TEMA 2: PROGRAMACIÓN PARALELA

LECCIÓN 3: EVALUACIÓN DE PRESTACIONES EN PROCESAMIENTO PARALELO

1. Ganancia en prestaciones y escalabilidad

1.1 Medidas usuales para evaluar prestaciones

- Tiempo de respuesta (ejecución): Real (*elapsed time*)

  Usuario, sistema, CPU time = user + sys
- Productividad (throughput): Nº de entradas que el computador procesa por unidad de tiempo.

Dependiendo de la utilización del sistema, se dará más importancia a una medida u otra.

1.2 Otras medidas

- Escalabilidad
- Eficiencia
  - Relación prestaciones/prestaciones máximas
  - Rendimiento (prestaciones / nº recusos)
  - Otras relaciones:
    - Prestaciones / consumo_potencia
    - Prestaciones / área_ocupada

1.3 Ganancia en Prestaciones. Escalabilidad

- **Ganancia en prestaciones**

Se utiliza en velocidad, para estudiar en qué medida incrementan las prestaciones al ejecutar una aplicación en paralelo en un sistema con múltiples procesadores, frente a su ejecución con un solo procesador.

Esto interesa para ver la ganancia en velocidad que se alcanza al aplicar paralelismo.

$$S(p) = \frac{Prestaciones(p)}{Prestaciones(1)}$$

Utilizando el tiempo de respuesta para evaluar las prestaciones, éstas vendrían dadas por la inversa del tiempo; entonces:

$$S(p) = \frac{T_s}{T_p(p)}$$

Donde:

- Ts es el tiempo de ejecución del programa secuencial.
- Tp(p) es el tiempo de ejecución del programa paralelo con *p* procesadores

El tiempo de ejecución en paralelo no depende solo del tiempo de cómputo en paralelo de las tareas detectadas en la aplicación **[Tc(p)]**, sino también de un tiempo de sobrecarga **[To(p)]**

En el tiempo de **sobrecarga (overhead)** influyen varios factores como:

- Comunicación/sincronización entre procesos
- Crear/terminar (enrolar/desenrolar) los procesos
- Cálculos o funciones añadidos a la versión paralela no presentes en la secuencial
- Falta de equilibrio

- **Escalabilidad**

La representación de la **ganancia en función del nº de procesadores** nos permite evaluar la escalabilidad.

Por lo tanto, la escalabilidad representa la ganancia en prestaciones que se consigue en un sistema conforme se añaden procesadores (aunque también se puede tener en cuenta otros recursos)

Ganancia en prestaciones:

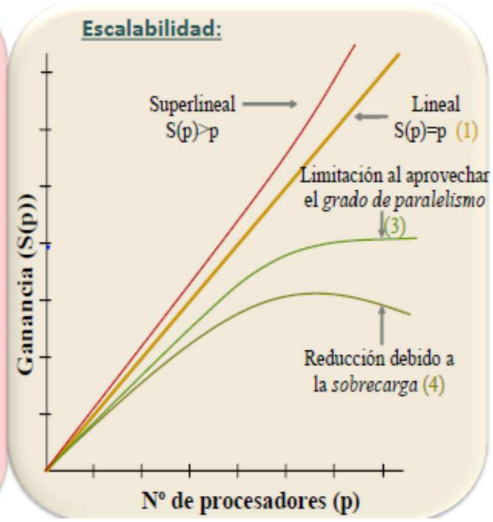
$$S(p) = \frac{\text{Prestaciones}(p)}{\text{Prestaciones}(1)} = \frac{T_S}{T_P(p)}$$

Ganancia en velocidad (Speedup)

$$T_P(p) = T_C(p) + T_O(p)$$

Sobrecarga *(Overhead)*:

- Comunicación/sincronización.
- Crear/terminar procesos/threads.
- Cálculos o funciones no presentes en versión secuencial.
- Falta de equilibrado.



