

Arquitectura de Computadores II

Clase #9

Facultad de Ingeniería
Universidad de la República

Instituto de Computación
Curso 2010

Veremos...

- Rendimiento

Rendimiento (Performance)

- Ver a través de la “niebla” del marketing
- Tomar opciones inteligentes
- Entender el problema/motivación
 - ¿Por qué algunos programas se comportan mejor en determinado hardware y no en otro?
 - ¿Qué factores del rendimiento de un sistema dependen del hardware?
 - (ejemplo: se necesita una nueva máquina, o un nuevo sistema operativo?)
 - ¿Cómo afecta el set de instrucciones al rendimiento? ¿Y la organización?

¿Cuál es mejor?

Plane	DC to Paris	Speed	Passengers	Throughput (pmp)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- Tiempo de ejecución de una tarea
 - Tiempo de Ejecución, tiempo de respuesta, latencia
- Tareas por día, hora, semana, seg, ns ...
 - Throughput

Rendimiento del Computador: TIEMPO

- Tiempo de ejecución o respuesta (latencia)
 - ¿Cuánto tiempo lleva ejecutar un trabajo?
 - ¿Cuánto hay que esperar por el resultado de una consulta a la base de datos?
- Throughput
 - ¿Cuántos trabajos pueden ejecutarse a la vez?
 - ¿Cuál es el tiempo de ejecución total de varias tareas?
 - ¿Cuántos trabajos terminan por unidad de tiempo?
- Preguntas típicas
 - ¿Cuánto se mejora si agregamos un procesador al servidor?
 - ¿Qué mejora se obtiene agregando una nueva máquina?

MIPS

- MIPS (millones de instrucciones por segundo)
- $MIPS = \text{Recuento de Instrucciones} / (\text{Tiempo_Ejec} \times 10^6)$
- Ejemplo
 - Dos diferentes compiladores se prueban en una máquina de 100 MHz con tres clases de instrucciones A, B, y C, que requieren uno, dos y tres ciclos, respectivamente. Ambos compiladores producen código para un programa que al correrse provoca la ejecución de:
 - Compilador 1: 5 millones de instrucciones A, 1 millón de B y 1 millón de C
 - Compilador 2: 10 millones de instrucciones A, 1 millón de B y 1 millón de C
 - Cual ejecuta más rápido?
 - Cual es "mejor" de acuerdo a los MIPS?

Tiempo de Ejecución

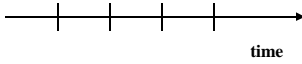
- Tiempo transcurrido
 - Mide "todo"
 - Accesos a memoria y disco, E/S, etc
 - Una medida útil, pero difícil de usar para comparar sistemas
- Tiempo de CPU
 - No cuenta E/S, o el tiempo que se emplea en ejecutar otros programas (en sistemas multitarea)
 - Se puede dividir en tiempo del sistema y tiempo de usuario

Definición de Performance

- Performance: unidades de ejecución por unidad de tiempo
 - Más grande es mejor
- $\text{Performance}(x) = 1 / \text{Execution_time}(x)$
- "X es n veces más rápido que Y" significa

$$n = \frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}$$

Ciclos de Reloj

- Tiempo discreto en "ticks" de reloj: 
- Tiempo de ciclo = tiempo entre ticks = período (seg/ciclo)
- Frecuencia del reloj (ciclos / seg)
 - (1 Hz = 1 ciclo/seg)
 - Un reloj de 200 Mhz tiene un período de:

$$\frac{1}{200 \times 10^6 \text{ ciclo/seg}} = \frac{1}{2} \times 10^{-8} \text{ seg/ciclo} = 5 \times 10^{-9} \text{ seg} = 5 \text{ nanosegs/ciclo}$$

- Período constante => #ciclos de reloj es una medida alternativa para el tiempo de ejecución

$$\text{Tiempo de CPU} = \# \text{ciclos} \times \text{período}$$

$$\text{Tiempo de CPU} = \frac{\# \text{ciclos}}{\text{frecuencia}}$$

Ecuación de performance de CPU

$$\text{Tiempo de CPU} = \# \text{ciclos} \times \text{período}$$

$$\text{Tiempo de CPU} = \# \text{instrucciones} \times \frac{\# \text{ciclos}}{\# \text{instrucciones}} \times \text{período}$$

Tiempo CPU	=	Segundos	=	Instrucciones	x	Ciclos	x	Segundos
		Programa		Programa		Instrucciones		Ciclo

