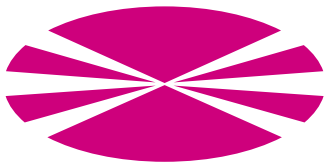
# T3: Principios de Programación Paralela

## T3.3: Análisis de Algoritmos Paralelos

Departamento de Ingeniería de Computadores

Primavera 2021



# Índice

1. Conceptos Básicos
2. Medidas de Prestaciones de Algoritmos Paralelos

# Conceptos Básicos

## Objetivos de la programación paralela

- Reducir el tiempo de ejecución.
- Acceso a recursos computacionales (e.g., memoria) no presentes en un único procesador.

## Necesidad de análisis de algoritmos paralelos

Selección del algoritmo más adecuado, en términos de tiempo de ejecución, consumo de memoria y eficiencia, de entre un conjunto de algoritmos.

# Conceptos Básicos

## Factores que influyen en el rendimiento de un programa secuencial

- Lenguaje de implementación y experiencia/productividad del programador.
- Compilador y opciones de compilación empleadas.
- Tipos de datos utilizados.
- Número de operaciones en punto flotante (flops).
- Complejidad del algoritmo implementado (e.g., $O(n^3)$, $O(nlog(n))$).
- Recursos hardware en los que se ejecuta un programa.
- Acceso compartido/exclusivo a dichos recursos.

# Conceptos Básicos

## Tiempo de Ejecución Paralelo

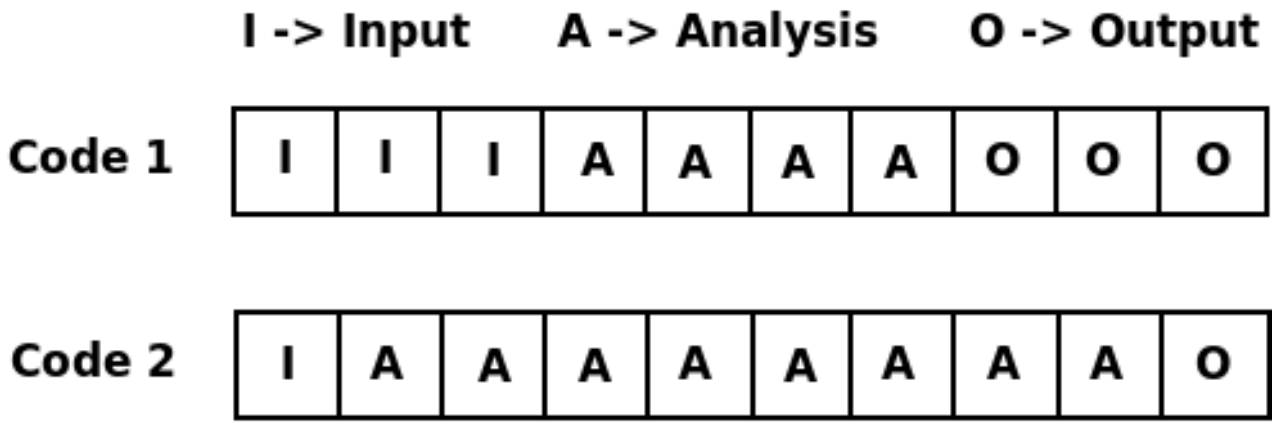
Es el tiempo que transcurre desde que empieza la ejecución el primer procesador hasta que termina el último procesador.

