Caracterizaciones simples del rendimiento de una máquina

1. Introducción

A continuación analizaremos los principales factores que determinan las prestaciones de un ordenador. Hay que tener en cuenta que la métrica utilizada para determinar el rendimiento puede depender de la aplicación particular. Por tanto, no existe una medida única que determine la prestación de un procesador determinado.

El rendimiento de un ordenador es un aspecto crítico cuando se trata de comparar máquinas diferentes: nos permite entender porqué un trozo de software se comporta de una forma determinada, porqué un juego de instrucciones es mejor que otro, o cómo de bien se comportan la memoria y la E/S.

2. Rendimiento y Productividad

Cuando decimos que un ordenador es más rápido que otro, ¿qué queremos decir?:

1) Que puede ejecutar un programa en menos tiempo
2) Que puede completar más tareas que otro en la misma unidad de tiempo

Ambas consideraciones son válidas. En la primera aproximación es importante el **TIEMPO DE RESPUESTA**: tiempo transcurrido entre el comienzo y el fin de un evento. También se le llama **TIEMPO DE EJECUCIÓN** o **TIEMPO DE LATENCIA**.

En la segunda aproximación es importante la **PRODUCTIVIDAD**: cantidad de trabajo realizado en un tiempo determinado. También se le llama **ANCHO DE BANDA**.

Normalmente el *tiempo de respuesta, tiempo de ejecución* y la *productividad* son términos que se usamos cuando nos referimos a una tarea de cálculo concreta. Los términos *tiempo de latencia* y *ancho de banda* los usamos al hablar de los sistemas de memoria.

El concepto más simple de entender y abordar es el de *tiempo de ejecución*. ***Para maximizar el rendimiento queremos minimizar el tiempo de ejecución de alguna***

tarea. Por lo tanto, el rendimiento de un ordenador X se calcula con la siguiente fórmula:

$$Rendimiento_X = \frac{1}{Tiempo\ de\ ejecucion_X}$$

Si comparamos dos ordenadores X e Y, y el *rendimiento* de X es mayor que el de Y, significa que el *tiempo de ejecución* de Y es mayor que el de X:

$$Rendimiento_X > Rendimiento_Y$$

$$\frac{1}{Tiempo\ de\ ejecucion_X} > \frac{1}{Tiempo\ de\ ejecucion_Y}$$

$$Tiempo\ de\ ejecucion_X < Tiempo\ de\ ejecucion_Y$$

Al comparar el rendimiento de dos ordenadores de forma cuantitativa, si decimos que **X es n veces más rápida que Y**:

$$\frac{Rendimiento_X}{Rendimiento_Y} = n$$

es que el tiempo de ejecución de Y es n veces mayor que el de X:

$$\frac{Tiempo\ de\ ejecucion_Y}{Tiempo\ de\ ejecucion_X} = n$$

Si se dice que el ordenador **X es un n% más rápido que Y:**

$$\frac{\mathrm{Re}\,n\,\mathrm{dim}\,iento_X}{\mathrm{Re}\,n\,\mathrm{dim}\,iento_Y} = \frac{Tiempo\_ejecución_Y}{Tiempo\_ejecución_X} = 1 + \frac{n}{100}$$

Incrementar rendimiento implica disminuir el tiempo de ejecución. Se suele utilizar la expresión “mejorar el tiempo de ejecución” o “mejorar el rendimiento” para indicar “disminuir el tiempo de ejecución”.

3. Rendimiento de la CPU

3.1 Definiciones previas

Antes de analizar el rendimiento de un procesador, debemos conocer las siguientes definiciones:

> ***Tiempo transcurrido***: Tiempo total necesario para completar una tarea, incluyendo accesos a discos, a memoria, E/S, etc...
>
> ***Tiempo de CPU***: Tiempo que la CPU emplea en una determinada tarea. No incluye el tiempo de E/S ni de atención a otros programas.
>
> ***Tiempo de CPU de usuario***: Tiempo que la CPU pasa en el programa del usuario.
>
> ***Tiempo de CPU de sistema***: Tiempo que la CPU pasa ejecutando tareas de sistema operativo en beneficio del programa del usuario.
>
> ***Rendimiento del sistema***: En este caso utilizamos como base el tiempo transcurrido en un sistema sin cargas (monotarea).
>
> ***Rendimiento de CPU***: En este caso se utiliza como base el tiempo de CPU.

