Tema 1. Evaluación de prestaciones

ESTRUCTURA DE COMPUTADORES

Grupo de Arquitectura de Computadores (GAC)

Índice

Índice

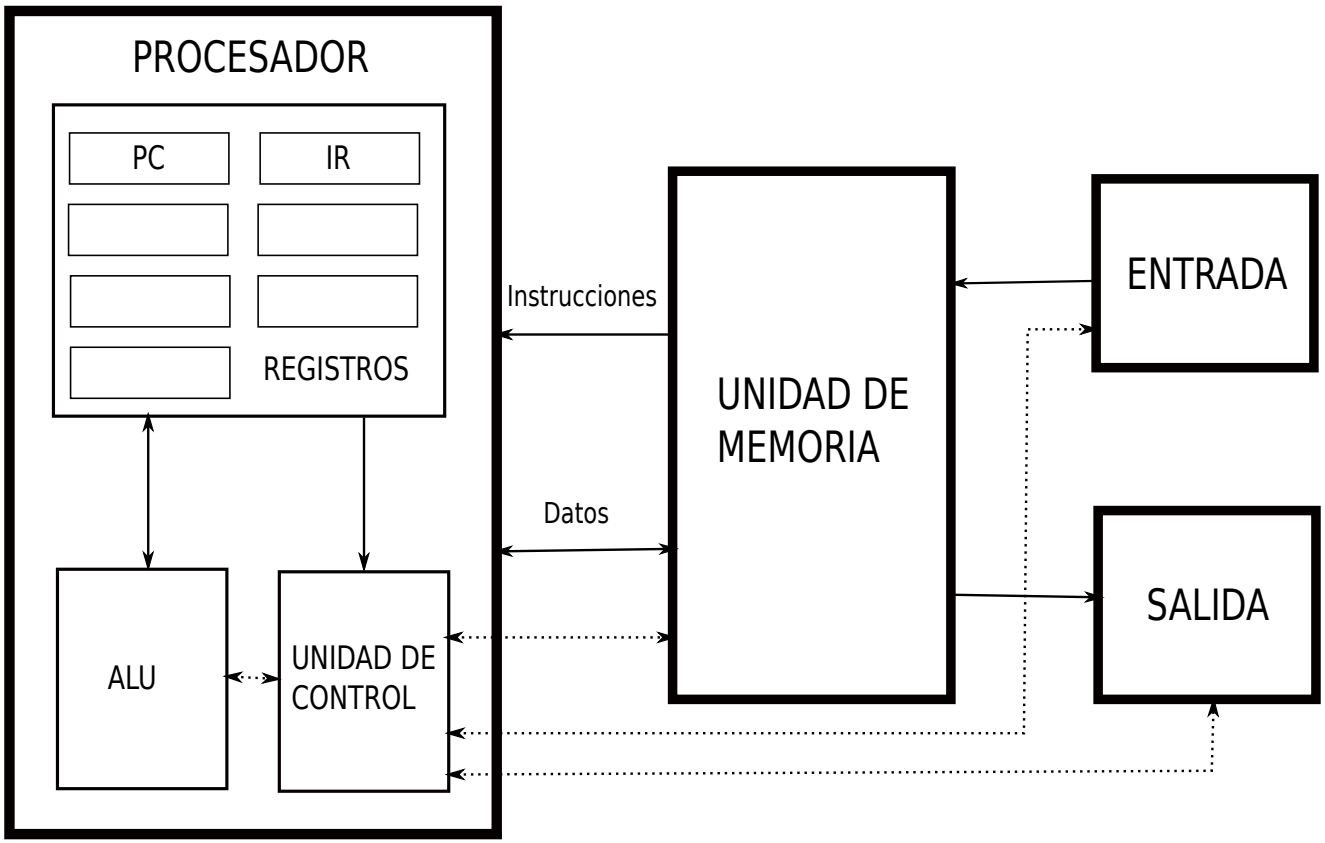
Objetivos del tema

En este tema se estudiará:

- La importancia de medir el rendimiento en los sistemas computadores
- Las métricas más populares de rendimiento en los computadores
- Qué son y cómo se usan los programas de prueba (*benchmarks*)
- La ley de Amdahl para estimar la aceleración que se puede obtener con una mejora

Índice

Introducción



Introducción

Cada componente refleja sus prestaciones atendiendo a parámetros diferentes. Parámetros a tener en cuenta:

- **Memoria**: capacidad, latencia, ancho de banda.
- **Bus**: número de líneas de comunicación y frecuencia de transmisión de datos (ancho de banda)
- **Procesador**: tiempo de CPU

Índice

Métricas de rendimiento

Tiempo de respuesta (latencia o tiempo de ejecución)

Tiempo entre el inicio y el final de una tarea. Para maximizar el rendimiento se minimiza el tiempo de ejecución de una tarea:

$$R_x = \frac{1}{tiempo\ de\ ejecucion_x}$$

Productividad

Cantidad de trabajo hecho en un cierto tiempo

Métricas de rendimiento

- **Teléfonos inteligentes** *(smartphones)*: tamaño, latencia, vida de la batería, coste, ...
- **Servidores**: productividad, latencia, consumo energético, calidad del servicio, fiabilidad, ...
- **Portátiles**: tamaño, latencia, vida de la batería, coste
- **PCs**: latencia, coste, ...

Diferente métrica implica diferente diseño

Métricas de rendimiento

Tiempo de respuesta

Incluye los accesos a disco, accesos a memoria, actividades de E/S, y la carga adicional del S.O.

Tiempo de ejecución de la CPU

Tiempo que la CPU dedica a ejecutar una tarea concreta

$$T_{cpu} = T_{user} + T_{s.o.}$$

Ejemplo

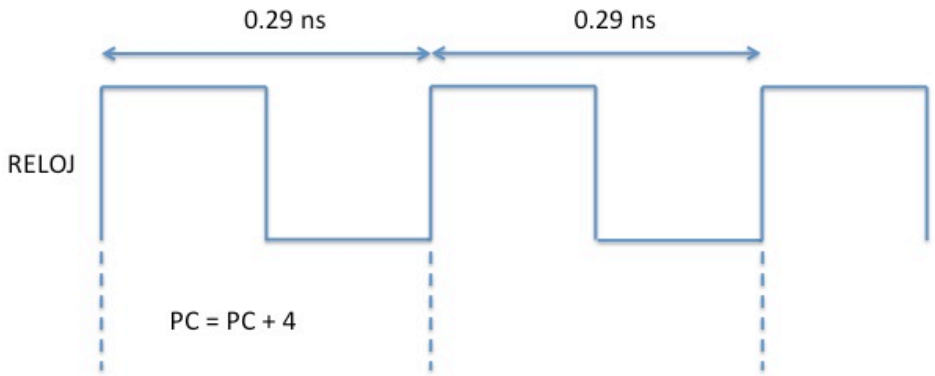
```
>time ./programa datos
real 0m11.588s  -> tiempo total
user 0m3.735s   -> tiempo de CPU del usuario
sys  0m3.286s   -> tiempo de CPU del S.O.
```

Ciclo de reloj

Los procesadores tienen un reloj que funciona a una frecuencia concreta y determina el momento en que tienen lugar los eventos

Ejemplo

Intel© Core^{TM} i7-990X processor Extreme Edition: 3.46 GHz core speed



Métricas de rendimiento

Tiempo de ejecución de CPU

$$Tiempo\ de\ CPU = ciclos\ de\ CPU \times tiempo\ de\ ciclo$$

$$Tiempo\ de\ CPU = \frac{ciclos\ de\ CPU}{frecuencia\ de\ reloj}$$

El compilador genera las instrucciones para ejecutar y el procesador las ejecuta. El número de ciclos de reloj requerido por un programa es:

$$ciclos\ de\ CPU = instrucciones \times media\ de\ ciclos\ por\ instruccion$$

Métricas de rendimiento

Ciclo de reloj por instrucción (CPI)

Es el número medio de ciclos de reloj que una instrucción necesita para ejecutarse

Instrucciones diferentes pueden necesitar diferente cantidad de ciclos de reloj. El CPI proporciona una media:

$$CPI = \frac{\sum_{i=i}^{m} CPI_i NI_i}{N}$$

donde NI_i es el número total de instrucciones tipo i, CPI_i es el número de ciclos de reloj para esa clase de instrucciones, N es el número total de instrucciones, y m es el número total de tipos de instrucciones existentes.

