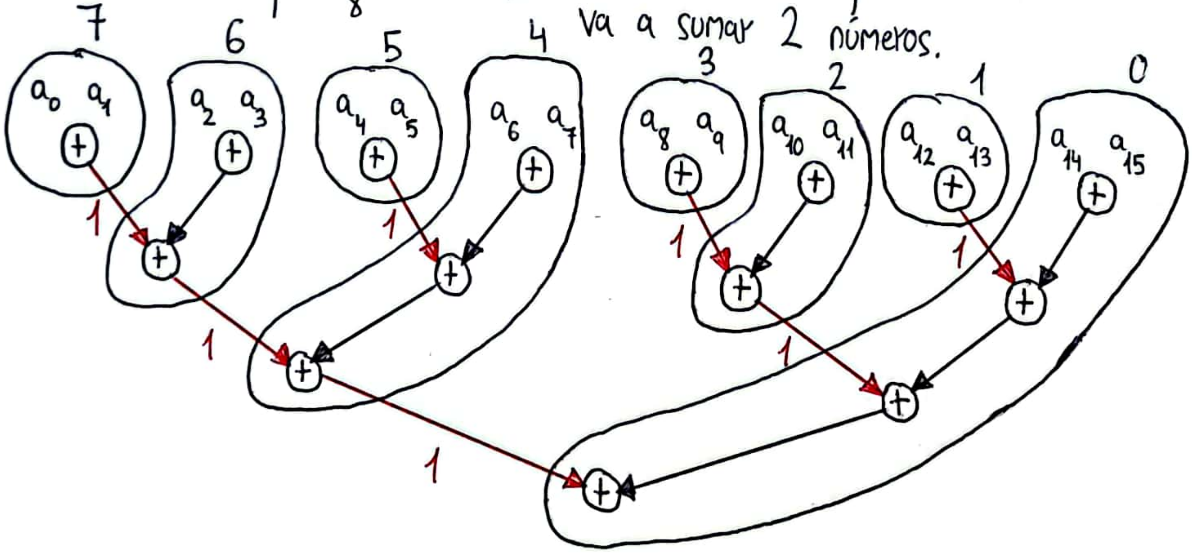
$$T''_p(p,w) = \frac{2 \cdot w \cdot t_i}{p^3} > 0 \quad \forall p,w \in \mathbb{N}$$

Por tanto, $p = \sqrt{\frac{w \cdot t_i}{k}}$ es un mínimo.

(8) Supongamos que se va a ejecutar en paralelo la suma de $n$ números en una arquitectura con $p$ procesadores o cores ($p$ y $n$ potencias de dos) utilizando un grafo de dependencias en forma de árbol (divide y vencerás) para las tareas.

a) Dibujar el grafo de dependencias entre tareas para $n = 16$ y $p = 8$. Hacer una asignación de tareas a procesos.

$\frac{n}{p} = \frac{16}{8} = 2 \Rightarrow$ Cada uno de los procesadores va a sumar 2 números.




Se han señalado en rojo las comunicaciones entre procesadores (el resto de comunicaciones son internas a su procesador correspondiente).

b) Obtener el tiempo de cálculo paralelo para cualquier $n$ y $p$ con $n > p$ suponiendo que se tarda una unidad de tiempo en realizar una suma.

| | $n=16, p=8$ | $n=32, p=8$ | $n, p=8$ | $n, p$ |
|---|---|---|---|---|
| Nivel 0 del grafo | 1 | 3 | $\frac{n}{8}-1$ | $\frac{n}{p}-1$ |
| Nivel 1 del grafo | 1 | 1 | 1 | 1 |
| Nivel 2 del grafo | 1 | 1 | 1 | 1 |
| Nivel 3 del grafo | 1 | 1 | 1 | 1 |
| | | | | ⋮ |

$\log_2(p)$

$$\boxed{T_p(n,p) = \left(\frac{n}{p}-1\right) + \log_2(p)}$$

c) Obtener el tiempo de comunicación del algoritmo suponiendo que:

1) Las comunicaciones en un nivel del árbol se pueden realizar en paralelo en un número de unidades de tiempo igual al número de datos que recibe o envía un proceso en cada nivel del grafo de tareas (tenga en cuenta la asignación de tareas a procesos que ha considerado en el apartado a)).
2) Los procesadores que realizan las tareas de las hojas del árbol tienen acceso sin coste de comunicación a los datos que utilizan dichas tareas.

