Módulo 3.1.3
Rendimiento del procesador

Ingeniería Técnica en Informática de Sistemas
Facultad de Informática
Universidad Complutense de Madrid

Introducción (1/2)

- Al diseñar un computador debemos tener en cuenta dos parámetros importantes
  - Coste
  - Rendimiento
- Dependiendo del equilibrio que se haga de ellos se creará desde un ordenador personal (barato y rendimiento medio) hasta un supercomputador (caro y con un rendimiento muy alto)
- ¿Cuál es la medida apropiada de rendimiento?

Introducción (2/2)

- El usuario de un procesador puede decir que el procesador A es mejor que el procesador B cuando A ejecuta su programa en menor tiempo que B.
- El responsable de un centro de cálculo entenderá que A es mejor que B si es capaz de ejecutar mayor número de trabajos por unidad de tiempo.
- El primero estará interesado en el tiempo de respuesta (**response time**) del procesador mientras que el segundo lo estará en la productividad (**throughput**). Pero *en ambos casos la clave es el tiempo*.

Tiempo de ciclo y frecuencia

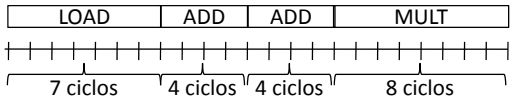
- Todos los computadores utilizan una señal periódica que determina el momento en que tienen lugar los eventos hardware: dicha señal se llama reloj
  - Tiempo de ciclo: tiempo que transcurre entre dos ticks (medido en s)
  - Frecuencia de reloj: la inversa del tiempo de ciclo (medido en Hz = s^{-1})

Ejemplo

Un computador trabajando a 200 MHz (frecuencia) tiene un tiempo de ciclo de: $\frac{1}{200MHz} = \frac{1}{2\times10^{8}} s = 0.5\times10^{-8} s = 5\times10^{-9} s = 5ns$

Ciclos de reloj por programa (1/2)

- ¿(ciclos de reloj por programa) = (instrucciones por programa)?
- **NO**: diferentes instrucciones tardan diferentes cantidades de tiempo:
  - La multiplicación tarda más ciclos que la suma
  - Operaciones en punto flotante tardan más que operaciones en punto fijo
  - Instrucciones con acceso a memoria tardan más que con acceso a registros



Ciclos de reloj por programa (2/2)

- Los "ciclos promedio por instrucción" (CPI) se calculan como una suma ponderada del número de ciclos que tarda por separado cada tipo de instrucción

Operación	Frecuencia	Ciclos	CPI	% tiempo
ALU	50 %	1	0.5	23 %
LOAD	20 %	5	1.0	45 %
STORE	10 %	3	0.3	14 %
BRANCH	20 %	2	0.4	18 %
			2.2	

$$\text{Ciclos de reloj por programa} = \text{Instrucciones por programa} \times CPI$$

Tiempo de ejecución (1/4)

- El tiempo de ejecución de un programa lo dividiremos en las siguientes componentes:
  - Tiempo de respuesta
  - Tiempo de CPU
    - Tiempo de CPU usado por el usuario.
    - Tiempo de CPU usado por el S.O.

Tiempo de ejecución (2/4)

- Tiempo de respuesta: Es el tiempo necesario para completar una tarea, incluyendo los accesos al disco, a la memoria, las actividades de E/S y los gastos del S.O. Es el tiempo que percibe el usuario.

Tiempo de ejecución (3/4)

- Tiempo de CPU: Es el tiempo que tarda en ejecutarse un programa, sin tener en cuenta el tiempo de espera debido a la E/S o el tiempo utilizado para ejecutar otros programas. Se divide en:
  - Tiempo de CPU utilizado por el usuario: Es el tiempo que la CPU utiliza para ejecutar el programa del usuario. No se tiene en cuenta el tiempo de espera debido a la E/S o el tiempo utilizado para ejecutar otros programas
  - Tiempo de CPU utilizado por el S.O.: Es el tiempo que el S.O. emplea para realizar su gestión interna.
- Generalmente, nos referiremos a tiempo de CPU como al tiempo de CPU utilizado por el usuario.

Tiempo de ejecución (4/4)

Ejemplo

La función time de Unix produce una salida de la forma: 90.7u 12.9s 2:39 65%
donde:
Tiempo de CPU del usuario = 90.7 segundos
Tiempo de CPU utilizado por el sistema = 12.9 segundos
Tiempo de CPU= 90.7 s+ 12.9 s = 103.6 s
Tiempo de respuesta = 2 minutos 39 segundos =159 segundos
Tiempo de CPU = 65% del tiempo de respuesta = 159 s*0.65 = 103.6 s
Tiempo esperando operaciones de E/S y/o el tiempo ejecutando otras tareas
35% del tiempo de respuesta = 159 segundos*0.35 = 55.6 segundos

El tiempo de respuesta se utiliza como medida del rendimiento del sistema (con el sistema no cargado), mientras que el rendimiento de la CPU normalmente hace referencia al tiempo de CPU del usuario sobre un sistema no cargado.

