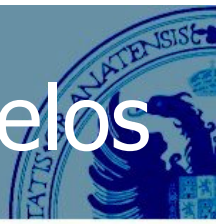
DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Eficiencia de algoritmos paralelos

Análisis y Diseño de Algoritmos

Eficiencia de algoritmos paralelos



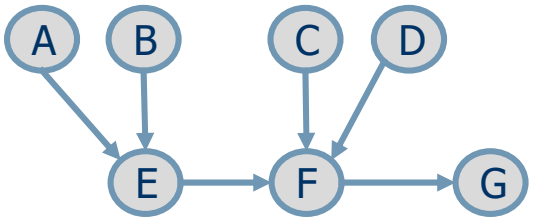
- Algoritmos paralelos
  - Modelo de programación: Grafos de tareas
  - Modelos computacionales, p.ej. PRAM
- Rendimiento de los algoritmos paralelos
  - Métricas de rendimiento
  - Ley de Amdahl
  - Ley de Gustafson
  - Isoeficiencia

Algoritmos paralelos

Modelo de programación: Grafos de tareas

Ejecución de múltiples fragmentos:

- Estado compartido.
- Fragmentos con estado privado.
- Usualmente asíncronos
  (la sincronía puede simplificar algunas cosas,
  p.ej. CUDA).



Algoritmos paralelos

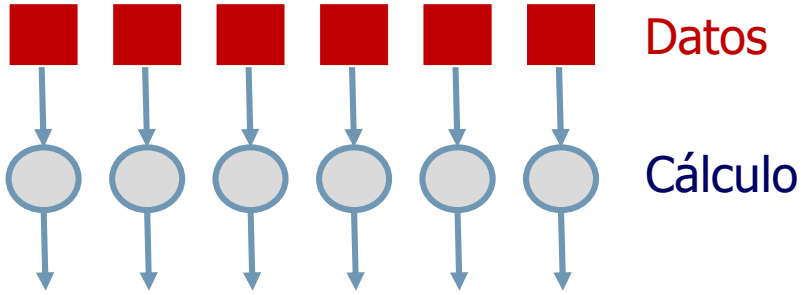
Modelo de programación:
Descomposición del problema

- Paralelismo de datos [data parallelism],
  i.e. calcular f(x) para muchos x en paralelo.
- Paralelismo de tareas [task parallelism],
  i.e. calcular muchas funciones f_i en paralelo.
- Pipeline

Algoritmos paralelos

Modelo de programación:
Descomposición del problema

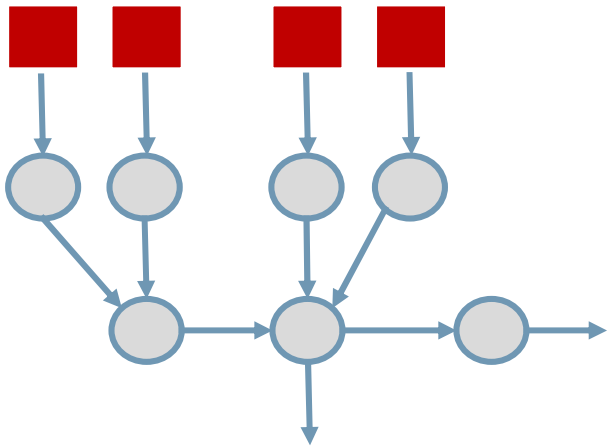
- Paralelismo de datos [data parallelism],
  i.e. calcular f(x) para muchos x en paralelo.



Algoritmos paralelos

Modelo de programación:
Descomposición del problema

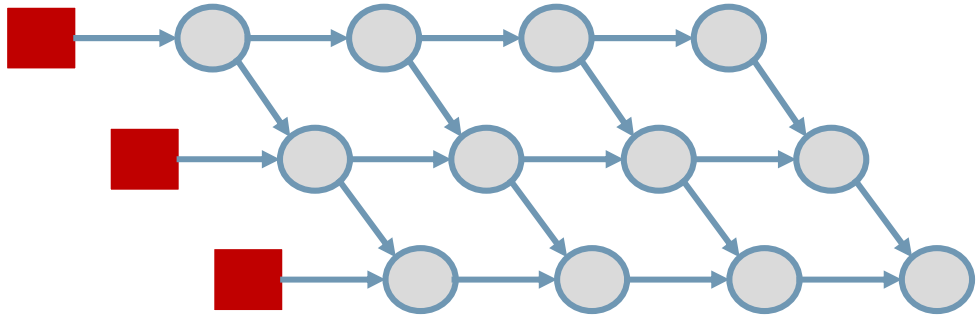
- Paralelismo de tareas [task parallelism],
  i.e. calcular muchas funciones f_i en paralelo.



Algoritmos paralelos

Modelo de programación:
Descomposición del problema

- Pipeline



Algoritmos paralelos

Modelos computacionales

- Simplifican el análisis de los algoritmos paralelos.
- Abstraen muchos detalles
  (necesarios para su implementación).
- Permiten evaluar el rendimiento
  de los algoritmos paralelos (antes de implementarlos).

Algoritmos paralelos

Modelos computacionales

TIPOS

- Memoria compartida/distribuida
- Comunicación síncrona/asíncrona

EJEMPLOS

- Actores
- CSP [Communicating Sequential Processes]
- PRAM [Parallel Random Access Machine]
- BSP [Bulk Synchronous Parallel]

Algoritmos paralelos

Modelos computacionales: Actores

Hewitt, Bishop & Steiger: "A Universal Modular Actor Formalism for Artificial Intelligence," IJCAI 1973

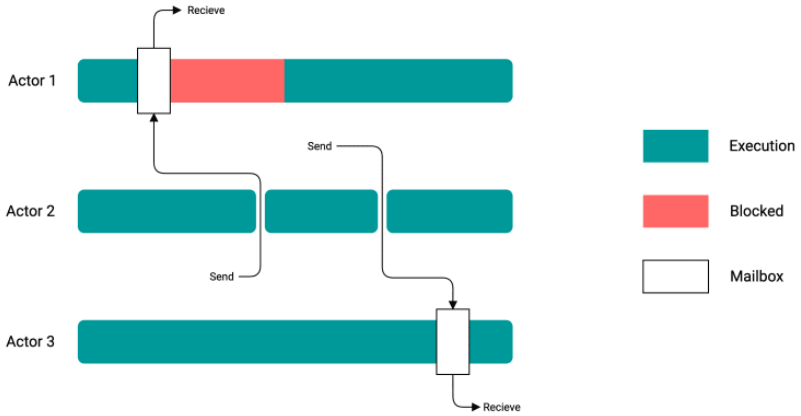
- Actor = Agente autónomo
- Sin estado compartido:
  interacción por medio de mensajes (asíncronos).
- Cada actor procesa los mensajes que recibe: actualiza su estado local (cálculo local), envía mensajes y puede crear otros actores.

Algoritmos paralelos

Modelos computacionales: Actores

Hewitt, Bishop & Steiger: "A Universal Modular Actor Formalism for Artificial Intelligence," IJCAI 1973

- Comunicación asíncrona, p.ej. Erlang & Scala.



Algoritmos paralelos

Modelos computacionales: CSP

Hoare: "Communicating sequential processes," CACM 21 (8), 1978

- Procesos anónimos
  (se utilizan canales para el paso de mensajes).

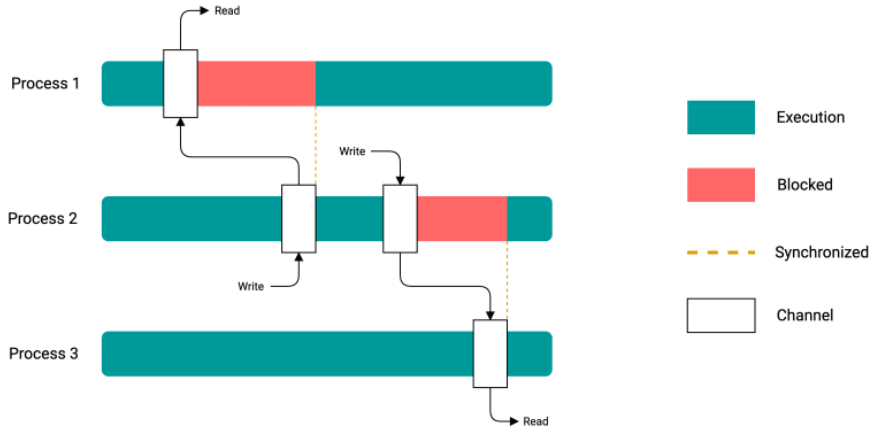
- Comunicación síncrona (los mensajes en CSP, se reciben en el orden en el que se enviaron).

Algoritmos paralelos

Modelos computacionales: CSP

C.A.R. Hoare: "Communicating sequential processes,"
Communications of the ACM 21(8), 1978

- Comunicación síncrona, p.ej. Go.



Algoritmos paralelos

Modelos computacionales: PRAM

Parallel Random-Access Machine

- Modelo teórico para el análisis de algoritmos paralelos.
- Memoria compartida.
- Múltiples procesadores con memoria local (ID).
- Un procesador puede acceder a la memoria compartida en tiempo constante (ignora la complejidad de la comunicación entre procesos).

Algoritmos paralelos

Modelos computacionales: PRAM

El acceso a la misma posición en la memoria compartida puede ocasionar conflictos, que se pueden resolver con diferentes estrategias:

- **EREW** [Exclusive Read Exclusive Write],
  i.e. sólo un procesador lee o escribe de la memoria.
- **CREW** [Concurrent Read Exclusive Write],
  i.e. se puede leer en paralelo, pero no escribir.
- **CRCW** [Concurrent Read Concurrent Write],
  i.e. sin restricciones.
- **ERCW** [Exclusive Read Concurrent Write],
  no suele usarse

Algoritmos paralelos

Modelos computacionales: PRAM

CW [Concurrent Write]

- Prioridad
  (el procesador con mayor prioridad, menor ID, gana).
- Arbitrario/aleatorio
  (se elige un valor aleatoriamente)
- Común
  (sólo tiene éxito si todos escriben el mismo valor).

EREW ≤ CREW ≤ común ≤ aleatorio ≤ prioridad

Algoritmos paralelos

Modelos computacionales: BSP

Leslie G. Valiant: "A bridging model for parallel computation", Communications of the ACM, 33(8), 1990

- BSP [Bulk Synchronous Parallel] es similar al modelo PRAM, pero sin ignorar el coste asociado a la comunicación y a la sincronización entre procesos.

p.ej. MapReduce, Hadoop, Spark…

Algoritmos paralelos

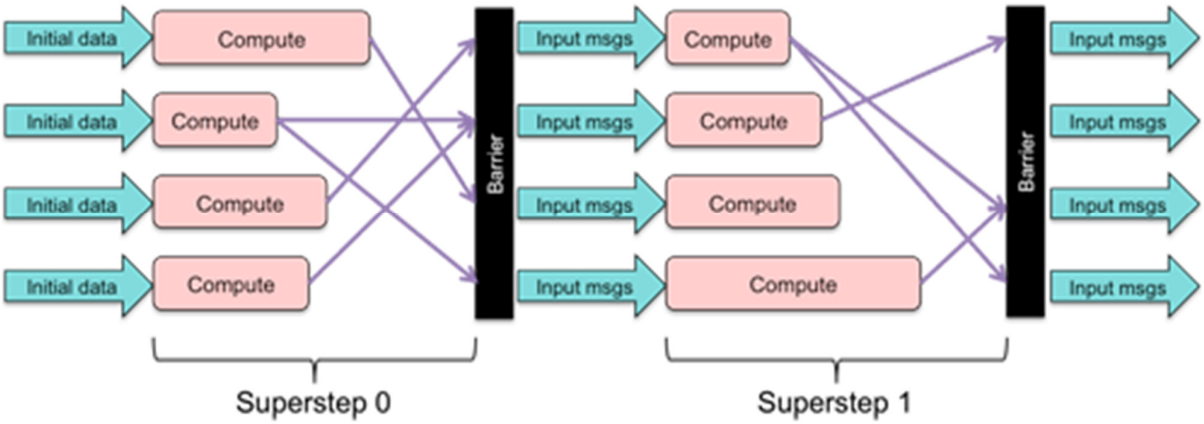
Modelos computacionales: BSP

Los algoritmos BSP proceden en pasos de 3 etapas:

1. Cálculo concurrente
   (cada procesador realiza cálculos locales),
2. Comunicación (intercambio de datos) y
3. Sincronización (cuando un proceso alcanza la barrera tiene que esperar hasta que lleguen los demás).