Fijándonos en el grafo de tareas del apartado a), tenemos que:

$$T_{c,s}(16,8) = 1+1+1 = 3$$

En general:

$$\boxed{T_{c,s}(n,p) = \log_2(p)}$$

d) Suponiendo que el tiempo de sobrecarga coincide con el tiempo de comunicación calculado en c), obtener la ganancia en prestaciones.

$$S(n,p) = \frac{T_s(n)}{T_p(n,p)} = \frac{T_s(n)}{T_c(n,p) + T_o(n,p)} = \frac{n-1}{\left(\frac{n}{p}-1\right) + \log_2(p) + \log_2(p)} \Rightarrow$$

$$\Rightarrow \boxed{S(n,p) = \frac{n-1}{\left(\frac{n}{p}-1\right) + 2\log_2(p)}}$$

e) Obtener el número de procesadores para el que se obtiene la máxima ganancia con n números.

Tenemos que:

$$T_p(n,p) = \left(\frac{n}{p}-1\right) + 2\log_2(p)$$

Para hallar el número de procesadores con el que se obtiene la máxima ganancia para n números:

$$T_p'(p) = -\frac{n}{p^2} + \frac{2}{p\ln(2)}$$

$$T_p'(p) = 0 \Leftrightarrow \frac{n}{p^2} = \frac{2}{p\ln(2)} \Leftrightarrow \frac{n}{p} = \frac{2}{\ln(2)} \Leftrightarrow p = \frac{n\cdot\ln(2)}{2}$$

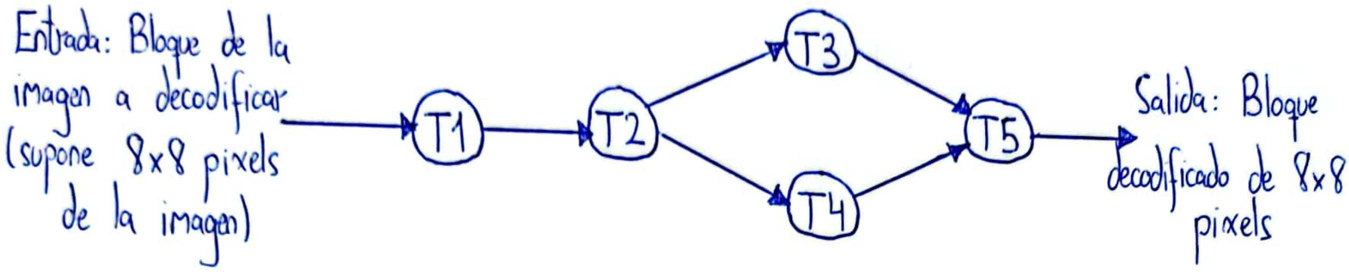
Veamos que es un máximo:

$$T_p''(p) = \frac{2n}{p^3} - \frac{2}{p^2\ln(2)}$$

$$T_p''\left(p = \frac{n\cdot\ln(2)}{2}\right) = \frac{2n}{\left(\frac{n\cdot\ln(2)}{2}\right)^3} - \frac{2}{\left(\frac{n\cdot\ln(2)}{2}\right)^2\cdot\ln(2)} = \frac{2}{\frac{n^2\cdot(\ln(2))^3}{8}} - \frac{2}{\frac{n^2\cdot(\ln(2))^3}{4}} = \frac{8}{n^2\cdot(\ln(2))^3} > 0 \Rightarrow$$

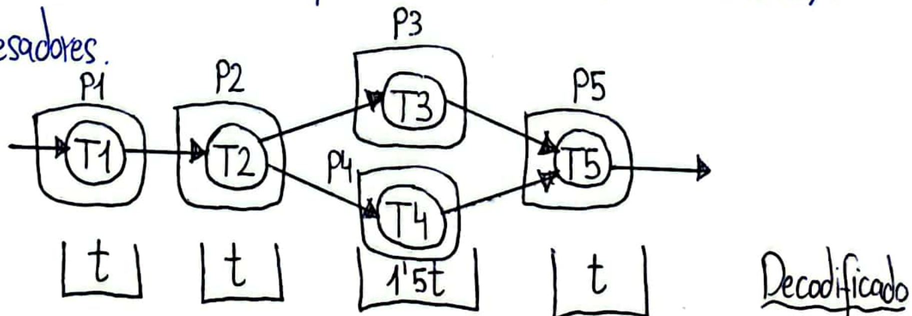
$\Rightarrow$ $p = \frac{n\cdot\ln(2)}{2}$ es un máximo.

