Estudio del Rendimiento

Arquitecturas Avanzadas

1. Introducción

1. Introducción

El ordenador ha sido el descubrimiento más importante de la historia, ya que no ha parado de evolucionar desde su aparición. Las razones de este éxito tan espectacular han sido esencialmente dos:

- La ruptura de la barrera entre el programador y el ensamblador.
- La aparición de los sistemas abiertos (*open systems*), y en particular aquellos basados en el sistema operativo UNIX.

La tendencia actual es la siguiente:

- Invención de arquitecturas nuevas cada dos o tres años.
- Mejora de los compiladores.
- Mejora de las comunicaciones.
- La aparición de las máquinas masivamente paralelas.

Por este motivo, el rendimiento de las máquinas ha experimentado un crecimiento enorme de unos años hasta ahora. En este trabajo se hace un resumen del estudio del rendimiento de los sistemas actuales.

2. Factores que influyen en el rendimiento de un Job

El rendimiento de un job depende de los siguientes factores:

- Hardware
- Software
- Contenido del Job
- Diseño de la aplicación

2.1. Hardware

El hardware condiciona de manera muy importante en rendimiento escalar y vectorial− Este rendimiento, en general, va a depender del número de procesadores escalares y/o vectoriales y de la potencia del conjunto de instrucciones de la máquina.

2.2. Software

El software es también un factor muy importante del rendimiento de un job. Este rendimiento dependerá básicamente de la capacidad de los compiladores vectorizantes, compiladores paralelos y de la biblioteca de subrutinas de que se disponga.

El compilador es el encargado de cerrar la brecha entre el paralelismo software y hardware. Algunas tareas de paralelización las realizará el compilador (las más complejas) y otras las realizará el propio usuario (programando).

2.3 Contenido del Job

El contenido del Job también es un factor importante que influye en su rendimiento.

- La cantidad de operaciones en **coma flotante** que se realicen es importante, ya que estas son muy costosas debido a que emplean mucho más tiempo de la ALU que las operaciones en coma fija.
- El tanto por ciento de código vectorizable
- El tanto por ciento de código paralelo

2.4 Diseño de la aplicación

Es importante que se ajuste al diseño de la máquina y que se organicen los datos.

2.5 Conclusiones

Si tenemos un software que no se adapta bien al hardware, el rendimiento del sistema y en particular del job será muy pobre.

3. Rendimiento del Sistema

Para hacer un modelo de rendimiento tendríamos que tener en cuenta muchos parámetros, pero esto no es rentable. Por consiguiente nos conformamos con modelos simplificados para la medida del rendimiento de un sistema.

Para obtener un alto rendimiento del sistema es necesario que haya una sintonía entre la capacidad de la máquina y el comportamiento del programa.

La capacidad de la máquina es susceptible de mejora con las nuevas tecnologías hardware y software, además de la gestión eficiente de los recursos.

El comportamiento del programa depende básicamente de los siguiente factores:

- Diseño del algoritmo
- Estructuras de datos
- Eficiencia de los lenguajes
- Conocimientos del programador
- Tecnología de los compiladores

Las estructuras de datos proporcionan un alto grado de paralelismo y le condicionan. Así mismo, los lenguajes y compiladores son muy importantes en cuenta a eficiencia de los primeros y la inteligencia de los segundos para detectar dentro del código aquellas partes que pueden ser paralelizables. Los conocimientos del programador también son muy importantes ya que junto con el diseño del algoritmo, los desarrollos pueden adaptarse mucho mejor al hardware del sistema.

El rendimiento de un sistema varía según el programa. Esto lo podemos observar con las siguientes características intrínsecas de la relación entre el sistema y el programa:

- Imposibilidad de alcanzar un rendimiento máximo.
- Resultados de BENCHMARKING ligados a la composición del programa.

4. Indicadores del Rendimiento de un Computador

Los indicadores del rendimiento de un computador son una serie de parámetros que conforma una modelo simplificado de la medida del rendimiento de un sistema y son utilizados por los arquitectos de sistemas, los programadores y los constructores de compiladores, para la optimización del código y obtención de una ejecución más eficiente. Dentro de este modelo, estos son los indicadores de rendimiento más utilizados:

4.1 Turnaround Time

El tiempo de respuesta. Desde la entrada hasta la salida, por lo que incluye accesos a disco y memoria, compilación, sobrecargas y tiempos de CPU. Es la medida más simple del rendimiento.

En sistemas multiprogramados no nos vale la medida del rendimiento anterior, ya que la máquina comparte el tiempo, se produce solapamiento E/S del programa con tiempo de CPU de otros programas. Necesitamos otra medida como es el *TIEMPO CPU USUARIO.*

4.2 Tiempo de cada ciclo ( )

El tiempo empleado por cada ciclo. Es la constante de reloj del procesador. Medida en nanosegundos.

4.3 Frecuencia de reloj (f)

Es la inversa del tiempo de ciclo. f = 1/ . Medida en Megahertz.

