ARQUITECTURA E INGENIERÍA DE COMPUTADORES

Tema 1.2

Miquel Gómez Corral

Ing. Informática ETSINF, UPV

TEMA 1.2

Análisis de prestaciones

DEFINICIÓN DE PRESTACIONES

Prestaciones: Tiene diferentes puntos de vista, con solo 1 usuario o con varios (*administrador de sistemas*)

- **Usuario**: Están relacionadas con tener el mínimo tiempo de ejecución para un programa dado. Lo que se le puede llamar *TIEMPO DE RESPUESTA O EJECUCIÓN*.
- **Administrador**: Cierta cantidad de usuarios ejecutando programas, se usa la productividad, que sería la cantidad de trabajos que se hacen en el tiempo.

Varios trabajos: $Prouctividad = \frac{n}{Tiempo\ ejecución}$

Un trabajo (Usuario): $Productivadad = \frac{1}{Tiempo\ ejecución}$

Comparaciones

Cuando se **comparan 2 computadores** "X" e "Y", se escoge el más lento como referencia y se obtiene la *ACELERACIÓN* "S" dividiendo el tiempo de uno entre el del otro ($\frac{Tx}{Ty}$ o al revés).

Si **hay varios** se escoge uno diferente que no está en el grupo para que actúe de referencia y se dividen todos los tiempos entre el de ese computador. *Se supone que este computador ajeno es más lento y siempre será por el que dividas. Si es más rápido dan menor que 1 y ya.*

El grande siempre entre el más pequeño, el lento entre el rápido.

Aceleración S: Se puede calcular como $S = \frac{Ty}{Tx} = \frac{Px}{Py} = 1 + \frac{n}{100}$.

- **N**: Es la cantidad de mejora en % $n = (El\ valor\ de\ la\ aceleración\ (resultado) - 1) * 100$

Se dice que "X es S veces mas rápido que Y" y/o que "X es n % más rápido que Y"

PRINCIPIOS CUANTITATIVOS DEL DISEÑO DE COMPUTADORES

Prestaciones del Procesador

$$\boldsymbol{Tejecución} = \frac{Segundos}{Programa} = \frac{num.Instr.}{Programa} \cdot \frac{Ciclos}{num.Instr.} \cdot \frac{Seg}{Ciclos} = \boldsymbol{I \cdot CPI \cdot T}$$

- I = Son el **número de Instrucciones**. Depende del **ISA** y del **compilador**.
- CPI = **Cantidad de ciclos** por Instrucción. Depende del **ISA** y de la **organización** (el Hardware, la estructura).
- T = **Cuanto dura cada ciclo**, es la inversa de la frecuencia. Depende de la **tecnología** y la **organización**.

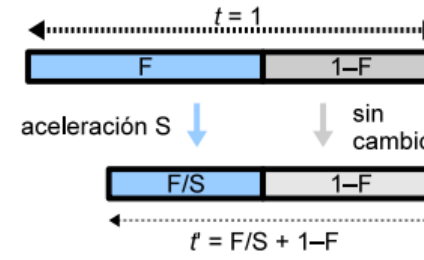
Dependientes unos de otros, no es posible reducir uno sin afectar otro, HAY QUE TENER EN CUENTA A LOS 3.

Reducir el Tiempo de ejecución: Habría que reducir alguno de los 3 factores, sin embargo, dependen de casi las mismas cosas, por lo que es muy MUY difícil. *Ejemplo: se cambia el ISA para mejorar el "I" pero se puede empeorar el CPI que también depende la complejidad de la ISA.*

Ley de Amdahl

Describe como afecta el cambio de una parte de un proceso en el total. *No porque una parte sea el doble de rápida el resultado final también lo será.*

Nuevo tiempo: **F** es la **fracción de tiempo que cambia** y **S** la **aceleración Local** que se le aplica. Por lo que ahora el tiempo pasa de ser 1 a $t' = \frac{F}{S} + 1 - F$



Para t /= 1 (general): $t' = t \cdot \frac{F}{S} + t \cdot (1 - F) = t \cdot (\frac{F}{S} + 1 - F)$

Aceleración Global: Para sacar **S'** la **Aceleración Global**: $S' = \frac{t}{t'} = \frac{1}{\frac{F}{S}+1-F}$

Por otra parte, dado el **porcentaje de en el que se gasta el componente que quieres mejorar**, se puede *sacar el límite para saber hasta CUANTO podrías mejorar el computador SOLO mejorando eso*.

$$S'_{\infty} = lim_{S\to\infty} \qquad S' = \frac{1}{1-F}$$

Mejorando varios componentes

La ley de Amdahl se **puede generalizar para múltiples fracciones/Componentes**. Los que mejores serán divididos por su factor de aceleración mientras que los que no se mejoren permanecerán igual (*Lo mismo que dividir entre S = 1).*

$$S' = \frac{1}{\frac{F1}{S1} + \frac{F2}{S2} + \cdots + \frac{Fn}{Sn}}$$

Donde $F1 + F2 + \ldots + Fn = 1$, los valores originales. Algún "Si" podría ser 1 (*sin aceleración, se queda igual*).

