UT 1. Introducción a la Arquitectura de Computadores
Tema 1.2 Análisis de prestaciones

J. Flich, P. López, V. Lorente,
A. Pérez, S. Petit, J.C. Ruiz, S. Sáez, J. Sahuquillo

Departamento de Informática de Sistemas y Computadores
Universitat Politècnica de València

UNIVERSIDAD POLITECNICA DE VALENCIA

DISCA

Índice

Bibliografía

John L. Hennessy and David A. Patterson.
Computer Architecture, Fifth Edition: A Quantitative Approach.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5 edition, 2012.

Índice

Tiempo y productividad

Dos estilos o puntos de vista

Visión del usuario: *El usuario quiere acabar pronto*

↓ Tiempo de respuesta (*response time*) o tiempo de ejecución (*execution time*): Tiempo para completar un trabajo.

Visión del administrador del sistema: *El administrador del sistema quiere muchos trabajos por unidad de tiempo.*

↑ Productividad (*throughput*): Operaciones o ejecuciones completadas por unidad de tiempo.

Relación entre los dos estilos de medida:

$$\text{Productividad} = \frac{1}{\text{Tiempo de ejecución}}$$

Comparaciones

Comparando alternativas:

- Cuando hay que comparar dos computadores X e Y, se escoge el más lento Y para que actúe como referencia.
- Cuando se examina un conjunto de diseños $X_1, \ldots, X_n$, se escoge un computador Y (incluído o no en el conjunto) para que actúe de referencia común.

En una comparación elemental entre dos computadores X e Y, hay que escoger una carga (un trabajo de prueba) y medir las prestaciones en ambos. Según el estilo de la medida, obtendremos:

- Tiempo de ejecución: T_X y T_Y
- Productividades: P_X y P_Y

Comparaciones (cont.)

La relación S entre ambos se calcula:

$$S = \frac{T_Y}{T_X} = \frac{P_X}{P_Y} = 1 + \frac{n}{100}$$

Fraseología:

- "X es S veces más rápido que Y"
- "X es n% más rápido que Y"

Índice

Prestaciones del procesador

$$T_{\text{ejecución}} = \frac{\text{seg}}{\text{programa}} = \frac{\text{núm. instr.}}{\text{programa}} \times \frac{\text{ciclos}}{\text{num. instr.}} \times \frac{\text{seg}}{\text{ciclo}} = I * CPI * T$$

- Los tres parámetros están relacionados:
  - I = f(arquitectura del juego de instrucciones, compilador)
  - CPI = f(arquitectura del juego de instrucciones, organización)
  - T = f(tecnología, organización)

$\Rightarrow$ No es posible reducir cada uno de ellos sin afectar los demás.

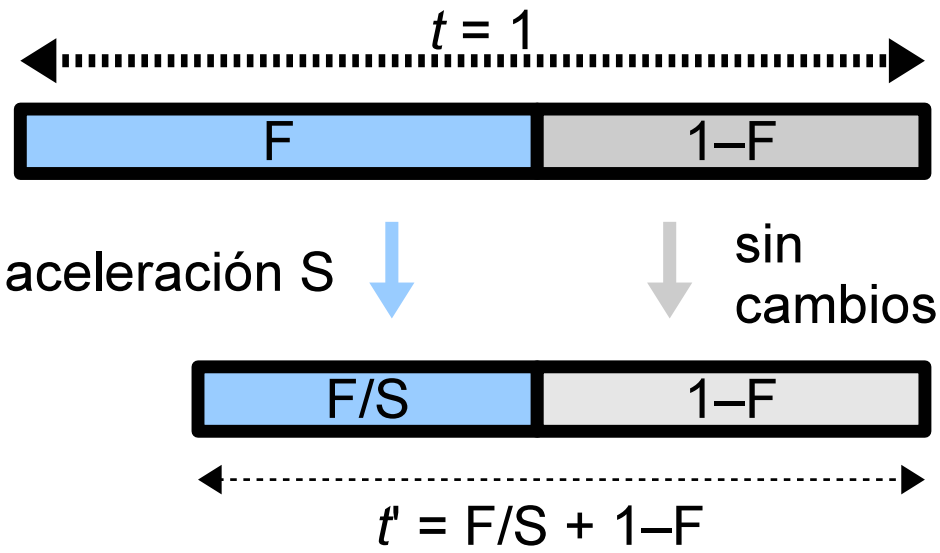
Prestaciones del procesador (cont.)