4.4 Total de Instrucciones (Ic)

Es el número de instrucciones objeto a ejecutar en un programa.

4.5 Ciclos por instrucción (CPI)

Es el número de ciclos que requiere cada instrucción. Normalmente, CPI = CPI medio.

4.6 Tiempo de ejecución de programa (Tp)

Es el tiempo que tarda un programa en ejecutarse.

Tp = Ic * CPI * = Ic * CPI/f = C/f

Total de ciclos de reloj en la ejecución de un programa (C)

C = Ic * CPI

4.7 Ciclo de memoria (mc)

Es tiempo que se tarda en completar una referencia a memoria.

mc = k * klatencia >1

4.8 Componentes del CPI

A partir de las nuevas definiciones de referencias a memoria por ciclo y el total de ciclos del procesador, las fórmulas del CPI y del Tp se pueden de la siguiente forma:

- Total de ciclos del procesador (p).

- Referencias a memoria por ciclo (mr).

CPI = p + mr * k (ciclos/instrucción)

Tp = Ic * CPI * = Ic * (p + mr * k) * (nanosegundos)

4.9 Relación entre factores de rendimiento y atributos del sistema

	Ic	p	mr	k	
Arquitectura	X	x			
Tecnología compilador	X	x	x		
Implantación y control CPU		x			x
Jerarquía memoria				x	x

En este cuadro resumen, se muestra la relación entre los factores del rendimiento (Ic, p, mr, k y) y algunas características del sistema (arquitectura, tecnología del compilador, implantación y control CPU y jerarquía de la memoria caché).

4.10 Relación MIPS

Podemos utilizar un nuevo modelo del rendimiento deducido a partir del parámetro MIPS (Millones de instrucciones por segundo). Es una medida de la velocidad del ordenador, que depende de la frecuencia del reloj (f), del total de instrucciones (Ic), y de los ciclos por instrucción (CPI).

MIPS = Ic (Tp * 106) = (Ic * f) / (Ic * CPI * 106) = f / (CPI * 106)

MIPS = f / (C/Ic * 106) = (f * Ic) / (C * 106) (instrucciones/segundo)

A partir de la definición de MIPS se puede utilizar la siguiente fórmula para el tiempo de CPU:

Tiempo CPU = Tp = (Ic * 10−6)/MIPS (segundos)

4.11 THROUGHPUT del sistema (Ws)

Es la cantidad de trabajo por unidad de tiempo que realiza el sistema. Total de programas (resultados) ejecutados por el sistema en unidad de tiempo.

Ws (programas/segundo)

4.12 THROUGHPUT de UCP (Wp)

Es la cantidad de trabajo de la UCP.

Wp = f / (Ic * CPI) = (MIPS * CPI * 106)/(Ic * CPI) = (MIPS * 106)/Ic (programas/segundo)

5. Benchmark

Un *Benchmark* es una prueba que mide el rendimiento de un sistema o subsistema en una tarea o conjunto de tareas bien definidas.

Los *Benchmarks* se utilizan normalmente para predecir el rendimiento de un sistema desconocido en una tarea o carga de trabajo bien definida o conocida. Pueden ser también utilizados como herramientas de monitorización y diagnóstico.

Al realizar un *benchmark* y comparar los resultados con una configuración conocida, potencialmente se puede precisar la causa de un rendimiento pobre. Similarmente, un desarrollador (programador) puede realizar un *benchmark* después de realizar un cambio que pueda impactar en el rendimiento para determinar la magnitud del mismo.

Los *benchmarks* son normalmente usados para asegurar un nivel mínimo de rendimiento en la especificación procurada. Aunque el rendimiento es uno de los factores más importante en una compra, no nos podemos olvidar que es más importante ser capaz de realizar un job correctamente, que no obtener una respuesta errónea en la mitad de tiempo.

5.1 Ejemplo

MAQUINA	RELOJ=f	RENDIMIENTO	TIEMPO UCP = Tp
VAX 11/780	5 MHz	1 MIPS	12x seg
IBM RS/6000	25 MHz	18 MIPS	x seg

- TIEMPO UCP EN VAX: 12 veces superior

- CODIGOS OBJETO TIENEN DIFERENTES LONGITUDES:
- DIFERENTES ARQUITECTURAS
- DIFERENTES COMPILADORES

- Ic = MIPS * Tp * 106

VAX Ic = 1*12x*106

IBM Ic = 18*x*106

VAX/IBM = 12/18 IBM = VAX 18/12 = 1.5 VAX

TOTAL INSTRUCCIONES CODIGO OBJETO PROCESADO EN RS/6000 1.5 VECES SUPERIOR AL PROCESADO EN VAX

- CPI = f / (MIPS * 106)