Conforme a la escalabilidad, vemos que en procesamiento paralelo no siempre es más rentable usar más procesos/hebras que procesadores.

Ganancia máxima en prestaciones

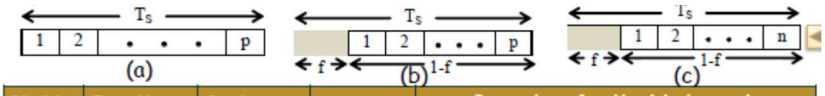
Idealmente, se debería disminuir el tiempo de ejecución a Ts/p con *p* procesadores.

Para conseguir esto, se debería poder distribuir todo el programa secuencial en partes iguales entre los procesadores, y ser el tiempo de sobrecarga despreciable. Así la ganancia sería **lineal** [S = p(1)].

En ocasiones podemos encontrar una escalabilidad **superlineal** [S(p) > p]. Normalmente, esto se debe a que al aumentar el nº de procesadores, también aumentamos el nº de otros recursos como caché o memoria principal.

Además, podemos encontrar otros casos, como vemos en el gráfico anterior.

Ejemplos de diferentes expresiones de Ganancia de Velocidad para diferentes modelos de código secuencial.



Modelo código	Fracción no paral. en T_S	Grado paralelismo	Overhead	Ganancia en función del número de procesadores *p* con T_S constante
(a)	0	ilimitado	0	$S(p) = \frac{T_S}{T_P(p)} = p$ Ganancia lineal (1)
(b)	f	ilimitado	0	$S(p) = \frac{1}{f + \frac{(1-f)}{p}} \xrightarrow{p\to\infty} \frac{1}{f}$ (2)
(c)	f	n	0	$S(p) = \frac{1}{f + \frac{(1-f)}{p}} \xrightarrow{p=n} \frac{1}{f + \frac{(1-f)}{n}}$ (3)
(b)	f	ilimitado	Incrementa linealmente con *p*	$S(p) = \frac{1}{f + \frac{(1-f)}{p} + \frac{T_O(p)}{T_S}} \xrightarrow{p\to\infty} 0$ (4)

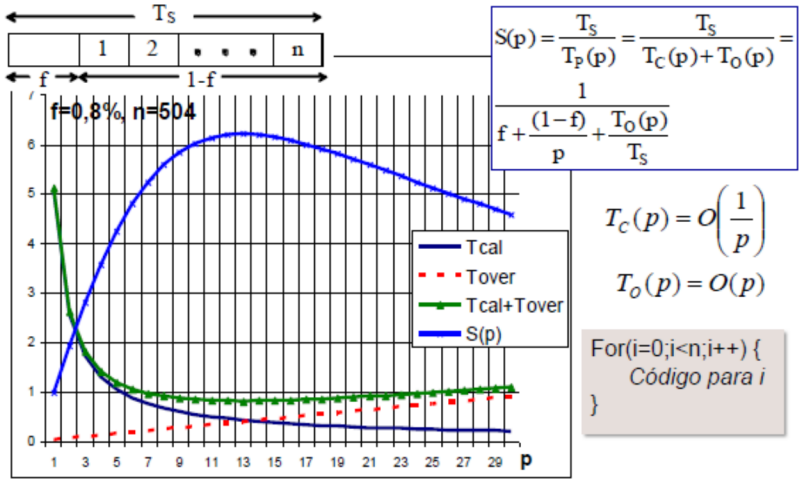
(a) La parte no paralelizable es despreciable (es 0) y el código siempre se puede dividir en partes iguales entre los *p* procesadores utilizados. No se tiene en cuenta overhead. Corresponde con la expresión (1) en el gráfico: ganancial lineal (ideal).

(b) La parte no paralelizable es *f*, y el código paralelizable siempre se puede dividir en partes iguales entre los *p* procesadores utilizados. No se tiene en cuenta sobrecarga.