Ejemplos:

- Un juego de instrucciones más rico puede reducir I, pero puede complicar la organización e implementación, aumentando los CPI y/o T.
- Una organización más compleja puede reducir los CPI, pero puede aumentar el número de puertas lógicas atravesadas por ciclo, aumentado el T.

La ley de Amdahl

Describe cómo afecta el cambio de una parte de un proceso en el total



- F es la fracción de tiempo que cambia
- S es la aceleración que se aplica a esa fracción

En general, para valores $t \neq 1$, el tiempo de proceso resultante es:

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

La ley de Amdahl (cont.)

La aceleración global S' queda:

$$S' = \frac{t}{t'} = \frac{1}{(1-F) + \frac{F}{S}}$$

La fracción de tiempo que no se mejora $(1 - F)$ impone una cota superior a la aceleración que se puede alcanzar:

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

La ley de Amdahl (cont.)

La ley de Amdahl se puede generalizar para múltiples fracciones (n), a cada una de las cuales se aplica una aceleración local distinta. Así,

$$S' = \frac{1}{\frac{F_1}{S_1} + \frac{F_2}{S_2} + ... + \frac{F_n}{S_n}},$$

donde $F_1 + F_2 + ... + F_n = 1$.

Por otro lado, una aceleración local S_i puede ser el resultado de la composición de varias (m_i) aceleraciones locales independientes. Así,

$$S_i = S_{i,1} \times S_{i,2} \times ... \times S_{i,m_i}.$$

La ley de Amdahl (cont.)

Ejemplo: El tiempo de ejecución de un programa P se compone de dos fracciones F_1 y F_2, siendo F_2 paralelizable. Originalmente P se ejecuta en un procesador con 2 núcleos que trabajan a 2,2 GHz. Para acelerar la ejecución se propone actualizar el procesador a 4 núcleos funcionando a 3,3 GHz. ¿Cúal sería la fórmula para obtener la aceleración global de la actualización?

Solución:

$$S' = \frac{1}{\frac{F_1}{S_1} + \frac{F_2}{S_2}}$$

$$S_1 = \frac{3{,}3}{2{,}2} = 1{,}5 \qquad S_2 = \frac{3{,}3}{2{,}2} \times \frac{4}{2} = 3$$

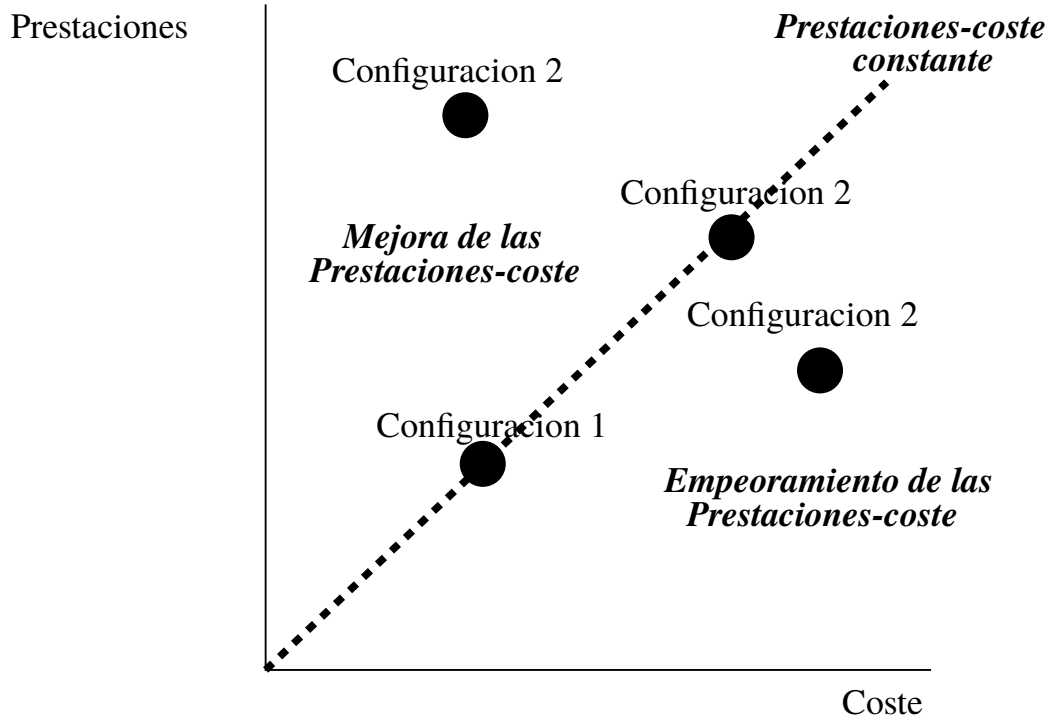
Ejemplos de aplicación de la ley de Amdahl