Métricas de rendimiento

Ecuación básica del rendimiento

$$T_{cpu} = N \times CPI \times T = \frac{N \times CPI}{F}$$

donde T es el tiempo de ciclo y F la frecuencia de reloj.

Esta fórmula se puede utilizar para comparar dos realizaciones diferentes o para evaluar un diseño alternativo si se conoce el impacto de los tres parámetros

Métricas de rendimiento

MIPS: millones de instrucciones por segundo

$$MIPS = \frac{N}{T_{cpu} \times 10^6}$$

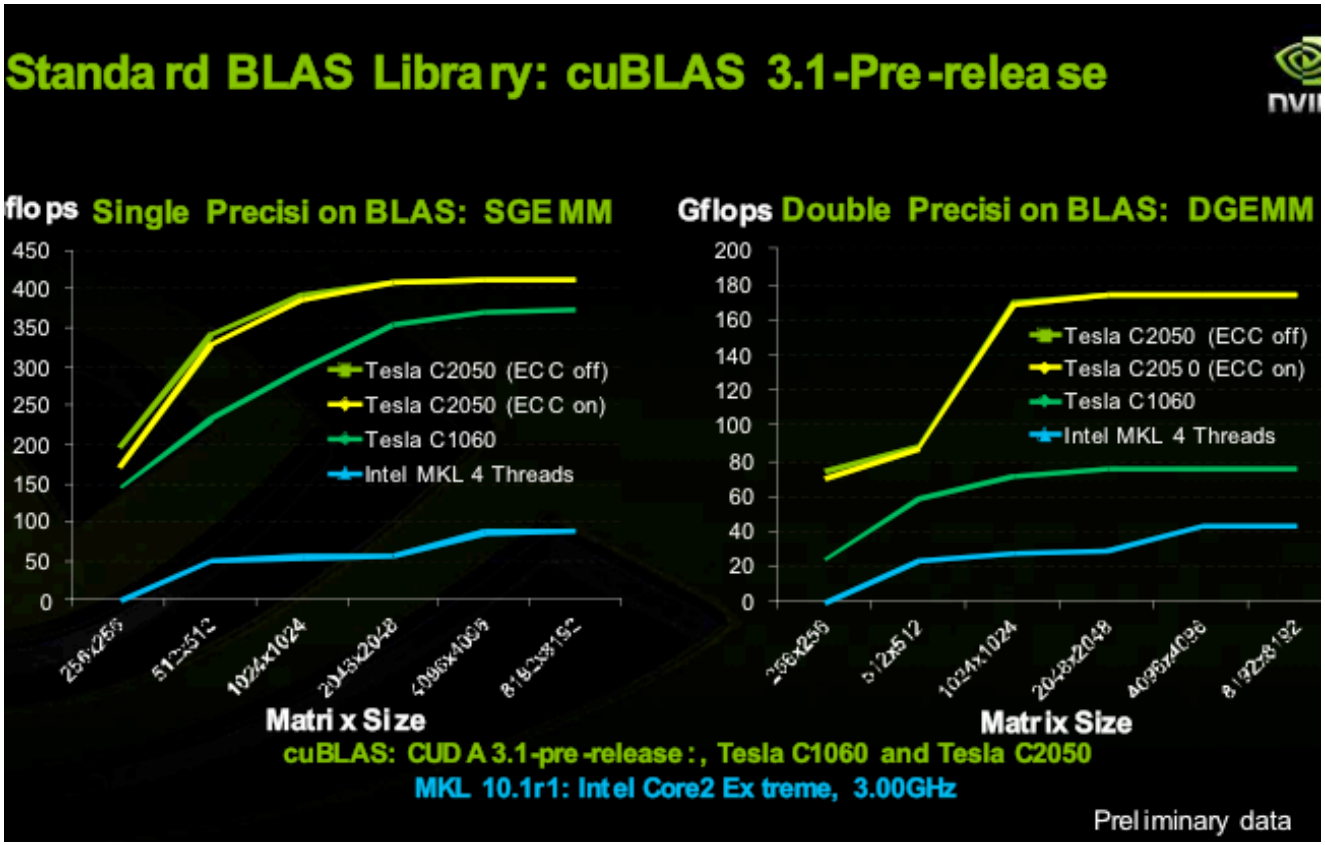
Problemas:

