Teoría 5 - análisis de rendimientos

jueves, 10 de abril de 2025 19:12

Tiempo de ejecución:
Algoritmo secuencial se evalua por esta medida. Hay una regla que permite asociar tiempo en base al tamaño de la enrada
En algoritmos paralelos, tiempo de ejecución no depende del tamaño de entrada sino de cant de procesadores y formas de comunicación.
El análisis se hace teniendo en cuenta el algoritmo, hardware y software.

Fuetes de overhead:
No siempre duplicar los procesadores duplica la velocidad. Hay ocio, interacción entre procesos, cómputo extra.
Speedup: beneficio de usar procesamieno paralelo comparado a secuencial.

$$S_p(n) = \frac{T_s(n)}{T_p(n)}$$

Cuánto más rápido es el algoritmo paralelo respecto al secuencial.
Se usa el mejor algoritmo secuencial(supongo que paralelo tb)
Si es menor a 1 entonces es mejor el secuencial
El mejor resultado es si podemos distribuir bien el trabajo sin generar ocio, interacción ni cómputo extra.

- Con p unidades de procesamiento → $S_p(n) = p$ (conocido como *Speedup lineal*, *Speedup óptimo*, *Speedup perfecto)*
- Teóricamente, siempre se cumple que $S_p(n) \leq p$
  - Un Speedup mayor a p sólo es posible si cada unidad de procesamiento requiere menos de $\frac{T_s(n)}{p}$ unidades de tiempo

Speedup superlineal(no entendi)

- La **Eficiencia** es una medida de la fracción de tiempo en la cual las unidades de procesamiento son empleadas en forma útil

$$E_p(n) = \frac{S_p(n)}{S_{Opt}}$$

- En arquitecturas homogéneas $S_{Opt} = p$ mientras que en heterogéneas $S_{Opt} = pct$
- Si $S_p(n) = p$ (sistema paralelo ideal), entonces $E_p(n) = 1$
- En la práctica, $S_p(n) \leq p$ lo que implica que $E_p(n) \leq 1$
- Por definición, $E_p(n) > 0$. Por lo tanto, $0 < E_p(n) \leq 1$

Overhead total: diferencia entre la suma del tiempo requerido por todas las undades de procesamiento y el del mejr algoritmo secuencial para el mismo problema con una sola unidad de procesamiento

Overhead de las comunicaciones: relación entre el tiempo requerido por las comunicaciones de la solución y el tiempo total que requiera

$$OC_p(n) = \frac{Tcomm_p(n)}{T_p(n)} \times 100$$

Ley de Amdahl

Hay una restriccion heavy que tiene que ver con las secciones de código que no se pueden paralelizar--bloque de ejecución secuencial
La ley permite estimar el speedup alcanzable en los programas paralelos pero con partes secuenciales

- Dada una fracción f, $0 \leq f \leq 1$, de un programa paralelo que debe ser ejecutada secuencialmente, el tiempo de ejecución paralela se calcula como:

$$T_p(n) = f \times T_s(n) + \frac{(1-f) \times T_s(n)}{p}$$

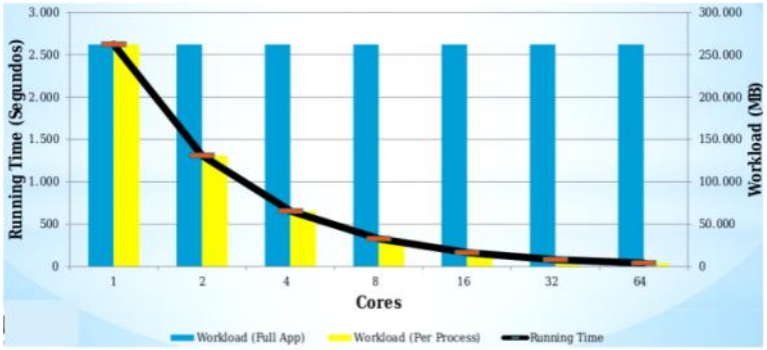
- Entonces el Speedup ahora puede re-escribirse de la siguiente forma:

$$S^A{}_P\,(n) = \frac{T_s(n)}{f \times T_s(n) + \frac{(1-f) \times T_s(n)}{p}} = \frac{1}{f + \frac{(1-f)}{p}}$$

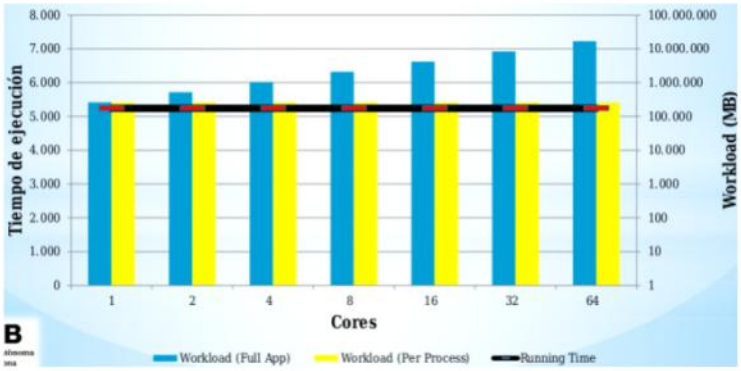
El speedup va a estar limitado a 1/f siempre