Aceleración local compuesta: Por otro lado, una aceleración local "Si" puede ser el resultado de la composición de varias aceleraciones locales independientes: $Si = Si1 \times Si2 \times \ldots \times Sim$

Ejemplo

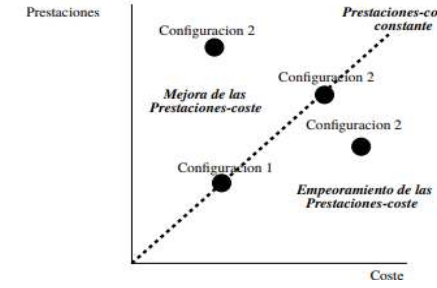
El Tejecución de un programa P tiene 2 partes: $Te = F1 + F2 = 1$. Siendo **F2 paralelizable** (*la cantidad de núcleos de afecta*). P se ejecuta en un procesador **con 2 núcleo a 2,2GHz**. Ahora lo hacen en un procesador de **4 Núcleos a 3,3GHz.**

$$S1 = \frac{3{,}3\ GHz}{2{,}2\ GHz} = 1{,}5 \qquad S2 = \frac{3{,}3\ GHz}{2{,}2\ GHz} \cdot \frac{4\ núcleos}{2\ núcleos} = 3$$

Solución: $S' = \frac{1}{\frac{F1}{S1}+\frac{F2}{S2}} = \frac{1}{\frac{F1}{1.5}+\frac{F2}{3}}$

Relación prestaciones coste

Mide la relación entre las prestaciones alcanzadas y el coste (precio, watios, …) de una configuración dada. Permite comparar varias alternativas



Optimo: Coste 0 y tiene x prestaciones justo en el eje Y.

Medio: Si me vale X y tiene Y prestaciones y mejoro el doble el coste (2x), las prestaciones serán 2Y. En la línea del medio.

Empeoro: Por debajo de la línea del medio, subo algo el coste, pero no consigo la misma cantidad de prestaciones (en proporción).

Mejoro: Por encima de la línea del medio, si subo algo el coste, me sube mucho muchas las prestaciones

LA MEDIDA DE PRESTACIONES

Para medir las prestaciones hace falta escoger programas de prueba para poder tener una referencia, programas de los que **se disponga su código fuente** para poder compilarlo para cada computador. Los mejores **programas** para probar son los **reales**, en concreto los que vayas a usar con dicho ordenador. En caso de que **no los elijas** (*porque tengas muchos, porque no lo puedas compilar…*) usar unos que representen a la gran mayoría de usuarios.

Desacreditados*:* Hay **programas que** NO se deben usar para eso: *Kernels, Toy Benchmarks, Synthetic Benchmarks.*

Benchmarks Suites (*Conjunto de aplicaciones*)

Son **programas reales sin interactividad y kernels** especializados para **medir las prestaciones dentro de un perfil de uso**. Testean la **E/S**, la **CPU**, los **gráficos**, **servidores**… **Estos se actualizan constantemente** para que Los programas incluidos en el paquete representen el trabajo que hace un usuario típico en el momento.

Reproducibilidad: Las **medidas han de ser reproducibles** → hay que indicar/especificar todos los detalles:

- **Hardware**: Procesador, cache, memoria, disco…
- **Software**: Sistema operativo, programas y versiones, datos de entrada, opciones de ejecución…

Si tu luego coges todos esos datos, el mismo Benchamark y lo recreas, tendrías que conseguir el mismo resultado.

Cuando haces las pruebas te dan un montón de numero cada test, esos resultados se juntan y terminan donde un valor que te dice: Tu ordenador es x veces más rápido que el ordenador de referencia. *El ordenador de referencia te da igual cual es, si el que tenía da 50 en el test y el nuevo 100 sé que el nuevo será el doble de rápido.*

Comparación de computadores

Para obtener una medida resumen de la ejecución de varios programas podemos apreciar que, una Característica de una buena medición de tiempos es: *El valor medio debe ser directamente proporcional al tiempo total de ejecución.*

Tiempo total: $T_t = \sum_{i=1}^{n} Tiempo_i$ **Media aritmética**: $T_a = \frac{1}{n} \cdot \sum_{i=1}^{n} Tiempo_i$

Tiempo de ejecución ponderado: $T_w = \sum_{i=1}^{n} w_i \cdot Tiempo_i$

Donde wi representa la frecuencia del programa i en la carga de trabajo.

Media geométrica (SPEC): $R = \sqrt[n]{\prod_{i=1}^{n} \frac{TiempoRefi}{Tiempoi}}$

Media geométrica de los tiempos de ejecución normalizados a una máquina de referencia: **R veces más rápido que la referencia.**

Esto es, si tienes 2 máquinas, sacas los tiempos para cada uno en cada test. De cada test 1, divides el tiempo de la referencia entre la otra máquina (los ratios, aceleraciones), del segundo igual… así para todos los test, n veces. Después, todos esos valores los multiplicas y les hacen la Raiz enésima (*la media geométrica*).

Otras medidas: *MIPS y MFLOPS*

MIPS: Millones de **instrucciones** por segundo. NO USAR: En la ecuación se hace **instrucciones**/tiempo. Pero como tiempo de ejecución es "I·CPI·T" puedes tachar las instrucciones, entonces deja de tener sentido xq NO usas las instrucciones. Además, depende del programa que se use y del juego de instrucciones de cada máquina (*si es distinto no vale*) porque el número de instrucciones que ejecutará cada programa es diferente, so malamente.

MFLOPS: Millones de **operaciones** por segundo. *En concreto mido las **operaciones** en coma flotante en el código fuente. Pueden variar entre ordenadores*. Solo es útil si el programa es el mismo y hace falta que usen coma flotante.