- No se pueden comparar computadores con diferentes repertorios de instrucciones
- Varía entre programas en el mismo computador
- Puede variar inversamente al rendimiento

GFLOPS: miles de millones de operaciones en punto flotante por segundo

$$GFLOPS = \frac{FLOPS}{T_{cpu} \times 10^9}$$

El problema es que sigue siendo dependiente del programa



Índice

Comparación de rendimiento

Comparación de una máquina X con una máquina Y

$$R_x > R_y$$

$$\frac{1}{Texec_x} > \frac{1}{Texec_y}$$

$$Texec_x < Texec_y$$

Relación cuantitativa entre dos máquinas diferentes

$$n = \frac{R_x}{R_y}$$

la máquina X es n veces más rápida que la máquina Y

Aceleración

Aceleración (*Speedup*)

Medida del tiempo logrado tras incorporar una mejora al sistema

$$A = \frac{R_{despues}}{R_{antes}} = \frac{T_{antes}}{T_{despues}}$$

Es fundamental acotar la aceleración potencial a obtener con una posible mejora (que puede ser muy costosa)

Ley de Amdahl

Ley de Amdahl

La mejora obtenida en el rendimiento al utilizar una parte optimizada está limitada por la fracción de tiempo que se puede utilizar esa parte

$$T_{despues} = \frac{T_{afectado}}{A_m} + T_{no\_afectado}$$

$$A = \frac{T_{antes}}{T_{despues}} = \frac{1}{(1 - F_m) + \frac{F_m}{A_m}}$$

donde F_m es la fracción de tiempo afectada por la mejora y A_m es el factor de mejora

Índice

Técnicas de medida

Existen diferentes técnicas de medida para obtener los valores de las métricas de rendimiento escogidas:

- Utilización directa de la información mantenida por el sistema operativo
- Monitores software
- Monitores hardware
- Análisis de programas (*profiling*)

Utilización directa de la información mantenida por el sistema operativo

Los actuales sistemas operativos mantienen información sobre las diferentes tareas que se están ejecutando: utilización de la memoria y del procesador, el número de tareas en el sistema o el tiempo de inicio y de finalización de cada tarea.

Ventaja

No hace falta desarrollar código específico para realizar las medidas de rendimiento.

Desventajas

- Sobrecarga en el acceso a esta información
- Formato poco amigable que impone bastante restricciones
- No mantiene información de todos los eventos

Monitores software

Se ejecuta una aplicación específicamente programanda para recoger y tratar la información necesaria

Tipo de monitores:

- Detección de eventos
  - Cuenta de eventos
  - Traza
- Muestreo

Ventaja

Se adecuan a las necesidades de los usuarios y recogen la información necesaria

Desventaja

Sobrecarga en el acceso a esta información si se produce con una gran frecuencia

Monitores hardware

Los fabricantes incluyen en el hardware monitores que permiten contar con gran precisión una multitud de eventos

Ventaja

- No se consumen recursos (tiempo del procesador y almacenamiento)
- Se pueden monitorizar eventos de muy bajo nivel que son completamente transparente para el software e incluso para el sistema operativo
- Muy precisos

Desventaja

- Se miden magnitudes físicas y no lógicas
- Librerías específicas

Análisis de pogramas (profiling)

- Se emplean medias de tiempo y recursos
- Se utiliza cuando la evaluación del rendimiento es a un nivel alto y en términos de optimización de código o de un sistema

Ventaja

No está sujeta a errores aleatorios como el muestreo.

Desventaja

Sobrecarga la aplicación

Programas de prueba (*benchmarks*)

- Para evaluar el rendimiento de un sistema se usa un conjunto específico de programas de pruebas (*benchmarks*)
- La práctica aceptada hoy es usar una selección de programas de aplicación reales
- Los programas de prueba forman una carga con la que el usuario espera predecir el rendimiento de la carga real del sistema
- Para indicar las medidas del rendimiento se debe hacer un informe con una lista detallada de forma que otra persona pueda reproducir los resultados (versión del s.o., compilador, entradas, configuración de la máquina, etc...)