Cálculo del tiempo de CPU

Tiempo de CPU

$$(\text{Instrucciones por programa}) \times CPI \times (\text{tiempo de ciclo})$$

- ¿De quién dependen las “instrucciones por programa”?
  - programadores / compiladores / arquitectura del repertorio de instrucciones
- ¿De quién depende el “tiempo de ciclo”?
  - tecnología / organización
- ¿De quién dependen el CPI?
  - organización / arquitectura del repertorio de instrucciones

Definición de rendimiento

Rendimiento

La definición del rendimiento (y la métrica más comúnmente utilizada) viene dada por la siguiente expresión

$$R = \frac{1}{(\text{tiempo ejecución})}$$

Existen, además, otras métricas muy populares pero que pueden resultar engañosas por no incluir todas las variables anteriormente estudiadas.

MIPS (1/3)

- MIPS: Millones de instrucciones por segundo

MIPS

$$\text{MIPS} = \frac{(\text{instrucciones por programa})}{(\text{tiempo ejecución}) \times 10^6}$$

MIPS

$$\text{MIPS} = \frac{(\text{frecuencia de reloj})}{(\text{CPI}) \times 10^6}$$

MIPS (2/3)

- Depende del repertorio de instrucciones
  - El tiempo que tarda en ejecutarse una misma instrucción y el nº de instrucciones máquina que genera el compilador puede variar de un repertorio a otro
  - Dos programas "iguales" pueden tener comportamientos opuestos
- La medida de MIPS puede variar mucho de un programa a otro
  - Existen instrucciones que tardan más tiempo en ejecutarse que otras
  - Los programas en los que abundan instrucciones "rápidas" tardan menos en ejecutarse
  - Los programas en los que abundan instrucciones "lentas" tardan más en ejecutarse
- Los fabricantes suelen dar medidas de MIPS muy optimistas
  - Utilizan programas donde predominan instrucciones que tardan poco en ejecutarse

MIPS (3/3)

Ejemplo

Generación	Potencia de cálculo
1ª Válvula de vacío	0.04 MIPS
2ª Transistores	0.2 MIPS
3ª Circuitos integrados	1 MIPS
4ª Microprocesador	10 MIPS
5ª Sistemas basados en micro	100 MIPS

MFLOPS (1/2)

- MFLOPS
  - Millones de instrucciones en punto flotante por segundo.
  - Las instrucciones en punto flotante son las que más tardan en ejecutarse
  - Son una medida algo más fiable del rendimiento real del computador

MIPS

$$\text{MFLOPS} = \frac{(\text{instrucciones FLOP por programa})}{(\text{tiempo ejecución}) \times 10^6}$$

MFLOPS (2/2)

- Depende del repertorio de instrucciones en punto flotante
  - No todos los computadores ofrecen las mismas operaciones en punto flotante
  - Dos programas distintos pueden tener comportamientos opuestos
- Es inútil para muchos programas
  - Los programas enteros (sin operaciones en punto flotante) no pueden medirse en MFLOPS
- Existen instrucciones en punto flotante de distinto tiempo de ejecución
  - Por ejemplo: suma, resta, ... son rápidas; división, seno, exponencial, ... Son lentas
  - Los fabricantes pueden dar también medidas de MFLOPS demasiado optimistas

Evaluación del rendimiento

- El rendimiento se evalúa ejecutando programas reales
  - Programas “de juguete”: De 10 a 100 líneas de código con resultado conocido
  - Programas de prueba (benchmarks) sintéticos: simulan la frecuencia de operaciones y operandos de un abanico de programas reales
  - Fragmentos de programas reales
  - Programas típicos con cargas de trabajo fijas

Ley de Amdahl (1/2)

Ejemplo

Se pretende viajar desde el Yelmo a Ciudad Real y el viaje se realiza en 2 fases: del Yelmo a Madrid (20h caminando) y de Madrid a Ciudad Real (en otro medio de transporte)

Medio de transporte	Viaje desde Madrid (h)	Mejora parcial (h)	Viaje completo (h)	Mejora total (h)
Caminando	50	1	70	1
Bicicleta	20	2.5	40	1.8
Moto	4	12.5	24	2.9
Coche	1.67	30	21.67	3.2
Tren	0.33	150	20.33	3.4

x 150 x 3.4

Cuando se pretende mejorar el rendimiento, debe tenerse en cuenta que la mejora de un aspecto de la máquina no incrementa el rendimiento global en una cantidad proporcional al tamaño de la mejora

Ley de Amdahl (2/2)

Ley de Amdahl

La mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente

$$A = \frac{(\text{tiempo sin mejora})}{(\text{tiempo con mejora})} = \frac{1}{(1 - F_m) + \frac{F_m}{A_m}}$$

Donde:
A → Aceleración o ganancia en velocidad conseguida en el sistema completo debido a la mejora de uno de sus subsistemas.
A_m → Factor de mejora que ha introducido el subsistema alterado.
F_m → Fracción de tiempo que el sistema emplea el subsistema alterado.

Hacer común el caso rápido o acelerar el caso común