(c) La parte no paralelizable es *f*, y el grado de paralelismo es igual a *n*, es decir, el paralelismo es limitado. Corresponde con la expresión (3) en el gráfico. Aquí la ganancia se limita cuando se aprovecha todo el grado de paralelismo.

(d) Igual que en (b), pero se tiene en cuenta la sobrecarga, que incrementa linealmente con *p*, por lo tanto, la ganancia acabará siendo 0 si ponemos muchos procesadores. Corresponde con la expresión (4) en el gráfico.

Número de procesadores óptimo

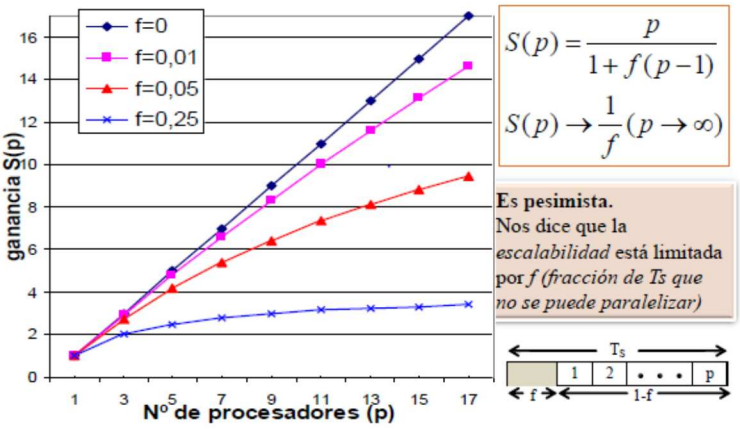


2. Ley de Amdahl

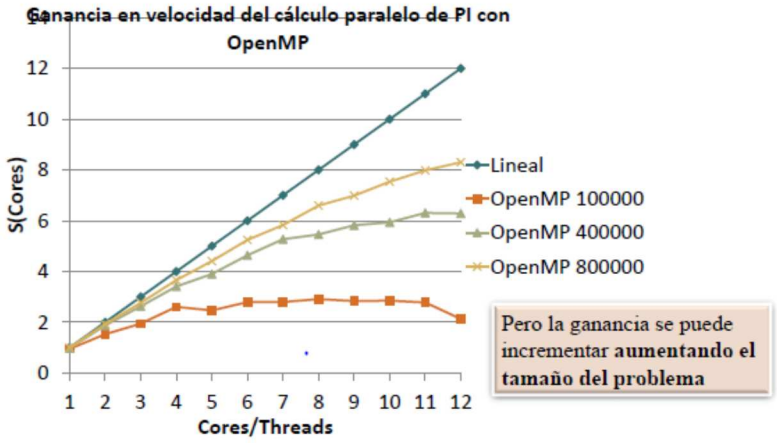
Podemos deducir del apartado anterior, la ganancia disminuye conforme aumenta la fracción de código que no se puede paralelizar.

Por lo tanto, la ganancia en prestaciones utilizando *p* procesadores está **limitada** por la **fracción de código no paralelizable.**

Ganancia – Fracción no paralelizable



Ganancia Escalable



3. Ganancia escalable. Ley de Gustafson

Cuando llegamos a un nivel aceptable de tiempo de ejecución paralelo, el siguiente objetivo es aumentar el tamaño del problema para mejorar otros aspectos, como por ejemplo aumentar la precisión del resultado.

Supongamos que tenemos un código secuencial con una parte no paralelizable y un número de tareas a paralelizar *n*, que se puede incrementar aumentando el tamaño del problema. Si consideramos despreciable el tiempo de sobrecarga (overhead), podemos **mantener constante el tiempo de ejecución paralelo**, y variar el número de procesadores *p* y el tamaño *n*, de forma que $n = kp$, siendo *k* una constante.

En este caso, el tiempo de ejecución secuencial depende de *n* [T*s(n)*]. Así, la ganancia de prestaciones sería: