

# **Eficiencia de programas paralelos**

Victorio E. Sonzogni

CIMEC Centro Internacional de Métodos Computacionales en Ingeniería

INTEC, CONICET-UNL

FICH, UNL

Santa Fe, Argentina

# Eficiencia de programas paralelos

- La ejecución en paralelo exige, generalmente, un tiempo de CPU mayor que el de la ejecución secuencial.
- En cálculo paralelo interesa el tiempo transcurrido, total, para ejecutar el programa. Esto también importa en el caso de cálculos en tiempo real.
- Para un programa que corre en paralelo, con  $p$  procesadores, se introducen dos definiciones: “aceleración” (*speedup*) y *eficiencia*.

# Eficiencia de programas paralelos

## ● *Speedup*

$$S_p = \frac{t_s}{t_p}$$

donde

- $t_s$  : tiempo para procesarlo secuencialmente;
- $t_p$  : tiempo para procesarlo en paralelo con  $p$  procesadores.

## ● eficiencia

$$E_p = \frac{S_p}{p} = \frac{t_s}{p t_p}$$

# Eficiencia de programas paralelos

En condiciones ideales:

$$S_p \rightarrow p$$

$$E_p \rightarrow 1$$

En la práctica:

$$S_p \leq p$$

$$E_p \leq 1$$

# Eficiencia de programas paralelos

Efectividad:

$$F_p = \frac{S_p}{p t_p} = \frac{S_p}{C_p}$$

donde

$$C_p = p t_p$$

representa el “costo computacional”.

$F_p$  associates  $S_p$  and  $E_p$  :

$$F_p = \frac{E_p}{t_p} = \frac{E_p S_p}{t_s}$$

Se puede buscar maximizar  $S_p$  o bien  $E_p$ , pero no ambos simultaneamente.

# Ejemplo de eficiencia en programas paralelos

## Ejemplo

Algoritmo en paralelo para sumar 16 números reales  
( $a_i, i = 1, \dots, 16$ ).

Diferentes estrategias:

- Procesamiento secuencial (o paralelo con 1 proc. **p=1**)

$$S = a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16}$$

precisa 15 sumas. Suponiendo 1 unidad de tiempo por operación, el cálculo con 1 procesador insume  $t_1 = 15$ .

# Ejemplo de eficiencia en programas paralelos

## ● **p=2**

Se puede organizar el cálculo en dos fases:

Primera fase:

● en procesador 1:  $S_1 = a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8$

● en procesador 2:  $S_2 = a_9 + a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15} + a_{16}$

Segunda fase:

● en procesador 1:  $S = S_1 + S_2$

En la primera fase: 7 sumas (en paralelo); en la segunda: 1 suma:

Tiempo total:  $t_2 = 7 + 1 = 8$ .

# Ejemplo de eficiencia en programas paralelos

## ● p=3

En este caso se puede organizar:

Primera fase:

- en procesador 1:  $S_1 = a_1 + a_2 + a_3 + a_4 + a_5$
- en procesador 2:  $S_2 = a_6 + a_7 + a_8 + a_9 + a_{10}$
- en procesador 3:  $S_3 = a_{11} + a_{12} + a_{13} + a_{14} + a_{15}$

Segunda fase:

- en procesador 1:  $b_1 = S_1 + S_2$
- en procesador 2:  $b_2 = S_3 + a_{16}$

Tercera fase:

- en procesador 1:  $S = b_1 + b_2$

En la primera fase: 4 sumas(en paralelo); en la segunda: 1 suma (en paralelo); y en la tercera: 1 suma.

$$t_3 = 4 + 1 + 1 = 6.$$



# Ejemplo de eficiencia en programas paralelos

## ● p=4

En este caso se puede organizar:

Primera fase:

- en procesador 1:  $S_1 = a_1 + a_2 + a_3 + a_4$
- en procesador 2:  $S_2 = a_5 + a_6 + a_7 + a_8$
- en procesador 3:  $S_3 = a_9 + a_{10} + a_{11} + a_{12}$
- en procesador 4:  $S_4 = a_{13} + a_{14} + a_{15} + a_{16}$

Segunda fase:

- en procesador 1:  $b_1 = S_1 + S_2$
- en procesador 2:  $b_2 = S_3 + S_4$

Tercera fase:

- en procesador 1:  $S = b_1 + b_2$

En la primera fase: 3 sumas(en paralelo); en la segunda: 1 suma (en paralelo); y en la tercera: 1 suma.

$$t_4 = 3 + 1 + 1 = 5.$$

# Ejemplo de eficiencia en programas paralelos

● **p=8**

Primera fase:

- en procesador 1:  $S_1 = a_1 + a_2$
- en procesador 2:  $S_2 = a_3 + a_4$
- en procesador 3:  $S_3 = a_5 + a_6$
- en procesador 4:  $S_4 = a_7 + a_8$
- en procesador 5:  $S_5 = a_9 + a_{10}$
- en procesador 6:  $S_6 = a_{11} + a_{12}$
- en procesador 7:  $S_7 = a_{13} + a_{14}$
- en procesador 8:  $S_8 = a_{15} + a_{16}$


Segunda fase:


- en procesador 1:  $b_1 = S_1 + S_2$
- en procesador 2:  $b_2 = S_3 + S_4$
- en procesador 3:  $b_3 = S_5 + S_6$
- en procesador 4:  $b_4 = S_7 + S_8$

# Ejemplo de eficiencia en programas paralelos


 **p=8**

Tercera fase:

 en procesador 1:  $c_1 = b_1 + b_2$

 en procesador 2:  $c_2 = b_3 + b_4$

Cuarta fase:

 en procesador 1:  $S = c_1 + c_2$

En la primera fase: 1 sumas(en paralelo); en la segunda: 1 suma (en paralelo); y en la tercera: 1 suma(en paralelo).

El tiempo total es:  $t_8 = 1 + 1 + 1 + 1 = 4$

# Ejemplo de eficiencia en programas paralelos

$p$	$t_p$	$C_p = p t_p$	$S_p$	$E_p$	$F_p = \frac{E_p S_p}{t_1}$
1	15	15	1	1	0.0666
2	8	16	1.88	0.94	0.1172
3	6	18	2.5	0.83	0.1389
4	5	20	3	0.75	0.15
8	4	32	3.75	0.47	0.1172

# Computadoras de memoria local

- En memoria local el tiempo secuencial  $t_s$ , o  $t_1$ , no se puede calcular. Las fórmulas para  $S_p$  y  $E_p$  no pueden aplicarse.
- Se estima: tiempo de cálculo ( $t_{cal}$ ) y el de comunicación ( $t_{com}$ ) y coordinación ( $t_{coo}$ )  
El tiempo para resolver el problema en paralelo con  $p$  procesadores:

$$t_p = t_{cal} + t_{com} + t_{coo}$$

y

$$t_{cal} = \frac{t_s}{p} = \frac{t_1}{p}$$

# Computadoras de memoria local

## ● Speedup

$$S_p = \frac{t_s}{t_p} = \frac{t_s}{\frac{t_s}{p} + t_{com} + t_{coo}} = \frac{1}{\frac{1}{p} + \frac{t_{com} + t_{coo}}{t_s}} = \frac{p}{1 + p \frac{(t_{com} + t_{coo})}{t_s}} = \frac{p}{1 + \frac{(t_{com} + t_{coo})}{t_{cal}}}$$

## ● Eficiencia

$$E_p = \frac{S_p}{p} = \frac{1}{1 + \frac{t_{com} + t_{coo}}{t_{cal}}} = \frac{1}{1 + \omega}$$

donde

$$\omega = \frac{t_{com} + t_{coo}}{t_{cal}}$$

refleja la sobrecarga -o pérdida de eficiencia- debido al paralelismo.

# Computadoras de memoria local

$$\omega = \frac{t_{com} + t_{coo}}{t_{cal}}$$

tiene en cuenta:

- comunicación
- redundancia
- desequilibrio de carga
- gestión del paralelismo

(Estas fórmulas consideran que el programa es 100% paralelizable).

# Factores que afectan la eficiencia

La eficiencia puede ser vista como función de varios factores:

$$E_p = f(\eta_f, \eta_c, \eta_b, \eta_r)$$

donde

- $\eta_f$ : factor debido a la fracción no paralelizable del programa
- $\eta_c$ : factor debido a comunicación y coordinación
- $\eta_b$ : factor debido a desequilibrio de carga entre procesadores
- $\eta_r$ : factor debido a cálculos redundantes



# Fracción paralelizable del programa

En un programa hay tareas que pueden ser ejecutadas en paralelo y otra que no.

## Ejemplo:

Sumar  $N$  numeros, con  $p$  procesadores (supóngase  $\frac{N}{p} = k$  entero). Puede hacerse en dos pasos:

1. Suma parcial en cada procesador

$$S_i = \sum_{j=(i-1)k+1}^{ik} a_j \quad (i = 1, \dots, p)$$

2. Suma de las sumas parciales

$$S = \sum_{i=1}^p S_i$$

el primer paso puede hacerse en paralelo, el segundo secuencialmente.

# Fracción paralelizable del programa

Se puede definir una fracción paralelizable del programa ( $f_p$ ) y una serial (o secuencial) ( $f_s$ )

$$f_p + f_s = 1$$

El tiempo de procesamiento secuencial:

$$t_s = (1 - f_p) t_s + f_p t_s$$

y el de procesamiento paralelo con  $p$  procesadores:

$$t_p = (1 - f_p) t_s + \frac{f_p}{p} t_s$$

# Fracción paralelizable del programa

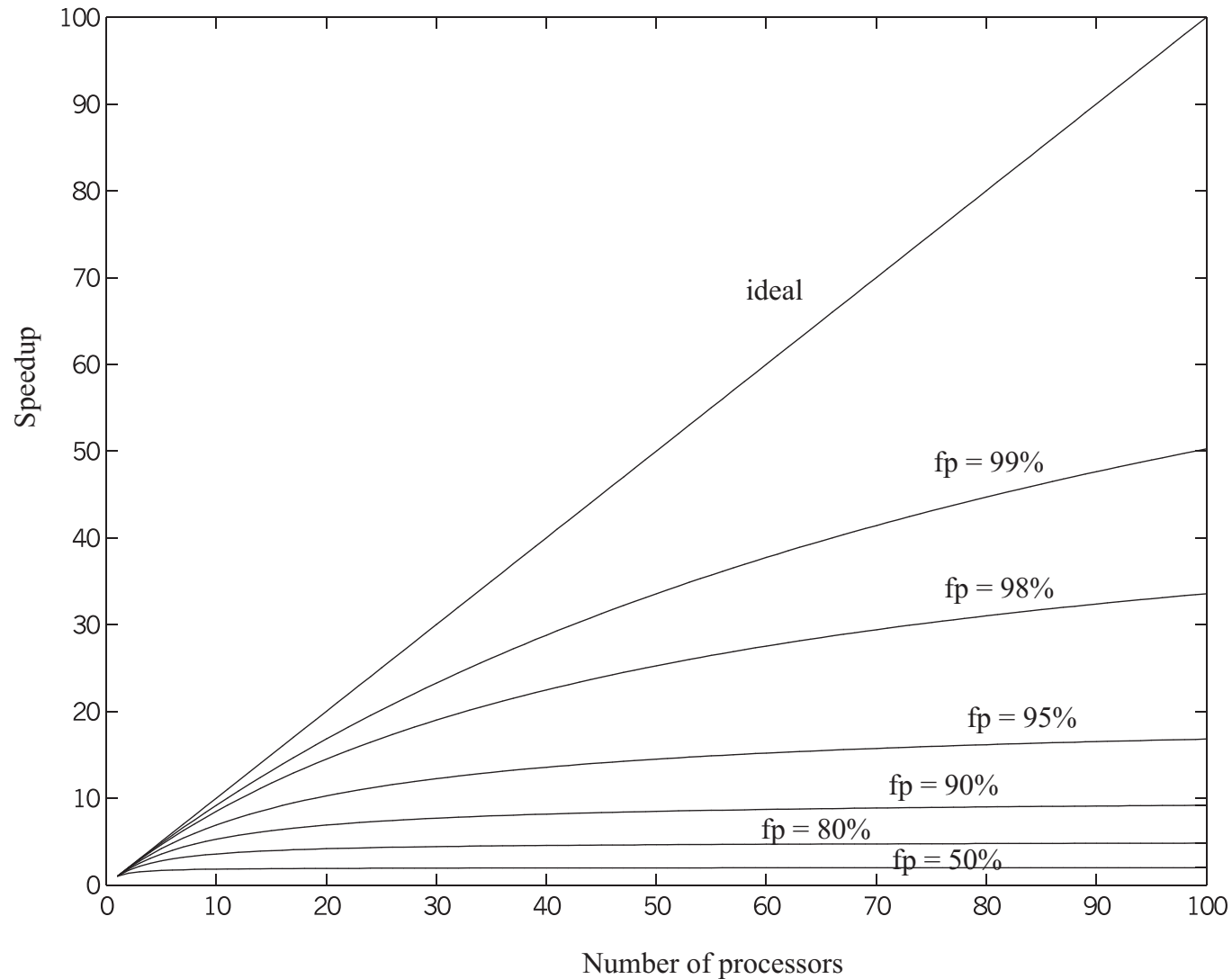
Speedup:

$$S_p = \frac{t_s}{t_p} = \frac{1}{(1 - f_p) + \frac{f_p}{p}} = \frac{p}{p(1 - f_p) + f_p}$$

Eficiencia:

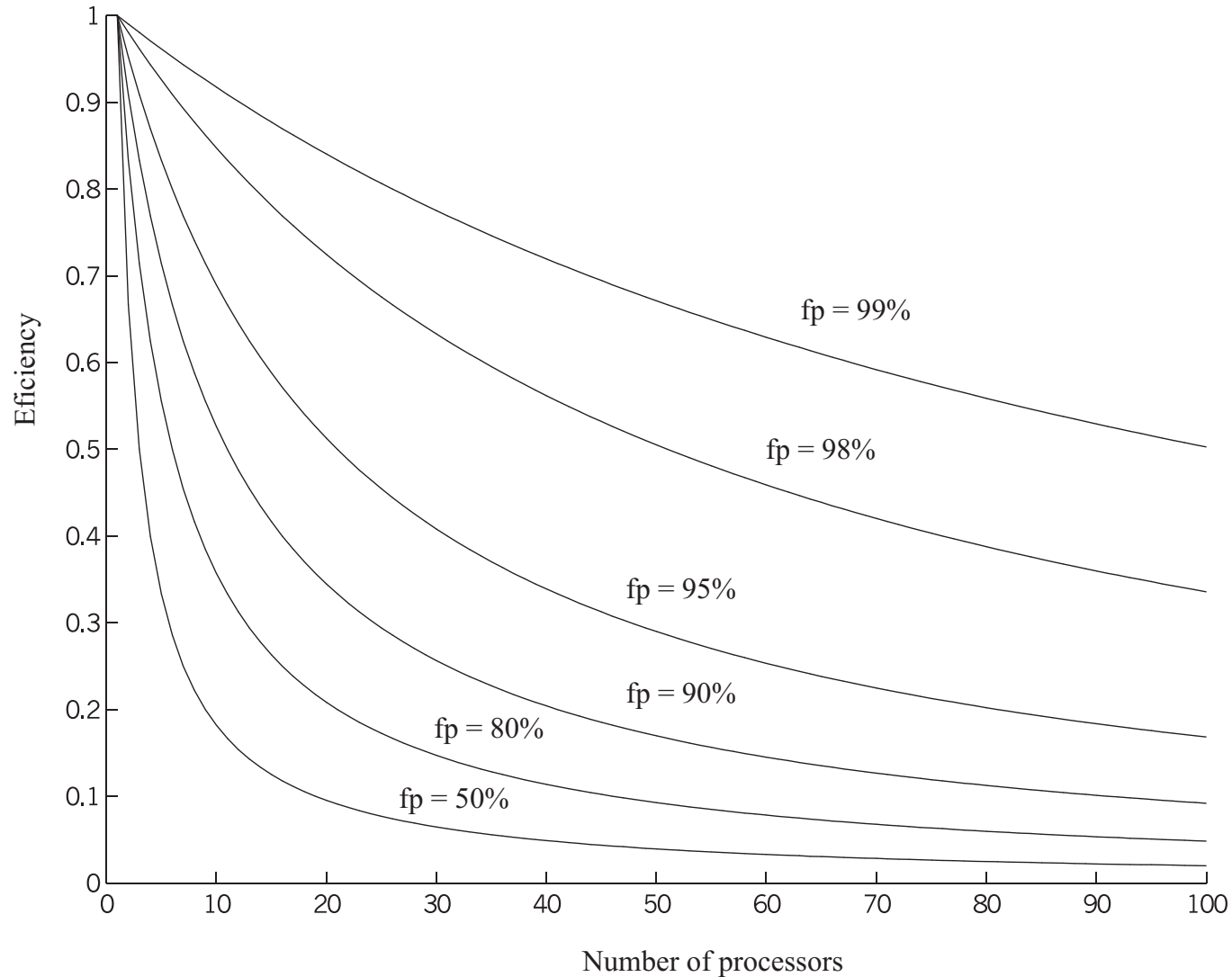
$$\eta_f = E_p = \frac{S_p}{p} = \frac{1}{p(1 - f_p) + f_p}$$

# Fracción paralelizable del programa



Ley de Amdahl. Speedup

# Fracción paralelizable del programa



Ley de Amdahl. Eficiencia

# Fracción paralelizable del programa

## Ley de Amdahl

Gene Amdahl (1967) observó que el  $S_p$  ideal se reduce por un factor  $\frac{1}{p(1-f_p)+f_p}$ .

Para  $p \rightarrow \infty$

$$S_p \rightarrow \frac{1}{f_s} = \frac{1}{1-f_p}$$

y

$$\eta_f \rightarrow 0$$

Por ejemplo: para fracción serial  $f_s = 10\%$  (o  $(f_p = 90\%)$  el speedup es  $S_p \leq 10$  para  $p \rightarrow \infty$

La eficiencia, debido a la fracción paralelizable,  $\eta_f$  disminuye cuando aumenta  $p$ .

# Fracción paralelizable del programa

## Ley de Amdahl

El Speedup también puede escribirse:  $S_p = \tilde{\eta} S_\infty$

donde

$$S_\infty = S_p_{p \rightarrow \infty} = \frac{1}{f_s} = \frac{1}{1 - f_p}$$

$$\tilde{\eta} = \frac{S_p}{S_\infty} = \frac{p}{p(1 - f_p) + f_p} (1 - f_p) = \frac{p}{p + \frac{f_p}{(1 - f_p)}} = \frac{p}{p + \frac{f_p}{f_s}} = \frac{1}{1 + \frac{p_c}{p}}$$

con

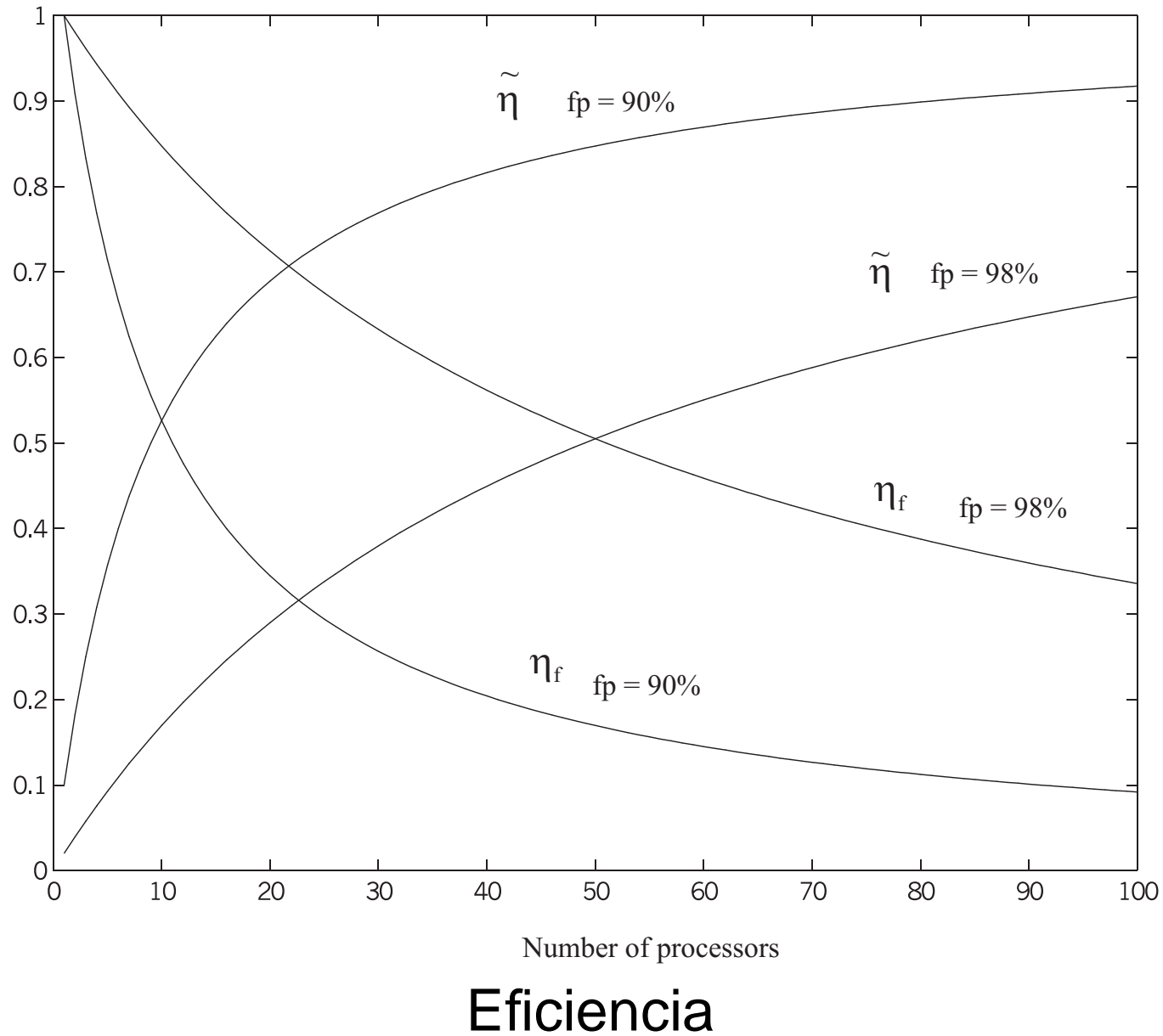
$$p_c = \frac{f_p}{f_s}$$

$p_c$  da la idea de un número razonable de procesadores para un programa dado, en función de  $f_p$ .

Si  $p = p_c$  el speedup es  $S_p = \frac{1}{2} S_\infty$

Para  $p > p_c$  el speedup decrece.

# Fracción paralelizable del programa





# Fracción paralelizable del programa

## Ley de Gustafson

- Ley de Amdahl presupone  $f_s$  independiente de  $p$ .
- Es válido para un problema de tamaño fijo.
- Al crecer  $p$  aumenta el tamaño del problema que se puede resolver.
- La fracción serial depende de la cantidad de operaciones ( $M$ ). Al aumentar  $M$ , disminuye  $f_s$ .
- Gustafson (1988) plantea así: cuánto tardaría un programa paralelo en correr secuencialmente.
- Sea  $f_s^*$  fracción serial de las operaciones en el programa paralelo con  $p$  procesadores.

# Fracción paralelizable del programa

## Ley de Gustafson

● Tiempo paralelo:  $t_p = f_s^* t_p + (1 - f_s^*) t_p$

● Tiempo serial:  $t_s = f_s^* t_p + p (1 - f_s^*) t_p$

● Speedup

$$S_p = \frac{t_s}{t_p} = f_s^* + p (1 - f_s^*) = p + f_s^* (1 - p)$$

# Fracción paralelizable del programa

## Ley de Gustafson

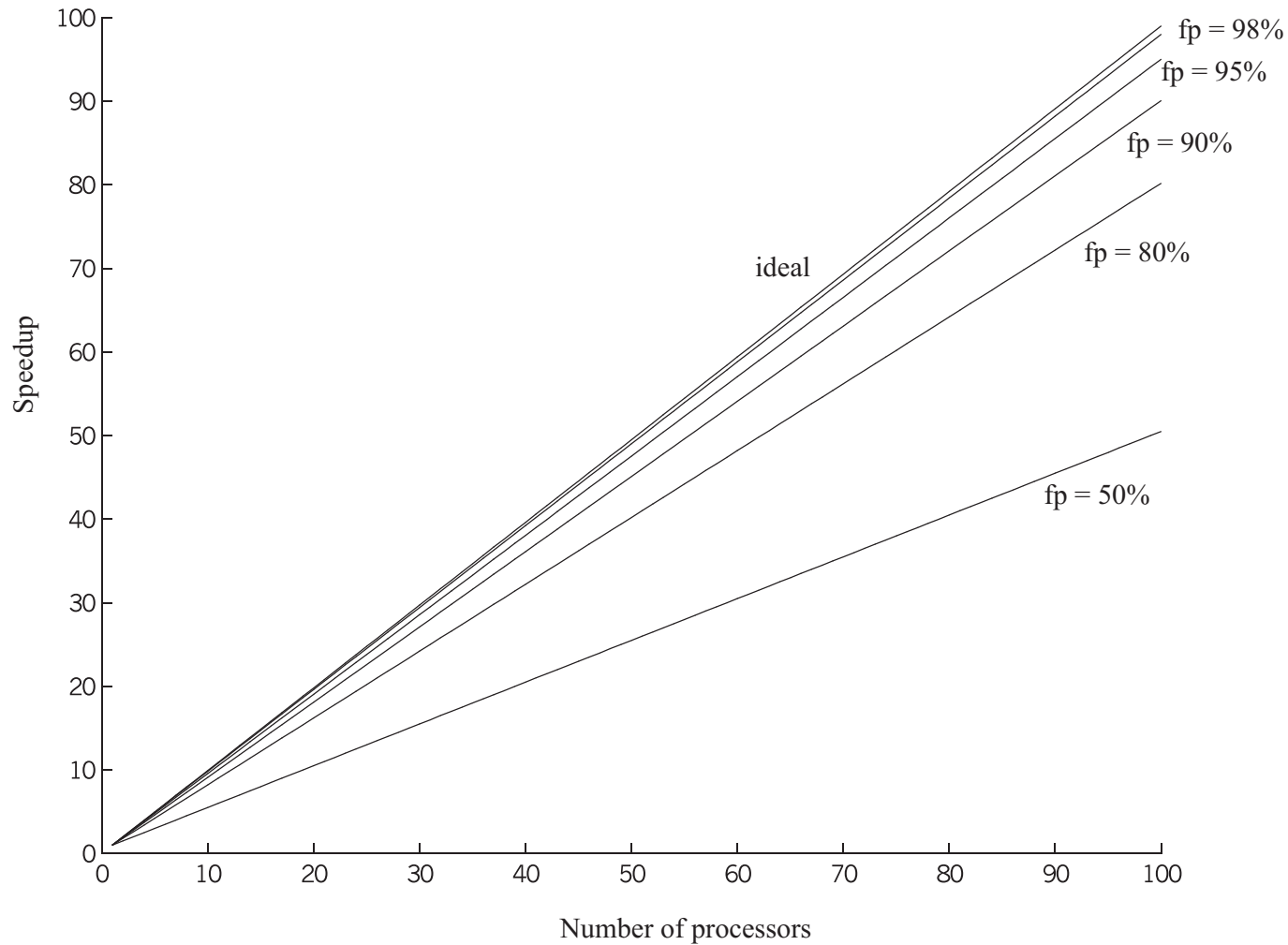
● Tiempo paralelo:  $t_p = f_s^* t_p + (1 - f_s^*) t_p$

● Tiempo serial:  $t_s = f_s^* t_p + p (1 - f_s^*) t_p$

● Speedup

$$S_p = \frac{t_s}{t_p} = f_s^* + p (1 - f_s^*) = p + f_s^* (1 - p)$$

# Fracción paralelizable del programa



Ley de Gustafson

# Comunicación y coordinación

Suponiendo que no hay otros factores que produzcan pérdida de eficiencia (fracción paralelizable, redundancia, equilibrio de carga), la eficiencia debido a comunicación y coordinación es:

$$\eta_c = \frac{1}{1 + \omega}$$

donde

$$\omega = \frac{t_{com} + t_{coo}}{t_{cal}}$$

El tiempo necesario para comunicación:  $t_{com} = \alpha + \beta l$

donde

$l$ : tamaño del mensaje (*bits* o *bytes*);

$\alpha$ : latencia (tiempo de inicialización) del mensaje;

$\beta = \frac{1}{\theta}$ : tiempo para comunicar 1 *bit* (o *byte*);

$\theta$ : "bandwidth" (ancho de banda) ( $\frac{bit}{s}$  o  $\frac{byte}{s}$ )

Para mandar  $m$  mensajes:  $t_{com} = m \alpha + \sum_{i=1}^m \beta l_i$

# Comunicación y coordinación

Succi y Papetti proponen un modelo para  $\omega$ , para aplicaciones en mecánica computacional:

$$\omega = \underbrace{a p^{\frac{1}{3}}}_{\text{bandwidth}} + \underbrace{b p}_{\text{latency}} + \underbrace{c \log_2 p}_{\text{coordin.}}$$

communic.

Los parámetros de la computadora:

$p$  : cantidad de procesadores

$\theta$  : ancho de banda

$\alpha$  : latencia

$r_1$  : velocidad de un procesador

y del programa de aplicación (E.F., D.F., etc.)

$N$  : tamaño del problema (incógnitas)

$\chi$  : densidad de cálculo (flops por nodo)

$\sigma$  : densidad de comunicación (*bytes* por nodo, a comunicar)

$\gamma$  : cantidad de nodos de interfaz (a ser comunicados)

# Comunicación y coordinación

Cada procesador realiza operaciones sobre  $n = \frac{N}{p}$  nodos y el tiempo de cálculo:

$$t_{cal} = \frac{\chi n}{r_1}$$

El tiempo de comunicación:

$$t_{com} = \sigma \frac{\gamma}{\theta} + m \alpha$$

Y el de coordinación:

$$t_{coo} = 2 N_s \alpha \phi(p)$$

$N_s$ : número de puntos de coordinación;

el número 2 hace referencia a "fork-join";

$\phi(p)$  depende de la comunicación global (en árbol:  $\phi(p) \simeq \log_2 p$ ).

La relación entre tiempos de comunicación y cálculo:

$$\frac{t_{com}}{t_{cal}} = \frac{\sigma \gamma r_1}{\theta \chi n} + \frac{m \alpha r_1}{\chi n}$$

# Comunicación y coordinación

Si los nodos están dispuestos en un arreglo cúbico en 3D, el numero de nodos en una arista es  $L = N^{\frac{1}{3}}$  y en una cara  $O(L^2) = O(N^{\frac{2}{3}})$

Para cada procesador el volumen de puntos:  $n = \frac{N}{p}$

La longitud asociada:  $l \simeq n^{\frac{1}{3}} = \frac{N^{\frac{1}{3}}}{p^{\frac{1}{3}}}$

y la superficie:  $s \simeq l^2 = n^{\frac{2}{3}} = \frac{N^{\frac{2}{3}}}{p^{\frac{2}{3}}} = \frac{L^2}{p^{\frac{2}{3}}}$

luego

$$\frac{\gamma}{n} \simeq \frac{L^2}{p^{\frac{2}{3}}} \frac{p}{N} = \frac{1}{L} p^{\frac{1}{3}}$$

se puede escribir

$$\frac{t_{com}}{t_{cal}} = \frac{\sigma r_1}{\chi \theta L} p^{\frac{1}{3}} + \frac{\alpha r_1 m}{\chi N} p = a p^{\frac{1}{3}} + b p$$

La relación de tiempos coordinación/cálculo:

$$\frac{t_{coo}}{t_{cal}} = \frac{2 N_s \alpha r_1}{\chi N} p \log_2 p = c p \log_2 p$$



# Comunicación y coordinación

Coeficiente  $a$ :

$$a = \frac{\frac{\sigma}{\chi}}{\frac{\theta}{r_1}} \cdot \frac{1}{L}$$

- $\sigma/\chi$  relación *comunicación/cálculo* del programa: *bytes* comunicados sobre operaciones (*flops*).
- $\theta/r_1$  es la misma relación pero para la máquina: el ancho de banda (*bits/s*) sobre (*flops/s*).
- EL coeficiente  $a$  debería ser pequeño. Una buena aplicación debería dar  $\sigma/\chi \simeq 0.1$ , o unos 10 *flops* por cada *byte* transmitidos (o 80 *flops* por variable).  $\theta/r_1 \simeq 0.1$  *bytes/flops*.

# Comunicación y coordinación

Coeficiente  $b$

$$b = \frac{m \gamma r_1}{\chi n}$$

- el numerador puede ser visto como los *flops* gastados por la latencia.
- el denominador representa los *flops* útiles para el cálculo

El coeficiente  $c$

$$c = \frac{2 \gamma N_s r_1}{\chi n}$$

- el numerador representa los *flops* gastados en coordinación,
- el denominador representa los *flops* útiles para el cálculo

# Equilibrio de carga

El tiempo para realizar  $p$  tareas en  $p$  procesadores está dado por el proceso que tarda más:

$$t_{max} = \max_i t_i \quad (i = 1, \dots, p)$$

donde  $t_i$  es el tiempo para el procesador  $i$ .

El tiempo promedio:

$$\bar{t} = \frac{1}{p} \sum_{i=1}^p t_i$$

El tiempo secuencial:

$$t_s = \sum_{i=1}^p t_i = p \bar{t}$$

El tiempo paralelo con  $p$  procesos:  $t_p = t_{max}$

Speedup:

$$S_p = \frac{t_s}{t_p} = \frac{p \bar{t}}{t_{max}}$$

Eficiencia:

$$\eta_b = E_p = \frac{S_p}{p} = \frac{\bar{t}}{t_{max}}$$

# Equilibrio de carga

Ejemplo:

Suma de  $N$  números ( $a_i, i = 1, \dots, N$ ) con  $p$  procesadores, donde  $N$  no es múltiplo de  $p$

$$N = p k + R \quad (1 < R < p - 1)$$

Se puede hacer:

●  $p - R$  procesadores hacen  $k$  cálculos

●  $R$  procesadores hacen  $k + 1$  cálculos

Si ( $N \gg p$ ) los tiempos medios y máximo ( $c_1$  tiempo para una operación):

$$\bar{t} \simeq k \cdot c_1 \quad t_{max} \simeq (k + 1) \cdot c_1$$

La eficiencia:

$$\eta_b = \frac{k}{k + 1} = \frac{\frac{N}{p}}{\frac{N}{p} + 1} \simeq 1$$

Si  $p$  aumenta, la granularidad de tareas ( $k = \frac{N}{p}$ ) disminuye y también la eficiencia  $\eta_b$

# Equilibrio de carga

En este ejemplo los cálculos son iguales para todos los procesadores. Los procesadores se suponen homogéneos.

La eficiencia  $\eta_b$  disminuye aún más si:

- la carga de operaciones en cada procesador es diferente. Esto ocurre frecuentemente en mecánica, por ejemplo en problemas no lineales.
- el sistema computacional es heterogéneo (procesadores con diferentes velocidades)

# Redundancia

Hay redundancia de cálculo cuando las mismas operaciones están siendo llevadas a cabo simultáneamente en diferentes procesadores.

La redundancia disminuye la eficiencia

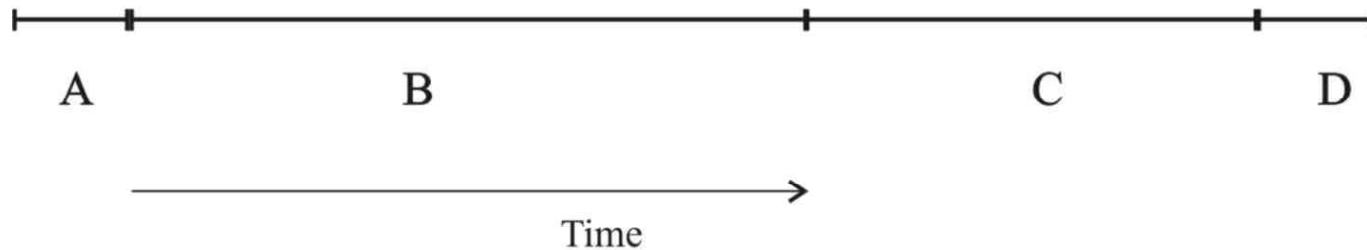
Sin embargo a veces se prefiere realizar algunos cálculos redundantes y evitar comunicaciones.

$$\eta_r = \frac{t_{cal_s}}{\sum_{i=1}^p t_{cal_i}}$$

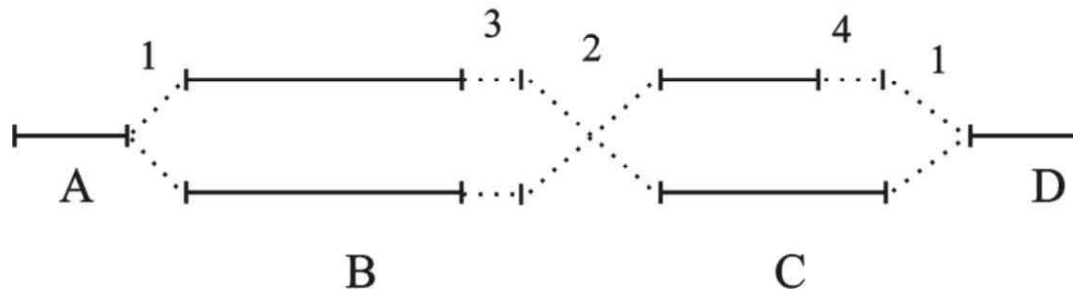
donde  $t_{cal_s}$  es el tiempo de cálculo del programa secuencial y  $t_{cal_i}$  es el tiempo de cálculo en paralelo, en el procesador  $i$ .