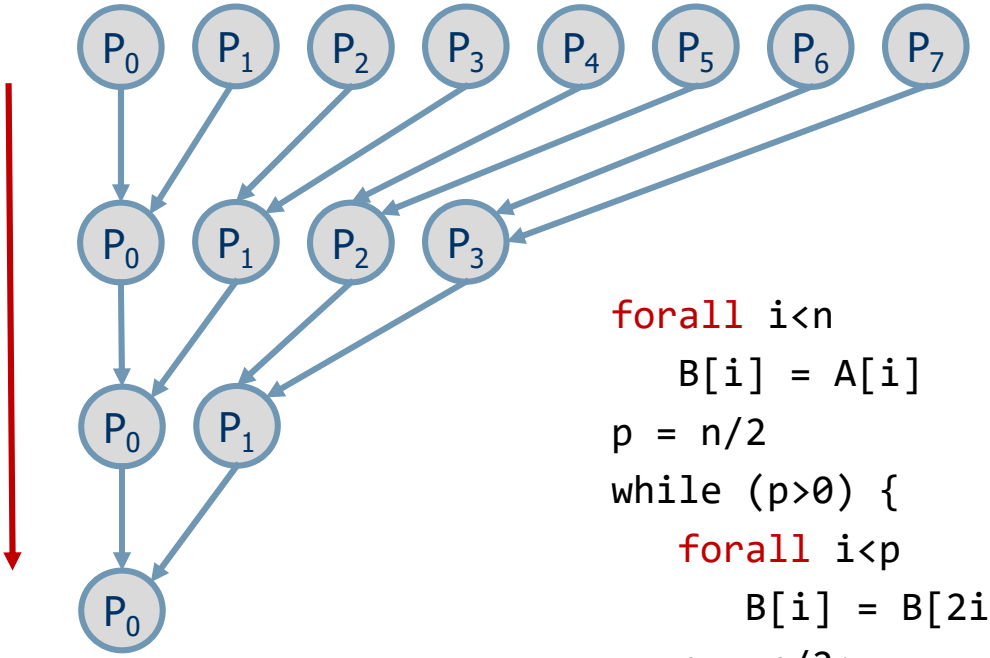
Algoritmos paralelos

Modelos computacionales: BSP



Algoritmos paralelos

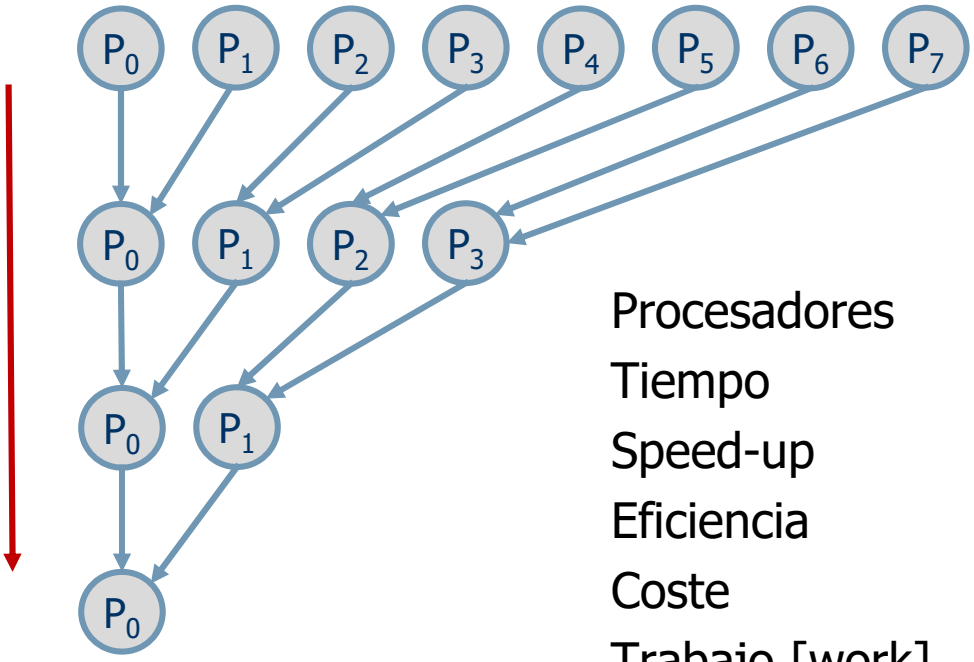
Ejemplo: Suma en paralelo




```
forall i<n
   B[i] = A[i]
p = n/2
while (p>0) {
   forall i<p
      B[i] = B[2i]+B[2i+1]
   p = p/2;
}
```

Algoritmos paralelos

Ejemplo: Suma en paralelo



Procesadores	n
Tiempo	O(log n)
Speed-up	n / log n
Eficiencia	1 / log n
Coste	n log n
Trabajo [work]	n

Métricas de rendimiento

Evaluación del rendimiento de un algoritmo parlalelo:

- **Tiempo de ejecución**: t(n)
- **Trabajo realizado**: W(n)
- **Speed-up / escalabilidad**: S(n)
  (absoluta, con respecto al mejor algoritmo secuencial, o relativa, con respecto a la implementación secuencial del mismo algoritmo en un procesador).
- **Coste**: C(n) = procesadores * t(n)

Métricas de rendimiento

- Un algoritmo paralelo es óptimo con respecto al trabajo realizado si W(n) es del mismo orden de eficiencia que su algoritmo secuencial
  (la paralelización no implica trabajo "adicional").