- Identificamos tres factores que afectan la performance
 - **CPI**: # ciclos / # instrucciones
 - Recuento de instrucciones: # de instrucciones del programa
 - Frecuencia (o período): # de ciclos por segundo
- Un error habitual es considerar que uno o dos de estos factores son determinantes de la performance
 - Caso típico: frecuencia de reloj
 - MIPS = Frecuencia del Reloj / (CPI x 10⁶)

Aspectos de la Performance de la CPU

	Recuento Inst	CPI	Ciclo de Reloj
Programa	X		
Compilador	X	(X)	
Set de Inst.	X	X	
Organización		X	X
Tecnología			X

Ciclos por Instrucción

En un set de n (categorías de) instrucciones

$$\#ciclos = \sum_{j=1}^n \#ciclos_j = \sum_{j=1}^n \frac{\#ciclos_j}{\#ocurrencias_j} \times \#ocurrencias_j$$

$$CPI = \frac{\#ciclos}{\#instrucciones} = \sum_{j=1}^n \frac{\#ciclos_j}{\#ocurrencias_j} \times \frac{\#ocurrencias_j}{\#instrucciones}$$

“Frecuencia de Instrucciones”

$$CPI = \sum_{j=1}^n CPI_j \times F_j, \text{ siendo } CPI_j = \frac{\#ciclos_j}{\#ocurrencias_j}, F_j = \frac{\#ocurrencias_j}{\#instrucciones}$$

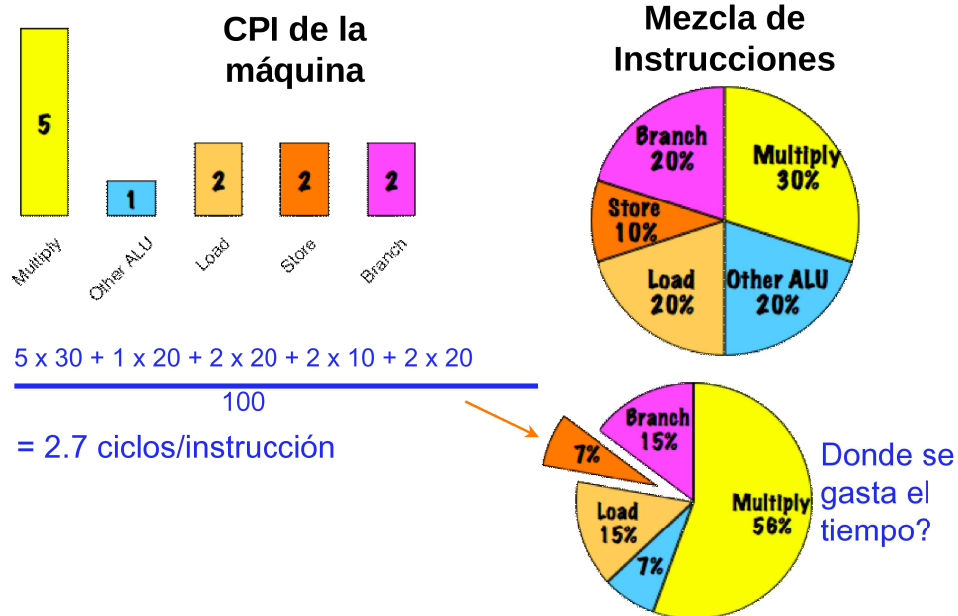
CPI_j debe medirse, ya que la referencia técnica no tiene en cuenta, por ejemplo, misses de cache o retardos en pipeline.

Ejemplo: Cálculo de CPI (1/2)

Op	Freq	CPI	$F_j * CPI_j$ (% Tiempo)
ALU	50%	1	.5 (33%)
Load	20%	2	.4 (27%)
Store	10%	2	.2 (13%)
Branch	20%	2	.4 (27%)
			<u>1.5</u>

Mezcla Típica

Ejemplo: Cálculo de CPI (2/2)



Ejemplo: Impacto del Branch Stall

- Se asume $CPI = 1.0$ ignorando bifurcaciones (ideal)
 - En realidad se produce un "stall" de 3 ciclos por bifurcación
 - Si 30% bifurcaciones, "stall" 3 ciclos en 30%
- | Op | Freq | Cycles | $F_j \times CPI_j$ | (% Tiempo) |
|---------|------|--------|--------------------|------------|
| Otras | 70% | 1 | .7 | (37%) |
| Bifurc. | 30% | 4 | 1.2 | (63%) |
- $CPI_{real} = 1.9$
 - La máquina real es $1/1.9 = 0.52$ veces "más rápida" (o sea el doble de lenta!)