VAX CPI = 5 / (1*106) = 5 *10−6

IBM CPI = 25 / (18*106) = 1.39 *10−6

CONSIDERANDO UN CPI DE 5 PARA VAX. CPI DE RS/6000 ES SOLO DE 1.39

- VAX ES CISC − COMPLEX INSTRUCTION SET COMPUTING
- SI PROCESO OTRO PROGRAMA EL BENCHMARK PUEDE SER DISTINTO

Apéndice A. Definición de Unidades de las Fórmulas

- **(tiempo de ciclo de reloj):** Unidad de tiempo, normalmente nanosegundos/ciclo.
- **f** (**frecuencia de reloj**)**: 1/ ,** Unidad de frecuencia, normalmente MegaHercios/ciclo

- **Ic (Instruction count):** Número de instrucciones objeto a ejecutar. Unidad : instrucciones.
- **CPI (Ciclos por instrucción):** Número de ciclos por instrucción (ordenadas por familia de instrucciones). Unidad : ciclos/instrucción.

- **C :** Total de ciclos de reloj de ejecución de un programa. Unidad : ciclos.

- **Tp (Tiempo CPU de ejecución de un programa):** Unidad tiempo, normalmente nanosegundos:

- **mc (ciclo de memoria): K (K>1), siendo K=latencia.** Unidad de tiempo, normalmente nanosegundos/ciclo.
- **MIPS (Millones de instrucciones por segundo):** Medida instrucciones/segundo.

- **Ws (throughtput del sistema):** Medida en programas/segundo.
- **Wp (throughtput de la CPU):** Medida en programas/segundo.

Apéndice B. Ejercicios

Problema 1

Un computador A tiene una frecuencia de reloj de 80 MHz. Ejecuta un programa en 15 segundos. Se quiere diseñar otro computador B, para que ejecute el mismo programa en 8 segundos.

Existe la posibilidad de incrementar la frecuencia de reloj de A, haciendo que B emplee 1.5 veces el total de ciclos de reloj de A, para el mismo programa.

Se pide calcular la frecuencia de reloj del computador B.

TpA = 15 segundos. TpB = 8 segundos

fA = 80 MHz. fB?

CB = 1.5 CA

Tp = C/f

TpA = CA/fA

CA = TpA*fA = 15 * 80 = 1200 ciclos

TpB = CB/fB

Resultado:

fB = CB/TpB = 1.5 CA / TpB = 1.5 * 1200 / 8 = **225 MHz**

Problema 2

Tenemos dos arquitecturas A y B con un mismo juego de instrucciones. A tiene un ciclo de reloj de 12 nseg y un CPI de 1.5 ciclos para un determinado programa. B tiene un ciclo de reloj de 7 nseg y un CPI de 3 ciclos para el mismo programa.

Se pide calcular que arquitectura es más rápida y en cuanto.

Arquitectura A:

A = 12 nseg.

CPIA = 1.5 ciclos.

CA = IcA * CPIA = 1.5 IcA

Arquitectura B:

B = 7 nseg.

CPIB = 3 ciclos.

CB = IcB * CPIB = 3 IcB

TpA = CA *

TpA = 1.5 IcA * 12 = 18 IcA nseg.

TpB = CB *

TpB = 3 IcB * 7 = 21 IcB nseg.

TpB > TpA, por tanto RA > RB

Conclusión:

LA ARQUITECTURA A ES UN 30% MÁS RÁPIDA QUE LA B

Problema 3

Tenemos dos maquina A y B. El reloj de la maquina A es de 50 MHz. Su rendimiento es igual a 100 MIPS y su tiempo de CPU 20x seg. El reloj de B es igual a 100 MHz, su rendimiento es de 130 MIPS y su tiempo CPU igual a 15x seg.

Se pide calcular Ic de A y B, y la relación entre ellos.

Se pide calcular el CPI de A y B, y su relación.

Ic = MIPS * Tp * 106

IcA = 100 * 20x * 106 = 200x * 107 instrucciones.

IcB = 130 * 15x * 106 = 195x * 107 instrucciones.

IcA = 200*x*107 instrucciones

IcB = 195*x*107 instrucciones

A = 1.02 B

Conclusión:

El número de instrucciones procesadas por A es 1.02 veces superior al número de instrucciones procesadas por B.

CPI = Tp * f / Ic

CPIA = 20x * 50 / 200*x*107 = 0.5*10−7

CPIB = 15x * 100 / 195*x*107 = 7.7 * 10−7

Conclusión:

Considerando un CPI de 7.7 para B, el CPI A ES de 0.5

1

2

Tp = Ic*CPI* (tiempo)

Tp = (Ic*CPI)/f (tiempo)

Tp = C/f (tiempo)

C = Ic*CPI (ciclos)

CPI = p + mr*k (ciclos/instrucción)

MIPS = Ic (Tp * 106) = (Ic * f) / (Ic * CPI * 106) = f / (CPI * 106)

MIPS = f / (C/Ic * 106) = (f * Ic) / (C * 106) (instrucciones/segundo)

Wp = f / (Ic * CPI) = (MIPS * CPI * 106)/(Ic * CPI)

Wp = (MIPS * 106)/Ic (programas/segundo)