- Teorema de Brent: **$t(n,p) \in O(W(n)/p + t(n))$**
  Tiempo necesario para ejecutar un algoritmo paralelo en p procesadores.

- W(n) coincide con C(n) si $p \in O(W(n)/t(n))$
  $C(n) = p * t(n,p) \in O(W(n) + p*t(n))$

Métricas de rendimiento

t_p Tiempo de ejecución usando p procesadores

Speed-up

$$S_p = t_1 / t_p$$

Eficiencia

$$E_p = S_p / p$$

Coste

$$C_p = p * t_p$$

Ley de Amdahl

Los algoritmos paralelos escalan
hasta cierto número de procesadores p.

f Fracción del problema que es secuencial

Mejor tiempo paralelo

$$t_{par} = t_{seq}\left(f + \frac{1-f}{p}\right)$$

Speedup

$$S_p = \frac{t_1}{t_p} = \frac{1}{f + \frac{1-f}{p}}$$

Ley de Amdahl

Con p procesadores:

$$S_p = \frac{t_1}{t_p} = \frac{1}{f + \frac{1-f}{p}}$$

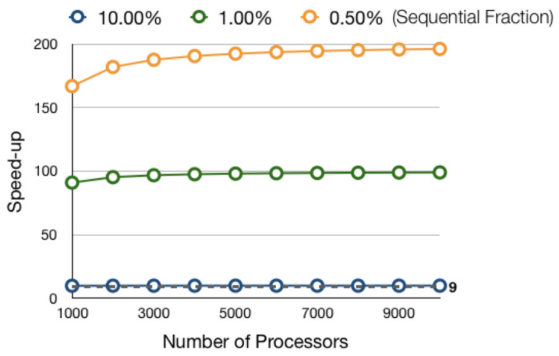
Cota superior, con p $\to \infty$:

$$S_\infty = \frac{1}{f}$$

f	S_∞
20%	5
10%	10
5%	20
2%	50
1%	100

Ley de Amdahl

Las ganancias proporcionadas por el algoritmo paralelo se saturan y la eficiencia disminuye…



… pero el límite suele depender de n, el tamaño del problema :-)

Ley de Gustafson

Conforme el número de procesadores p aumenta, también lo hacen las oportunidades de paralelización.