- En la programación: El tiempo de ejecución de un programa se concentra en una parte del código.
  Principio de localidad de referencia: el 90 % del tiempo se ejecuta el 10 % del código
  $\Rightarrow$ Conviene optimizar la parte del código más frecuente
- el diseño del juego de instrucciones: ¿cuáles son las instrucciones más frecuentes?
- los sistemas con múltiples procesadores: ¿qué fracción de los programas se pueden ejecutar en paralelo?
- en general, al diseño de las partes del computador

La relación prestaciones–coste

Mide la relación entre la prestaciones alcanzadas y el coste (en sentido genérico: precio, watios, . . .) de una configuración dada. Permite comparar varias alternativas.

Ejemplo: Dadas dos configuraciones de un sistema:



Índice

Workloads

Para medir prestaciones, hay que escoger programas de prueba.
Hay que disponer del código fuente para compilarlo para cada computador bajo análisis
La mejor opción:

- Programas reales

Otras opciones, desacreditadas:

- Núcleos (*kernels*). Fragmentos de código extraídos de programas reales. Exemples: *Livermore Loops* y *Linpack*.
- *Toy benchmarks*. Ejercicios académicos de programación con resultados de ejecución conocidos.
  Ejemplos: *Quicksort*, *Puzzle*, etc.
- *Synthetic Benchmarks*. Programas escritos con el propósito de representar el programa promedio. Ejemplos: *Whetstone*, *Dhrystone*, etc.

Benchmark suites

Composición: programas reales sin interactividad y kernels. Especializados para medir las prestaciones dentro de un perfil de uso.

Ejemplos: SPEC (CPU, gráficos, servidores, etc), MediaBench (multimedia) TPC-xx (Transacciones), EEMBC (empotrados) ...

Actualización: Los programas incluídos en el paquete han de representar el trabajo que hace un usuario típico *en el momento* $\rightarrow$ se actualizan periódicamente.

Reproducibilidad: Las medidas han de ser reproducibles $\rightarrow$ hay que indicar todos los detalles:

- hardware: procesador, cache, memoria, disco, ...
- software: sistema operativo, programas y versiones, datos de entrada, opciones de ejecución, ...