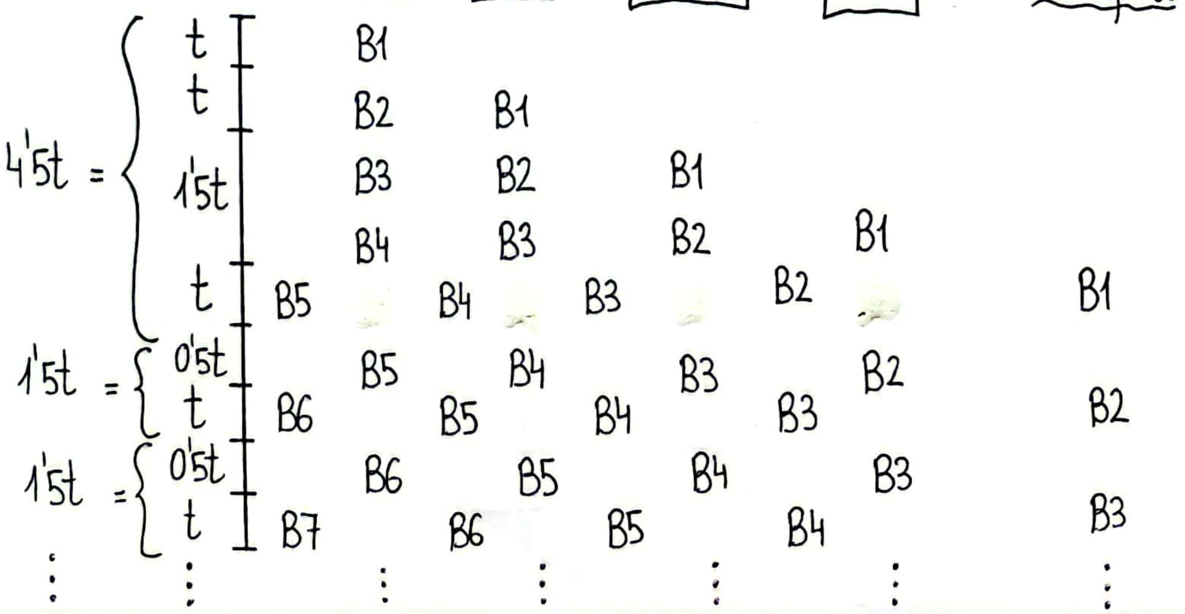
9) Se va a paralelizar un decodificador JPEG en un multiprocesador. Se ha extraído para la aplicación el siguiente grafo de tareas que presenta una estructura segmentada (o de flujo de datos):




Las tareas 1,2 y 5 se ejecutan en un tiempo igual a t, mientras que las tareas 3 y 4 suponen 1'5t. El decodificador JPEG aplica el grafo de tareas de la figura a bloques de la imagen, cada uno de 8x8 píxeles. Si se procesa una imagen que se puede dividir en n bloques de 8x8 píxeles, a cada uno de esos n bloques se aplica el grafo de tareas de la figura. Obtenga la mayor ganancia en prestaciones que se puede conseguir paralelizando el decodificador JPEG (suponiendo que se procesa una imagen con un total de n bloques de 8x8 píxeles) en (suponga despreciable el tiempo de comunicación/sincronización):

a) 5 procesadores.

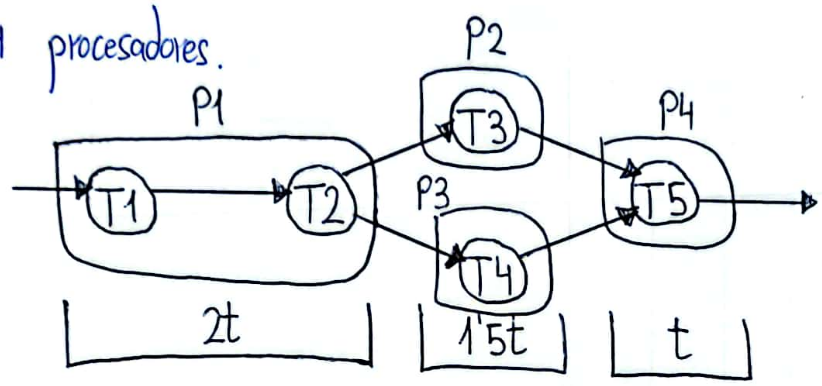


$$T_p(5,n) = \underbrace{4'5t}_{1^a\text{ Entrada}} + (n-1)\cdot 1'5t = 3t + n\cdot 1'5t$$

$$S(5,n) = \frac{T_s}{T_p(5,n)} = \frac{n\cdot 6\cancel{t}}{3\cancel{t} + n\cdot 1'5\cancel{t}} \Rightarrow \boxed{S(5,n) = \frac{4n}{2+n} \xrightarrow[n\to\infty]{} S_{máx} = 4}$$

b) 4 procesadores.