# Efecto combinado sobre la eficiencia

Sequential



Parallel



A y D: serial; B y C: paralelo; 1: coord.; 2: com.; 3: redund.; 4: t. ocioso (deseq. carga)

# Efecto combinado sobre la eficiencia

El tiempo de procesamiento paralelo, incluyendo: fracción paralelizable; comunicación y coord. y desequilibrio de cargas:

$$t_p = (1 - f_p) t_s + \frac{f_p t_s t_{max}}{p \bar{t}} + t_{com} + t_{coo}$$

La relación  $t_{max}/\bar{t}$  es una estimación de falta de equilibrio de carga.

Speedup:

$$S_p = \frac{t_s}{t_p} = \frac{1}{(1 - f_p) + \frac{f_p}{p} \frac{t_{max}}{\bar{t}} + \frac{t_{com} + t_{coo}}{t_s}} = \frac{1}{1 - f_p \left(1 - \frac{1}{p} \frac{t_{max}}{\bar{t}}\right) + \frac{t_{com} + t_{coo}}{t_s}}$$

Eficiencia:

$$E_p = \frac{S_p}{p} = \frac{1}{f_p \frac{t_{max}}{\bar{t}} + p(1 - f_p) + \frac{t_{com} + t_{coo}}{\frac{t_s}{p}}}$$



# Efecto combinado sobre la eficiencia

Caso particular: Carga balanceada

$$\frac{t_{max}}{\bar{t}} = 1$$

Considerando la existencia de una fracción paralelizable del programa y sobrecarga por comunicación y coordinación, el speedup y la eficiencia son:

Speedup:

$$S_p = \frac{1}{(1 - f_p) + \frac{f_p}{p} + \frac{t_{com} + t_{coo}}{t_s}}$$

Eficiencia:

$$E_p = \frac{1}{f_p + p(1 - f_p) + \frac{t_{com} + t_{coo}}{\frac{t_s}{p}}}$$

# Efecto combinado sobre la eficiencia

Caso particular: Program 100% paralelizable

Considerando la existencia de desequilibrio de carga y sobrecarga por comunicación y coordinación, el speedup y la eficiencia son:

Speedup:

$$S_p = \frac{1}{\frac{1}{p} \frac{t_{max}}{t} + \frac{t_{com} + t_{coo}}{t_s}}$$

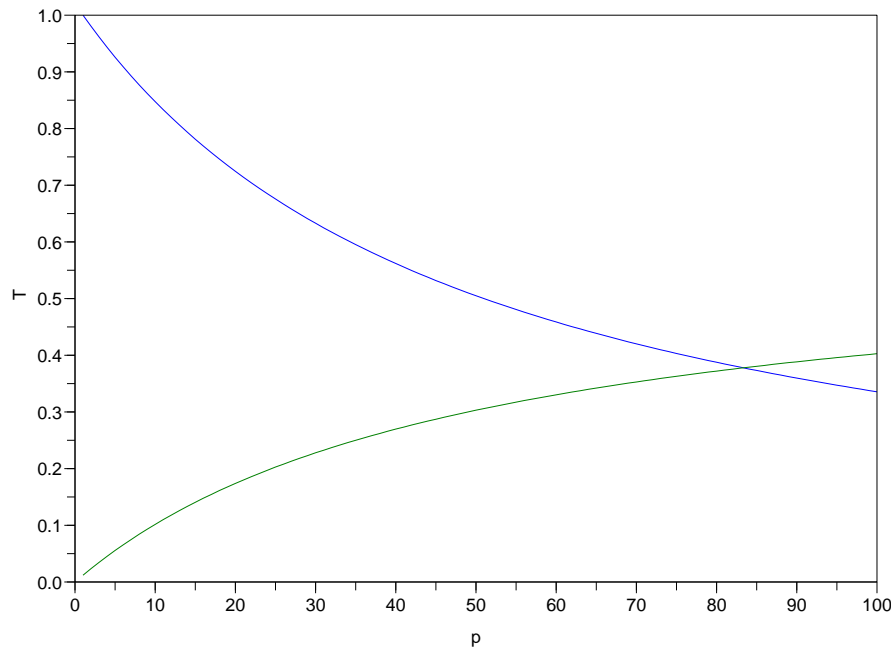
Eficiencia:

$$E_p = \frac{1}{\frac{t_{max}}{t} + \frac{t_{com} + t_{coo}}{\frac{t_s}{p}}}$$

# Escalabilidad

Para un problema de tamaño fijo, a medida que aumenta la cantidad de procesadores  $p$  el tiempo de cálculo disminuye. Pero las comunicaciones aumentan.

A partir de una determinada cantidad de procesadores se emplea más tiempo en comunicaciones que en cálculos.



# Escalabilidad

En problemas de tamaño variable (con  $p$ ), manteniendo al máximo la capacidad de cada procesador, el tiempo total de procesamiento se mantiene aproximadamente constante con  $p$ . Pero el tiempo de comunicaciones aumenta.

La *escalabilidad* es la propiedad de un programa paralelo, de mantener su eficiencia, cuando se aumenta la cantidad de procesadores.