Escalabilidad: capacidad de un sistema de mantener un nivel de eficiencia fijo al incrementar el nro de proceesamiento y el taaño del problema a resolver.
Capacidad de incrementar el Speedup en forma proporcional al número de unidades de procesamiento empleadas

- Casos especiales:
  - *Escalabilidad fuerte*: Cuando al incrementar el número de unidades de procesamiento, no resulta necesario aumentar el tamaño de problema para mantener la eficiencia en un valor fijo.



  - *Escalabilidad débil*: Cuando al incrementar el número de unidades de procesamiento, resulta necesario también aumentar el tamaño de problema para mantener la eficiencia en un valor fijo.



La ley de amdahl no se da cuenta si el speedup incrementa por un tamaño mayor de problema
Ley de Gustafson:

🔁 Primero, la Ley de Amdahl:

Dice que **el speedup (aceleración)** que se puede obtener al paralelizar un programa **está limitado por la parte secuencial** (la que no se puede paralelizar).

Ejemplo simple:

- Supongamos que un 10% de tu programa no se puede paralelizar (es secuencial).
- Aunque tengas 1000 procesadores trabajando, ese 10% siempre va a tardar lo mismo.
- Entonces, el speedup total está limitado por ese pedacito secuencial, aunque agregues más procesadores.

😕 ¿Cuál es el problema con Amdahl?

Amdahl **asume que el tamaño del problema es fijo**. O sea, siempre estás ejecutando el mismo trabajo, sin importar cuántos procesadores tengas.

💡 Entra Gustafson (años 80):

Gustafson observó algo diferente y más realista:

> **"Cuando tenemos más procesadores, en vez de ejecutar el mismo problema más rápido, ejecutamos un problema más grande en el mismo tiempo."**

Eso se llama **escalabilidad**.

🧠 ¿Qué implica esto?

1. Si tenés más procesadores, probablemente quieras **resolver un problema más grande** (porque podés hacerlo en el mismo tiempo).
2. Al agrandar el problema, la parte secuencial del programa **no necesariamente crece igual de rápido** que el resto.
3. Entonces, **la fracción secuencial no domina tanto** como en la Ley de Amdahl.
4. Por eso, **el speedup real puede ser mucho mayor** de lo que Amdahl predice.

Ninguna de las dos visiones está mal, solo **modelan situaciones distintas**:

- **Amdahl**: Mismo problema, queremos hacerlo más rápido.
- **Gustafson**: Mismo tiempo, queremos resolver más problema.

- Dada una fracción f', $0 \leq f' \leq 1$, de un programa paralelo que debe ser ejecutada secuencialmente pero que no crece en forma proporcional al tamaño de problema, el Speedup escalado se calcula como:

$$S^{S}{}_{p}(n) = \frac{T_s(n)}{T_p(n)} = \frac{f' \times T_p(n) + (1 - f') \times T_p(n) \times p}{T_p(n)} = p + (1 - p) \times f'$$

- Esta versión requiere 2 suposiciones: (1) $T_p(n)$ se mantiene constante y (2) $f' \times T_p(n)$ no escala en forma proporcional al aumento de n y p

Speedup escalado

Escalabilidad fuerte (Amdhal)	Escalabilidad débil (Gustafson)
• El tamaño del problema se mantiene fijo a medida que se incrementa la cantidad de procesadores.	• El tamaño del problema *por procesador* se mantiene fijo a medida que se incrementa la cantidad de procesadores → El tamaño total del problema es proporcional al número de procesadores usados.
• El objetivo es resolver el mismo problema de forma más rápida	• El objetivo es resolver un problema más grande en la misma cantidad de tiempo
• El *escalado perfecto* se logra cuando el problema se resuelve en 1/P unidades de tiempo (comparado al secuencial)	• El *escalado perfecto* se logra cuando se resuelve un problema P veces más grande en la misma cantidad de tiempo que el secuencial.

En arquitecturas heterogéneas, se mide desbalance de carga

- ***Desbalance de carga***:

$$D = \frac{max_{i=0..p-1}\left(T_{pi}(n)\right) - min_{i=0..p-1}(T_{pi}(n))}{prom_{i=0..p-1}(T_{pi}(n))}$$

- Si todas las unidades de procesamiento toman el mismo tiempo, entonces $D = 0 \rightarrow$ Poco usual
- En general, se debe intentar que D esté lo más cerca posible de 0

CMuchas veces podemos identificar a mano que parte tiene mucha demanda computacional. Suelen ser bucles.
A veces en programas grandes dse hace profiling (perfilado): se mide el rendimiento con un programa para identificar los puntos críticos
Pueden usarse tareas para medir la parte determinada del programa que nos interese