Siguiendo el mismo esquema que en el apartado a), llegamos al siguiente resultado:

$$T_p(4,n) = 4'5t + (n-1)\cdot 2t = 2'5t + n\cdot 2t$$

$$S(4,n) = \frac{T_s}{T_p(4,n)} = \frac{n\cdot 6\cancel{t}}{2'5\cancel{t} + n\cdot 2\cancel{t}} \Rightarrow \boxed{S(4,n) = \frac{6n}{2'5+2n} \xrightarrow[n\to\infty]{} S_{máx} = 3}$$

10 Se quiere implementar un programa paralelo para un multicomputador que calcule la siguiente expresión para cualquier $x$ (es el polinomio de interpolación de Lagrange):

$$P(x) = \sum_{i=0}^{n} \left(b_i \cdot L_i(x)\right),$$

donde:

$$L_i(x) = \frac{(x-a_0)\cdots(x-a_{i-1})(x-a_{i+1})\cdots(x-a_n)}{k_i} = \frac{\prod_{\substack{j=0\\ j\neq i}}^{n}(x-a_j)}{k_i} \qquad \forall i = 0,1,\dots,n.$$

$$k_i = (a_i-a_0)\cdots(a_i-a_{i-1})(a_i-a_{i+1})\cdots(a_i-a_n) = \prod_{\substack{j=0\\ j\neq i}}^{n}(a_i-a_j) \qquad \forall i = 0,1,\dots,n.$$

Inicialmente, $k_i$, $a_i$ y $b_i$ se encuentran en el nodo $i$ y $x$ en todos los nodos. Solo se van a usar funciones de comunicación colectivas. Indique cuál es el número mínimo de funciones colectivas que se pueden usar, cuáles serían, en qué orden se utilizarían y para qué se usan en cada caso.

Los pasos del algoritmo para $n=3$ serían los siguientes:

| Procesador | Situación inicial | | | | (1) Resta paralela $A_i = (x-a_i)$ ($(x-a_i)$ se obtiene en $P_i$) | (2) Todos reducen $A_i$ con resultado en $B_i$: $B_i = \prod_{i=0} A_i$ |
|---|---|---|---|---|---|---|
| P0 | $a_0$ | $x$ | $k_0$ | $b_0$ | $(x-a_0)$ | $(x-a_0)(x-a_1)(x-a_2)(x-a_3)$ |
| P1 | $a_1$ | $x$ | $k_1$ | $b_1$ | $(x-a_1)$ | $(x-a_0)(x-a_1)(x-a_2)(x-a_3)$ |
| P2 | $a_2$ | $x$ | $k_2$ | $b_2$ | $(x-a_2)$ | $(x-a_0)(x-a_1)(x-a_2)(x-a_3)$ |
| P3 | $a_3$ | $x$ | $k_3$ | $b_3$ | $(x-a_3)$ | $(x-a_0)(x-a_1)(x-a_2)(x-a_3)$ |

| Procesador | (3) Cálculo de todos los $L_i(x)$ en paralelo: $L_i = B_i / (A_i \cdot k_i)$ ($L_i(x)$ se obtiene en $P_i$) | (4) Cálculo en paralelo de todos los $C_i = b_i \cdot L_i$ ($b_i \cdot L_i(x)$ se obtiene en $P_i$) | (5) Reducción del contenido de $C_i$ con resultado en P0 ($P(x)$ se obtiene en P0) |
|---|---|---|---|
| P0 | $(x-a_1)(x-a_2)(x-a_3)/k_0$ | $b_0 \cdot (x-a_1)(x-a_2)(x-a_3)/k_0$ | $P = \sum_{i=0}^{n} C_i = \sum_{i=0}^{n} (b_i \cdot L_i)$ |
| P1 | $(x-a_0)(x-a_2)(x-a_3)/k_1$ | $b_1 \cdot (x-a_0)(x-a_2)(x-a_3)/k_1$ | |
| P2 | $(x-a_0)(x-a_1)(x-a_3)/k_2$ | $b_2 \cdot (x-a_0)(x-a_1)(x-a_3)/k_2$ | |
| P3 | $(x-a_0)(x-a_1)(x-a_2)/k_3$ | $b_3 \cdot (x-a_0)(x-a_1)(x-a_2)/k_3$ | |

Como se puede ver en el trazado del algoritmo para $n=3$ mostrado en las tablas, se usan un total de 2 funciones de comunicación colectivas (pasos (2) y (5) en la tabla). En el paso (2) del algoritmo se usa una operación de "todos reducen" para obtener en todos los procesadores los productos de todas las restas $(x-a_i)$. En el paso (5), y último, se realiza una operación de reducción para obtener las sumas de todos los productos $(b_i \cdot L_i)$ en el proceso 0.