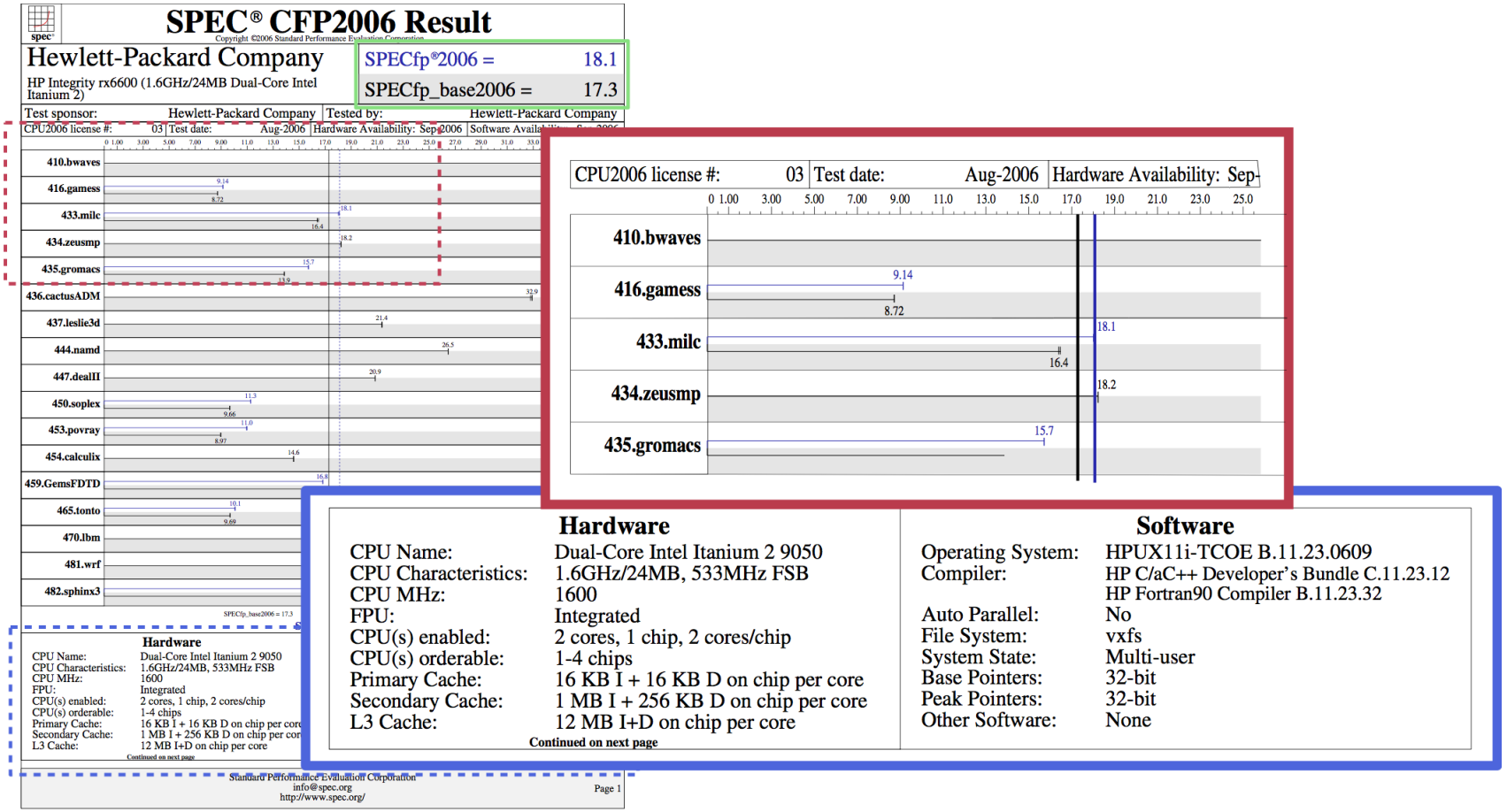
Ejemplo: SPEC CPU95-**CINT95** Benchmarks

Benchmark	Application Area	Specific Task
099.go	Game playing &	Plays the game Go against itself
124.m88ksim	Simulation	Simulates the M88100 processor running test programs
126.gcc	Progamming & compilation	Compiles pre-processed source into optimized SPARC assembly code
129.compress	Compression	Compresses large text files (about 16MB) using adaptive Limpel-Ziv coding
130.li	Language interpreter	Lisp interpreter
132.ijpeg	Imaging	Performs jpeg image compression with various parameters
134.perl	Shell interpreter	Performs text and numeric manipulations (anagrams/prime number factoring)
147.vortex	Database	Builds and manipulates three interrelated databases

Ejemplo: SPEC CPU95-**CFP95** Benchmarks

Benchmark	Application Area	Specific Task
101.tomcatv	Fluid Dynamics & Geometric Translation	Generation of 2D coordinate system around general geometric domains
102.swim	Weather Prediction	Solves shallow water equations using finite difference approximations (SP)
103.su2cor	Quantum Physics	Masses of elementary particles are computed in the Quark-Gluon theory
104.hydro2d	Astrophysics	Hydrodynamical Navier Stokes equations are used to compute galactic jets
107.mgrid	Electromagnetism	Calculation of a 3D potential field
110.applu	Fluid Dynamics/Math	Solves matrix system with pivoting
125.turb3d	Simulation	Simulates turbulence in a cubic area
141.apsi	Weather Prediction	Calculates statistics on temperature and pollutants in a grid
145.fpppp	Chemistry	Performs multi-electron derivatives
146.wave	Electromagnetics	Solve's Maxwell's equations on a cartesian mesh

Ejemplo: Resultados



Comparación de computadores

¿Cómo obtener una medida resumen de la ejecución de varios programas?