## Factores que influyen en el rendimiento de un programa paralelo

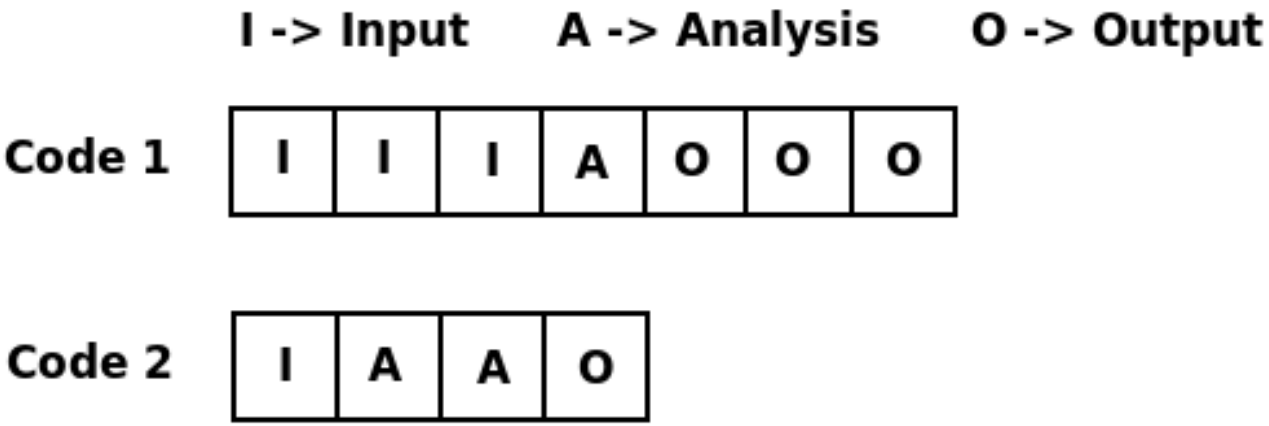
Además de los factores indicados para el caso secuencial,

- Compilación más óptima en códigos secuenciales que en paralelos.
- Las bibliotecas suelen estar más optimizadas en su versión secuencial que en la paralela, si es que llega a estar paralelizada.
- Mayor influencia de otros elementos (otros procesos, usuarios, tráfico en redes compartidas, carga/rendimiento desigual).
- La gestión de procesos/threads añade sobrecarga/*overhead* debido a su creación, sincronización, y reserva/liberación de recursos.
- Existencia de partes del problema no paralelizables y necesidad de código adicional en aquellas paralelizables.

# Conceptos Básicos



# Conceptos Básicos



# Conceptos Básicos

## Ley de Amdahl (I)

Con la programación paralela pretendemos que $p$ procesadores sean capaces de resolver un problema $p$ veces más rápido que en secuencial.

Existen una serie de factores, ya mencionados, que dificultan la consecución de este objetivo.

La ley de Amdahl mide el límite en la ganancia de velocidad obtenible en la resolución de un problema con una fracción intrínsecamente secuencial ($F_s$) y otra perfectamente paralelizable ($F_p$).

$$T_{secuencial} = F_s + F_p$$

$$T_{paralelo}(p) = F_s + F_p/p$$

$$A_p = \frac{T_{secuencial}}{T_{paralelo}(p)} = \frac{F_s + F_p}{F_s + \frac{F_p}{p}}$$

Ley de Amdahl también nos da la aceleración máxima teórica:

$$A_{max} = \lim_{p \to \infty} \frac{T_{secuencial}}{T_{paralelo}(p)} = \lim_{p \to \infty} \frac{F_s + F_p}{F_s + \frac{F_p}{p}} = \frac{F_s + F_p}{F_s}$$

# Conceptos Básicos

## Ley de Amdahl (II)

Cuando hablamos de *t*iempo, en realidad podemos utilizar cualquier otra medida representativa, como por ejemplo:

- Número de ciclos de reloj.
- Número de operaciones (considerando que tienen la misma duración).
- Fracción del tiempo total de ejecución, en cuyo caso consideramos que $T_{secuencial}$ es igual a 1.

En el fondo, podríamos convertir estas medidas a unidades de tiempo aplicando los mismos factores en numerador y denominador, por lo que se eliminarían de la ecuación.

# Medidas de Prestaciones de Algoritmos Paralelos

## Necesidad de medidas

- **Eficacia** en programación paralela: reducción del tiempo de ejecución.
- **Eficiencia** en programación paralela: reducción del tiempo de ejecución con los menores recursos.
- **Necesidad** de nuevas métricas de prestaciones de algoritmos paralelos:
  - *Speedup* o aceleración
  - *Eficiencia*
  - *Escalabilidad fuerte*
  - *Escalabilidad débil*

# Medidas de Prestaciones de Algoritmos Paralelos

## Speedup

$$Speedup(p) = \frac{T_{secuencial}}{T_{paralelo}(p)}$$

El *Speedup* toma valores entre 0 e idealmente $p$, aunque excepcionalmente puede ser superior a $p$ (*speedup superlineal*), cuando el algoritmo paralelo mejora la gestión de la memoria (menos fallos caché) del algoritmo secuencial.