Aunque como usuarios de ordenador nos preocupa el tiempo, los diseñadores de procesadores están más interesados en métricas que permitan describir cuanto de rápido puede el hardware ejecutar una función determinada.

Los procesadores se construyen utilizando un reloj a una frecuencia fija que determina cuando pueden tener lugar los eventos en el hardware. Estos intervalos discretos se denominan **ciclos de reloj**.

> ***Periodo de reloj***: Longitud de un periodo completo de reloj (en ns. por ejemplo).
>
> **Frecuencia de reloj:** Inversa del periodo de reloj (en GHz por ejemplo).

3.2 Rendimiento de la CPU

Las métricas utilizadas por usuarios y diseñadores suelen ser diferentes. Si podemos relacionar unas métricas con otras, podemos determinar el efecto de un cambio en el diseño como lo percibe otro usuario. Las dos métricas más básicas son el **periodo de reloj** y el **número de ciclos de reloj**. Una fórmula simple nos permite relacionar ambas métricas con el tiempo de CPU:

$$Tiempo_{CPU\ programa} = NúmeroCiclos\,\mathrm{Re}\,loj_{CPU\ programa} \times Periodo\,\mathrm{Re}\,loj$$

Puesto que el período y la frecuencia del reloj son inversos, también podemos escribir:

$$Tiempo_{CPU\ programa} = \frac{NúmeroCiclos\,\mathrm{Re}loj_{CPU\ programa}}{Frecuencia\,\mathrm{Re}loj}$$

La fórmula anterior nos da una interesante perspectiva. El diseñador de hardware puede mejorar el rendimiento de dos formas diferentes:

1. Reduciendo el periodo de reloj
2. Reduciendo el número de ciclos de reloj necesitados por el programa.

Este es un problema de compromiso. Muchas técnicas que reducen el número de ciclos de reloj también aumentan el periodo de reloj.

3.3 Rendimiento de la CPU considerando las instrucciones de los programas

Las ecuaciones que hemos visto hasta ahora no incluyen el número de instrucciones que necesita el programa. Puesto que el compilador genera instrucciones que el ordenador tiene que ejecutar para correr el programa, el tiempo de ejecución debe depender del número de instrucciones que tenga el programa. Por tanto, el número de ciclos de reloj necesitados por un programa se puede escribir como:

$$NúmeroCiclos\,\mathrm{Re}loj_{CPU\ programa} = NúmeroInstrucciones_{programa} \times \mathrm{Pr}omedioCiclos\,\mathrm{Re}loj_{instrucción}$$

El tercer término representa el **número de ciclos de reloj que tarda en promedio una instrucción** (ya que no todas las instrucciones necesitan los mismos ciclos de reloj). Se suele abreviar como **CPI**. La ventaja fundamental es que nos permite comparar dos implementaciones diferentes de la misma arquitectura del juego de instrucciones, puesto que el número de instrucciones para el mismo programa será el mismo.

Como antes establecimos:

$$Tiempo_{CPU\ programa} = NúmeroCiclos\,\mathrm{Re}loj_{CPU\ programa} \times Periodo\,\mathrm{Re}loj$$

Por lo tanto:

$$Tiempo_{CPU} = NumeroInstrucciones * CPI * Periodo_{reloj}$$

$$\frac{Segundos}{Programa} = \frac{Instrucciones}{programa} x \frac{CiclosReloj}{Instrucción} x \frac{Segundos}{CicloReloj}$$

Y también podemos escribir:

$$Tiempo_{CPU} = \frac{NumeroInstrucciones * CPI}{Frecuencia_{reloj}}$$

Estas fórmulas son bastante útiles porque separan los tres conceptos clave que afectan al rendimiento. Podemos utilizarlas para comparar dos implementaciones diferentes o para evaluar un diseño alternativo:

- El ciclo de reloj se conoce porque es un dato que forma parte de la documentación del ordenador.
- El tiempo de ejecución se puede medir ejecutando el programa.
- El número de instrucciones se puede medir observando la salida del compilador.

Una vez conocidos esos datos podemos determinar el CPI. Estos parámetros dependen no sólo del juego de instrucciones sino también del algoritmo, el lenguaje de programación y el compilador.