Ley de Amdahl (1/3)

- Aceleración (Speedup) debida a mejora E:

$$Speedup(E) = \frac{ExTime \text{ sin } E}{ExTime \text{ con } E} = \frac{Performance \text{ con } E}{Performance \text{ sin } E}$$



La aceleración global depende no sólo del factor de aceleración de la mejora, sino también del tiempo que se aprovecha esa mejora

- Ejemplo:
 - Consideramos un programa que se ejecuta en 100 segs; la multiplicación es responsable de 80 segs. del total. ¿Cuánto debemos mejorar la multiplicación para que el programa se ejecute 4 veces más rápido?
- Principio: mejorar el caso más común

Ley de Amdahl (2/3)

$$T_{\text{old}} = T_{\text{no afectado}} + T_{\text{afectado old}}$$

$$T_{\text{new}} = T_{\text{no afectado}} + T_{\text{afectado new}}$$

$$T_{\text{new}} = T_{\text{no afectado}} + \frac{T_{\text{afectado old}}}{\text{speedup}_{\text{afectado}}}, \text{ donde } \text{speedup}_{\text{afectado}} = \frac{T_{\text{afectado old}}}{T_{\text{afectado new}}}$$

$$T_{\text{new}} = (1 - \text{Fracción}_{\text{afectada}}) T_{\text{old}} + \frac{\text{Fracción}_{\text{afectada}} T_{\text{old}}}{\text{speedup}_{\text{afectado}}}$$

$$\text{speedup}_{\text{total}} = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{1}{(1 - \text{Fracción}_{\text{afectada}}) + \frac{\text{Fracción}_{\text{afectada}}}{\text{speedup}_{\text{afectado}}}}$$

Observación: $\text{Fracción}_{\text{afectada}}$ es con respecto a T_{old}

Ley de Amdahl (3/3)

$$\text{speedup}_{\text{total}} = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{1}{(1 - \text{Fracción}_{\text{afectada}}) + \frac{\text{Fracción}_{\text{afectada}}}{\text{speedup}_{\text{afectado}}}}$$

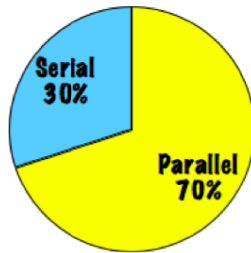
■ Ejemplo:

- Instrucciones de Punto Flotante mejorada para correr a 2X; pero únicamente 10% del tiempo se consume en instrucciones de Punto Flotante

$$\text{Speedup}_{\text{total}} = \frac{1}{0.9 + 0.1 / 2} = 1.053$$

Ley de Amdahl: ejemplo (1/2)

Queremos ejecutar un programa en N CPUs

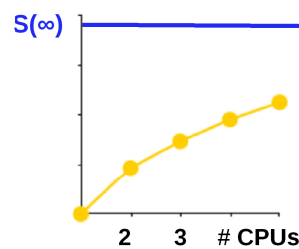
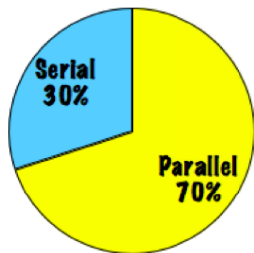


Se emplea el 30% del tiempo ejecutando código que no se puede paralelizar

Calcularemos aceleración para $N = 2, 3, 4, 5, \infty$

CPUs	2	3	4	5	∞
Aceleración					

Ley de Amdahl: ejemplo (2/2)



$$S = \frac{1}{(30\% + (70\% / N)) / 100\%}$$

CPUs	2	3	4	5	∞
Aceleración	1.54	1.85	2.1	2.3	3.3

Métricas de Performance

- Kernels
 - Extractos “clave” de programas reales
- Synthetic benchmarks
 - Filosofía similar a kernels, intentan promediar frecuencia de operaciones de un gran conjunto de programas típicos
 - Whetstone (numérico), Dhrystone (E/S datos)
- Fácil de estandarizar, pero...
 - Pueden ser “violados”
- Programas reales
 - Se usan programas que representan una carga de trabajo típica, para determinada clase de aplicaciones
 - Compiladores de C, procesadores de texto, herramientas CAD

Benchmarks

- Benchmark Suites
 - Combinaciones de benchmarks, enfocados en algún aspecto relevante: SPECCPU2000, SPECint1997, SPECfp1992, SPECWeb, SPECSFS, TPC-C, etc.
- SPEC (System Performance Evaluation Cooperative)
 - Las compañías se ponen de acuerdo en un conjunto de programas y entradas reales
 - Aún pueden ser “violados”...
 - Buen indicador de performance (y tecnología de compiladores)
- SPEC89, 92, 95, 2000, 2006
- Los programas varían entre generaciones de SPEC
- Más información en <http://www.spec.org/>