Elección de $T_{secuencial}$:

- Mejor algoritmo secuencial, aunque suele ser difícil de determinar (p.ej., el mejor algoritmo de ordenación depende de la entrada). Se elige el más extendido/estándar.
- $T_{paralelo}(1)$, tiempo de ejecución en un procesador del algoritmo paralelo, aunque el algoritmo paralelo puede no ser el óptimo para la ejecución secuencial.

# Medidas de Prestaciones de Algoritmos Paralelos

## Eficiencia

$$Eficiencia(p) = \frac{Speedup(p)}{p} = \frac{T_{secuencial}}{p \times T_{paralelo}(p)}$$

La *Eficiencia* toma valores entre 0 e idealmente 1, aunque excepcionalmente puede ser superior a 1 en el caso de *Speedup superlineal*.

## Coste/Overhead

El *Coste* de un algoritmo paralelo es el tiempo dedicado por todo el sistema a la resolución de un problema: $Coste = p \times T_{paralelo}(p)$. Idealmente coincide con $T_{secuencial}$ pero generalmente es mayor.

Sobrecarga u $Overhead = Coste - T_{secuencial}$

# Medidas de Prestaciones de Algoritmos Paralelos

## Escalabilidad fuerte/débil

Los sistemas paralelos se hallan en constante evolución para proporcionar mayores prestaciones o abordar problemas de mayor tamaño, resolviendo cuestiones o tamaños de problema inabordables previamente. Suele ser más factible duplicar el tamaño del problema a resolver que reducir el tiempo de ejecución a la mitad.

**Escalabilidad** *es la capacidad de mantener la eficiencia en la resolución de un problema al disponer de mayores recursos.*

Dos conceptos de escalabilidad:

- **Escalado fuerte**: variación de la eficiencia al emplear más recursos en presencia de un tamaño global de problema fijo.
- **Escalado débil**: variación de la eficiencia al emplear más recursos en presencia de un tamaño de problema fijo por cada elemento de procesado (tamaño global de problema variable).

# Medidas de Prestaciones de Algoritmos Paralelos

## Escalabilidad fuerte (ejemplo)

Tiempo secuencial: 3600 segundos

| Num. Procs. | Tiempo (s) | Speedup | Eficiencia |
|---|---|---|---|
| 2 | 1802 | 1.99 | 0.99 |
| 4 | 904 | 3.98 | 0.99 |
| .. | .. | .. | .. |
| 64 | 120 | 30 | 0.47 |
| 128 | 156 | 23 | 0.18 |

# Medidas de Prestaciones de Algoritmos Paralelos

## Escalabilidad débil (ejemplo)

| Num. Procs. | Tiempo sec. (s) | Tiempo par (s) | Speedup | Eficiencia |
|---|---|---|---|---|
| 2 | 7200 | 3602 | 1.99 | 0.99 |
| 4 | 14400 | 3604 | 3.99 | 0.99 |
| .. | .. | .. | .. | |
| 64 | 230400 | 3664 | 62.88 | 0.98 |
| 128 | 460800 | 3728 | 123.60 | 0.97 |

# Medidas de Prestaciones de Algoritmos Paralelos

## Análisis de prestaciones de un algoritmo paralelo (ejemplo)

Tiempo de ejecución (s):

| | Procesadores | | | |
|---|---|---|---|---|
| Tamaño | 1 | 2 | 4 | 8 |
| 1000 | 100 | 60 | 33 | 25 |
| 2000 | 210 | 110 | 60 | 40 |
| 4000 | 450 | 240 | 130 | 70 |
| 8000 | 1000 | 500 | 260 | 120 |

Speedup:

| | Procesadores | | | |
|---|---|---|---|---|
| Tamaño | 1 | 2 | 4 | 8 |
| 1000 | 1 | 1,67 | 3,03 | 4 |
| 2000 | 1 | 1,91 | 3,50 | 5,25 |
| 4000 | 1 | 1,88 | 3,46 | 6,43 |
| 8000 | 1 | 2 | 3,85 | 8,33 |

Eficiencia (%):

| | Procesadores | | | |
|---|---|---|---|---|
| Tamaño | 1 | 2 | 4 | 8 |
| 1000 | 100 | 83 | 76 | 50 |
| 2000 | 100 | 95 | 88 | 66 |
| 4000 | 100 | 94 | 87 | 80 |
| 8000 | 100 | 100 | 96 | 104 |

# Medidas de Prestaciones de Algoritmos Paralelos

Speedup (izq.) y eficiencia paralela (derecha):