Programas de prueba (*benchmarks*)

- El rendimiento de una máquina se suele dar como un único número
- El grupo de programas de prueba más popular y completo es el SPEC (*Standard Performance Evaluation Corporation - www.spec.org*)
- La última versión de los SPEC para el procesador es el banco de pruebas *SPEC CPU2017*
- Los programas SPEC se dividen en dos grupos:
  - con carga computacional intensa en punto flotante (*SPECfp2017*)
  - con carga computacional intensa en enteros (*SPECint2017*)

Programas de prueba (*benchmarks*)

SPEC2006 benchmark description	Benchmark name by SPEC generation				
	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

Programas de prueba (*benchmarks*)

SPEC CPU2017 has 43 benchmarks, organized into 4 suites:

SPECrate 2017 Integer	SPECspeed 2017 Integer	Language [1]	KLOC [2]	Application Area
500.perlbench_r	600.perlbench_s	C	362	Perl interpreter
502.gcc_r	602.gcc_s	C	1,304	GNU C compiler
505.mcf_r	605.mcf_s	C	3	Route planning
520.omnetpp_r	620.omnetpp_s	C++	134	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	96	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	33	General data compression

SPECrate 2017 Floating Point	SPECspeed 2017 Floating Point	Language [1]	KLOC [2]	Application Area
503.bwaves_r	603.bwaves_s	Fortran	1	Explosion modeling
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	257	Physics: relativity
508.namd_r		C++	8	Molecular dynamics
510.parest_r		C++	427	Biomedical imaging: optical tomography with finite elements
511.povray_r		C++, C	170	Ray tracing
519.lbm_r	619.lbm_s	C	1	Fluid dynamics
521.wrf_r	621.wrf_s	Fortran, C	991	Weather forecasting
526.blender_r		C++, C	1,577	3D rendering and animation
527.cam4_r	627.cam4_s	Fortran, C	407	Atmosphere modeling
	628.pop2_s	Fortran, C	338	Wide-scale ocean modeling (climate level)
538.imagick_r	638.imagick_s	C	259	Image manipulation
544.nab_r	644.nab_s	C	24	Molecular dynamics
549.fotonik3d_r	649.fotonik3d_s	Fortran	14	Computational Electromagnetics
554.roms_r	654.roms_s	Fortran	210	Regional ocean modeling

[1] For multi-language benchmarks, the first one listed determines library and link options (details)

[2] KLOC = line count (including comments/whitespace) for source files used in a build / 1000

Programas de prueba (*benchmarks*)

Los tiempos de ejecución de un sistema deben ser normalizados, para lo que se usa un sistema como referencia

- Para *SPEC CPU2017* el sistema de referencia usado es el servidor Sun Fire V490 con un procesador 2100 MHz UltraSPARC-IV+ (año 2006)

Ratio SPEC para un programa

$$velocidad\ SPEC = \frac{T_{compRef}}{T_{compTest}}$$

El test se repite para todos los programas del conjunto SPEC y se computa la media geométrica de los resultados.

Ratio SPEC para un conjunto

$$velocidad\ SPEC = \sqrt[n]{\prod_{i=1}^{n} SPEC_i}$$

Índice

Conclusiones

- La medida más importante del rendimiento de un computador es la rapidez con la que puede ejecutar programas
- En el caso del procesador, esta rapidez vendrá determinada por:
  - la organización de su hardware ($=>$ *CPI*)
  - su ciclo de reloj ($=>$ T_{ciclo})
  - sus instrucciones de lenguaje máquina ($=>$ *NI* y *CPI*)
  - el compilador que se utilice ($=>$ *NI*)
- Además del tiempo de ejecución, existen otras métricas del rendimiento como los MIPS y GFLOPS
- Los programas de prueba (*benchmarks*) nos permiten comparar tiempos de ejecución entre máquinas diferentes y obtener una medida normalizada del rendimiento de los computadores
- La ley de Amdahl permite acotar la aceleración que se obtendrá al mejorar alguno de los subsistemas del conjunto

Bibliografía

- David A. Patterson y John L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface*. Morgan Kaufmann 2009
- Marta Beltrán y Antonio Guzmán. *Diseño y Evaluación de Arquitecturas de Computadores*. Prentice Hall, 2010