SPEC CPU2000: benchmarks de enteros

164.gzip	C	Compression
175.vpr	C	FPGA Circuit Placement and Routing
176.gcc	C	C Programming Language Compiler
181.mcf	C	Combinatorial Optimization
186.crafty	C	Game Playing: Chess
197.parser	C	Word Processing
252.eon	C++	Computer Visualization
253.perlbmk	C	PERL Programming Language
254.gap	C	Group Theory, Interpreter
255.vortex	C	Object-oriented Database
256.bzip2	C	Compression
300.twolf	C	Place and Route Simulator

SPEC CPU2000: benchmarks de punto flotante

168.wupwise	Fortran 77	Physics / Quantum Chromodynamics
171.swim	Fortran 77	Shallow Water Modeling
172.mgrid	Fortran 77	Multi-grid Solver: 3D Potential Field
173.applu	Fortran 77	Parabolic / Elliptic Partial Differential Equations
177.mesa	C	3-D Graphics Library
178.galgel	Fortran 90	Computational Fluid Dynamics
179.art	C	Image Recognition / Neural Networks
183.quake	C	Seismic Wave Propagation Simulation
187.facerec	Fortran 90	Image Processing: Face Recognition
188.amp	C	Computational Chemistry
189.lucas	Fortran 90	Number Theory / Primality Testing
191.fma3d	Fortran 90	Finite-element Crash Simulation
200.sixtrack	Fortran 77	High Energy Nuclear Physics Accelerator Design
301.apsi	Fortran 77	Meteorology: Pollutant Distribution

La CPU no es todo... ...otros benchmarks SPEC

- SPECint y SPECfp miden tareas de computación intensiva, enfatizando los siguientes elementos de la arquitectura:
 - CPU
 - Arquitectura de memoria
 - Compiladores
- NO atacan otros elementos tales como el sistema operativo, la red, los gráficos o el subsistema de E/S. Existen otros benchmarks, por ejemplo para:
 - Graphics and Workstation Performance
 - High Performance Computing, OpenMP, MPI
 - Java Client/Server
 - Mail Servers
 - Network File System
 - Web Servers

Reportes de performance

- Los reportes deben permitir **reproducir** las mediciones de performance

SPEC [®] CINT2006 Result																											
Copyright ©2006 Standard Performance Evaluation Corporation																											
Apple Computer, Inc.	SPECint [®] 2006 = 10.1																										
iMac 20-inch 2.0GHz Intel Core Duo	SPECint_base2006 = 10.1																										
Test sponsor: Apple Computer, Inc. Tested by: Apple Computer, Inc.																											
CPU2006 license #: 777 Test date: Apr-2006 Hardware Availability: Jan-2006 Software Availability: Apr-2006																											
SPECint_base2006 = 10.1																											
SPECint2006 = 10.1																											
<table border="1"> <thead> <tr> <th>Hardware</th><th>Software</th></tr> </thead> <tbody> <tr> <td>CPU Name: Intel Core Duo T2500</td><td>Operating System: Mac OS X, v10.4.0</td></tr> <tr> <td>CPU Characteristics: Integrated</td><td>Compiler: Intel C++ Compiler for Mac OS X, v9.1 (Build 20060323)</td></tr> <tr> <td>CPU MHz: 2000</td><td>Auto Parallel: No</td></tr> <tr> <td>CPU(s) enabled: 2 cores, 1 chip, 2 cores/chip</td><td>File System: HPFS</td></tr> <tr> <td>CPU(s) orderable: 1 chip</td><td>System State: Default</td></tr> <tr> <td>Primary Cache: 32 KB L1 + 32 KB D on chip per core</td><td>Base Pointers: 32 bit</td></tr> <tr> <td>Secondary Cache: 2 MB L2 on chip per chip</td><td>Peak Pointers: 32 bit</td></tr> <tr> <td>L3 Cache: None</td><td>Other Software: None</td></tr> <tr> <td>Other Cache: None</td><td></td></tr> <tr> <td>Memory: 1 GB DDR2 (667 MHz, CL5)</td><td></td></tr> <tr> <td>Disk Subsystem: SATA Maxtor 6L250M0, 250 GB</td><td></td></tr> <tr> <td>Other Hardware: None</td><td></td></tr> </tbody> </table>		Hardware	Software	CPU Name: Intel Core Duo T2500	Operating System: Mac OS X, v10.4.0	CPU Characteristics: Integrated	Compiler: Intel C++ Compiler for Mac OS X, v9.1 (Build 20060323)	CPU MHz: 2000	Auto Parallel: No	CPU(s) enabled: 2 cores, 1 chip, 2 cores/chip	File System: HPFS	CPU(s) orderable: 1 chip	System State: Default	Primary Cache: 32 KB L1 + 32 KB D on chip per core	Base Pointers: 32 bit	Secondary Cache: 2 MB L2 on chip per chip	Peak Pointers: 32 bit	L3 Cache: None	Other Software: None	Other Cache: None		Memory: 1 GB DDR2 (667 MHz, CL5)		Disk Subsystem: SATA Maxtor 6L250M0, 250 GB		Other Hardware: None	
Hardware	Software																										
CPU Name: Intel Core Duo T2500	Operating System: Mac OS X, v10.4.0																										
CPU Characteristics: Integrated	Compiler: Intel C++ Compiler for Mac OS X, v9.1 (Build 20060323)																										
CPU MHz: 2000	Auto Parallel: No																										
CPU(s) enabled: 2 cores, 1 chip, 2 cores/chip	File System: HPFS																										
CPU(s) orderable: 1 chip	System State: Default																										
Primary Cache: 32 KB L1 + 32 KB D on chip per core	Base Pointers: 32 bit																										
Secondary Cache: 2 MB L2 on chip per chip	Peak Pointers: 32 bit																										
L3 Cache: None	Other Software: None																										
Other Cache: None																											
Memory: 1 GB DDR2 (667 MHz, CL5)																											
Disk Subsystem: SATA Maxtor 6L250M0, 250 GB																											
Other Hardware: None																											
Standard Performance Evaluation Corporation info@spec.org http://www.spec.org/																											
Page 1																											