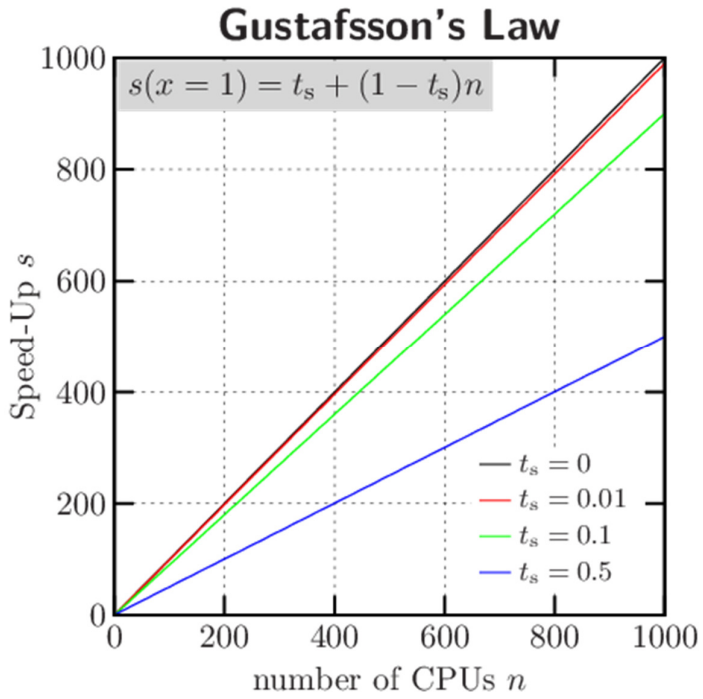
$$t(n,p) = t_{seq} + t_{par}$$

$$t_{seq} = f * t(n,p)$$
$$t_{par} = (1-f) * t(n,p)$$

$$t(n,1) = f * t(n,p) + p * (1-f) * t(n,p)$$

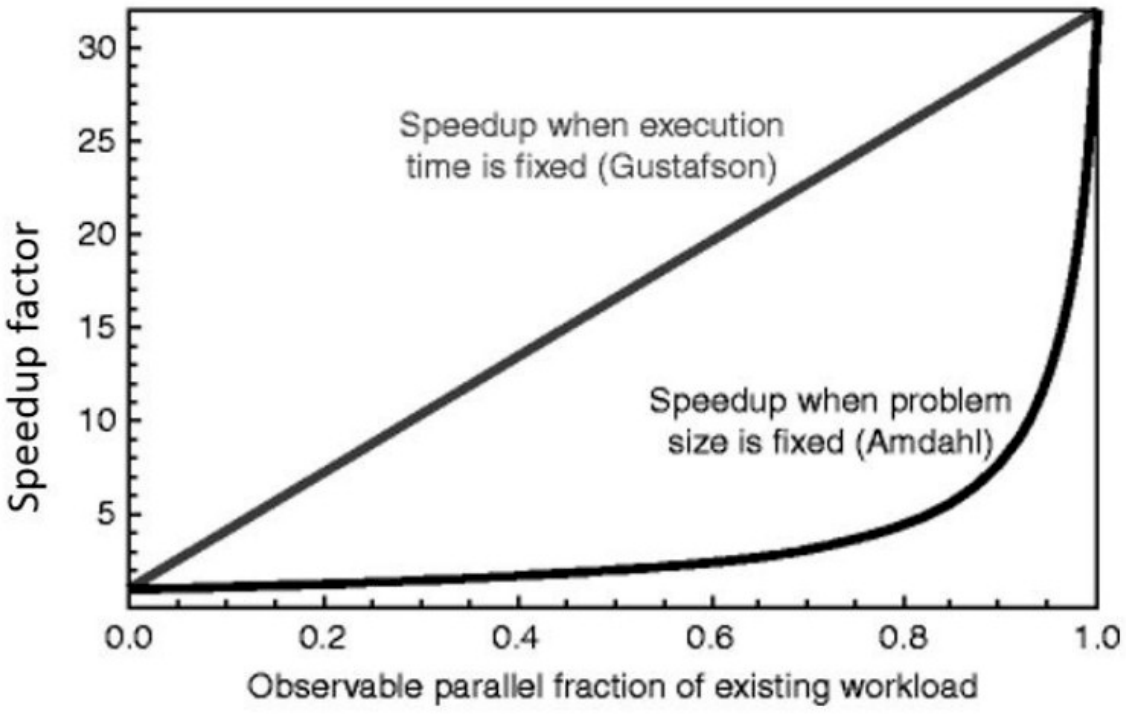
$$S_p = \frac{t(n,1)}{t(n,p)} = f + \boldsymbol{p} * (1-f)$$

Ley de Gustafson



Speedup

Ley de Amdahl vs. Ley de Gustafson



Isoeficiencia

Métrica de escalabilidad:

Tasa al que el tamaño del problema debe crecer para mantener una eficiencia constante.

- Si n no debe crecer con p,
  el algoritmo es fuertemente escalable.
- Cuanto menor sea la tasa de crecimiento de n,
  más escalable es el algoritmo.

Isoeficiencia

Eficiencia

$$E(n,p) = \frac{S(n,p)}{p} = \frac{1}{p}\frac{t(n,1)}{t(n,p)} = \frac{t(n,1)}{p\,t(n,p)}$$

Isoeficiencia

$$I(p) = n(p) \quad \text{tal que} \quad E(n,p) = k \in \Theta(1)$$

"Overhead" debido a la paralelización:

$$o(p) = p * t(n,p) - t(n,1)$$

$$\boldsymbol{I(p) \in \Omega(o(p))}$$

Isoeficiencia

EJEMPLO

$$t(n,p) \in \Theta\left(\frac{n}{p} + \log p\right)$$

$$t(n,1) \in \Theta(n)$$
$$p\, t(n,p) \in \Theta(n + p \log p)$$

$$E(n,p) = \frac{t(n,1)}{p\, t(n,p)} \in \Theta\left(\frac{n}{n + p \log p}\right) = \Theta(1)$$

$$\boldsymbol{I(p) = \Omega(p \log p)}$$