$\rightarrow$ Característica de una buena media de tiempos: el valor medio debe ser directamente propocional al tiempo total de ejecución.

- Tiempo total de ejecución:

$$T_T = \sum_{i=1}^{n} \text{Tiempo}_i$$

- Media aritmética.

$$T_A = \frac{1}{n} \sum_{i=1}^{n} \text{Tiempo}_i$$

Comparación de computadores (cont.)

- Tiempo de ejecución ponderado.

$$T_W = \sum_{i=1}^{n} w_i * \text{Tiempo}_i$$

donde w_i representa la frecuencia del programa i en la carga de trabajo.

- (SPEC) la media geométrica de los tiempos de ejecución normalizados a una máquina de referencia $\rightarrow$ R veces más rápido que la referencia:

$$R = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Tiempo}_{\text{ref}}}{\text{Tiempo}_i}}$$

Índice

MIPS

MIPS=Millones de instrucciones por segundo

$$\text{MIPS} = \frac{\text{nº de instrucciones ejecutadas}}{T_{\text{ejecucion}} * 10^6} =$$

$$= \frac{I}{I * CPI * T * 10^6} = \frac{1}{CPI * T * 10^6} = \frac{f}{CPI * 10^6}$$

- Es una magnitud intuitiva, directamente proporcional a las prestaciones.
- No tiene en cuenta el número de instrucciones ejecutadas:
- Depende del programa que se considere. Diferentes programas ejecutan distintas instrucciones, con distinta complejidad y tiempo de ejecución $\rightarrow$ ¡pero el programa ejecutado no se suele indicar!

MIPS (cont.)

- Depende del juego de instrucciones. El mismo programa ejecuta diferente número de instrucciones en cada máquina, según la complejidad de su juego de instrucciones
  $\rightarrow$ no sirven para comparar máquinas con juegos de instrucciones muy distintos.
- ¡Pueden ser inversamente proporcional a las prestaciones! Ejemplo: comparando un computador con y sin coprocesador de coma flotante $\rightarrow$ más prestaciones: **con** coprocesador; más MIPS: **sin** coprocesador.

MIPS (cont.)

Sea un programa que ejecuta n millones de instrucciones con coprocesador y $m > n$ millones de instrucciones sin coprocesador. Los tiempos requeridos por una instrucción de coprocesador y normal son t_c y t, respectivamente, siendo $t_c > t$. Tenemos:

	Sin coprocesador	Con coprocesador
Nº instr:	m instr	n instr
T_{ejec} :	mt	nt_c
MIPS:	$\frac{m}{mt \times 10^6}$	$\frac{n}{nt_c \times 10^6}$

Como $t_c > t \rightarrow \frac{1}{t} > \frac{1}{t_c} \Rightarrow \text{MIPS}_{\text{sin copro}} > \text{MIPS}_{\text{con copro}}$

MFLOPS

Millones de operaciones en coma flotante por segundo.

$$\text{MFLOPS} = \frac{\text{nº de op. en coma flotante del programa}}{T_{\text{ejecucion}} * 10^6}$$

- Contabiliza operaciones en lugar de instrucciones: El mismo programa ejecutándose en diferentes arquitecturas hará distinto número de instrucciones pero las mismas operaciones en CF.
- No es aplicable a programas que no hagan uso de coma flotante, como son los compiladores o los procesadores de textos.

MFLOPS (cont.)

- Aun así, depende del repertorio de instrucciones de coma flotante de la máquina, que no es siempre el mismo. Ejemplo: CRAY-2 no tiene div; M68882 sí y además sqr, sin y cos.
  $\rightarrow$ el número de operaciones en coma flotante a nivel máquina del mismo programa ya no permanece constante.
  Solución: Contabilizar las operaciones de coma flotante del *código fuente*.
- Programas diferentes ejecutan distintas operaciones en coma flotante y todas las operaciones en coma flotante no tienen el mismo coste.