Resumen de performance (1/3)

- Interesa tener UN número que resuma la performance de un sistema
- Media Aritmética basada en el tiempo de ejecución:
 - $\Sigma(T_i)/n$
 - T_i es el tiempo de ejecución del i-ésimo de los n programas que componen la carga de trabajo
- Pregunta: cual es la mezcla de programas mas adecuada para medir la performance? "Pesas" todos lo mismo?

Resumen de performance (2/3)

- Media Aritmética ponderada
 - $\Sigma(W_i * T_i)$
- Pero, cómo se eligen los pesos?
- SPEC benchmarks: SpecRatio = Tiempo de Ejecución Normalizado con respecto a una "máquina canónica", (Y corre n veces más rápido que X)
 - En cociente de ratios de dos sistemas NO influye la referencia!
- Para promediar razones (como SpecRatio) se usa la Media Geométrica:
 - $(\Pi \text{ muestra}_j)^{1/n} = \Pi (T_j / N_j)^{1/n}$
 - La media geométrica de las razones T_j / N_j es igual a la razón entre las medias geométricas de los T_j y los N_j
 - La razón entre las medias geométricas es igual a la media geométrica de las razones entre muestras una a una

Resumen de performance (3/3)

	Sistema A	Sistema B	Sistema C	W(1)	W(2)
Programa P1 (segs)	1	10	20	0.5	0.909
Programa P2 (segs)	1000	100	20	0.5	0.091
Tiempo total (segs)	1001	110	40		
Media aritmética: W(1)	500.5	55	20		
Media aritmética: W(2)	91.909	18.19	20		
Media aritmética: W(3)	1.999	10.09	20		

- W(1) pesa igual P1 y P2
- W(2) peso inversamente proporcional a Tiempo de Ejecución en B
- W(3) peso inversamente proporcional a Tiempo de Ejecución en A
- Comparar resultados de diferentes medias:

	Normalizado a A			Normalizado a B			Normalizado a	
	A	B	C	A	B	C	A	B
Programa P1	1	10	20	0.1	1	2	0.05	0.5
Programa P2	1	0.1	0.02	10	1	0.2	50	5
Media aritmética	1	5.05	10.01	5.05	1	1.1	25.025	2.75
Media geométrica	1	1	0.632	1	1	0.632	1.58114	1.581
Tiempo total	1	0.110	0.040	9.1	1	0.364	25.025	2.75

Evaluación de Performance

- Para bien o para mal, los benchmarks son tomados en cuenta.....
- El diseño se debe orientar a mejorar la performance de las cargas de trabajo (programas) reales, y no solamente los programas que ayudan a vender!
- Para una arquitectura dada la mejora de la performance viene de:
 - Incremento de la frecuencia de reloj (sin afectar los CPI!)
 - Mejora en la organización del procesador para baja los CPI
 - Mejora en los compiladores para bajar CPI y/o recuento de instrucciones