Componente	Parámetro	Cómo afecta
Algoritmo	Número de instrucciones, CPI	El algoritmo determina el número de instrucciones fuente, y por tanto el número de instrucciones del procesador. El algoritmo también puede afectar al CPI, debido a que puede favorecer instrucciones más lentas o rápidas.
Lenguaje de programación	Número de instrucciones, CPI	El lenguaje de programación traduce las instrucciones fuente a instrucciones de procesador. Por tanto, afecta tanto al número de instrucciones generadas como a su tipo (puede generar instrucciones más largas o cortas)
Compilador	Número de instrucciones, CPI	El compilador es quien efectúa la traducción de las instrucciones a lenguaje máquina. Los motivos son idénticos a los anteriores.
Arquitectura del juego de instrucciones	Número de instrucciones, Periodo de reloj, CPI	El juego de instrucciones afecta a los tres parámetros.

No podemos fijarnos en uno sólo de estos tres componentes (número de instrucciones, CPI, ciclos de reloj) ya que los tres afectan al tiempo de ejecución. Si alguno de los tres factores es idéntico, podemos comparar los que difieren. Por ejemplo en dos máquinas con el mismo ciclo de reloj podemos comparar tanto el número de instrucciones como el CPI.

4. Ley de Amdahl

4.1 Acelerar el caso común

Los computadores se diseñan en base a un principio importante y generalizado: *acelerar el caso común.* Al realizar un diseño se debe favorecer el caso más frecuente sobre el infrecuente. Esto mejorará el rendimiento. Además, el caso frecuente suele ser más simple y se puede realizar más rápido que el caso infrecuente.

Por ejemplo, cuando sumamos dos números en la CPU podemos esperar que el desbordamiento (overflow) sea una circunstancia infrecuente. Por tanto, podemos mejorar el rendimiento optimizando el diseño de los circuitos para satisfacer el caso más frecuente, que es el de realizar sumas con ausencia de desbordamiento. Este hecho ralentizará la situación en que se presente un desbordamiento, pero si este caso es infrecuente, el rendimiento global mejorará al optimizar el caso más común.

4.2 Ley de Amdahl

El aumento del rendimiento que puede obtenerse al mejorar alguna parte de un computador puede calcularse usando la **LEY DE AMDAHL**. Esta ley establece que:

"La mejora obtenida en el rendimiento al utilizar algún modo de ejecución más rápido está limitada por la fracción de tiempo en que se pueda utilizar ese modo más rápido"

Esta ley define la **GANANCIA DEL RENDIMIENTO** o **ACELERACIÓN DEL RENDIMIENTO** que puede lograrse al usar una característica particular. Se puede calcular de dos modos:

1) Dividiendo el rendimiento de la tarea completa usando la mejora cuando sea posible entre el rendimiento de la tarea completa sin usar la mejora.
2) Dividiendo el tiempo de ejecución de la tarea completa sin usar la mejora entre el tiempo de ejecución de la tarea usando la mejora cuando sea posible.

La aceleración indica la rapidez con la que se realiza una tarea utilizando una máquina con la mejora respecto a cuando se usa la máquina original. La aceleración depende de dos factores:

1) Fracción de tiempo de cálculo de la máquina original que puede usarse para aprovechar la mejora: **FRACCIÓN MEJORADA (FM)**. Esta magnitud es igual o menor que 1.
2) Optimización lograda por el modo de ejecución mejorado: cuánto más rápido se ejecutaría la tarea si solamente se usara el modo mejorado: **ACELERACIÓN MEJORADA (AM).** Esta magnitud es mayor que 1.

El tiempo de ejecución (TE) que tarda una máquina en hacer una determinada tarea cuando se le añade una mejora será el tiempo de ejecución que tarda la máquina

mientras puede usar esa mejora más el tiempo de ejecución que tarda la máquina cuando no puede hacer uso de esa mejora.

$$TE_{nuevo} = TE_{antiguo} \times \left((1 - FM) + \frac{FM}{AM} \right)$$

La **ACELERACIÓN GLOBAL (AG)** es la relación de los tiempos de ejecución (TE) nuevo y antiguo:

$$AG = \frac{TE_{antiguo}}{TE_{nuevo}} = \frac{1}{\left((1 - FM) + \frac{FM}{AM} \right)}$$

La ley de Amdahl expresa la **LEY DE LOS RENDIMIENTOS DECRECIENTES**:

"La mejora incremental en la aceleración conseguida por una mejora adicional en el rendimiento de una parte del cálculo disminuye a medida que se van añadiendo mejoras"

La ley de Amdahl es una guía para evaluar el aumento del rendimiento de un ordenador al introducir en él una mejora y para saber cómo distribuir los recursos del ordenador para mejorar la relación coste/rendimiento.

5. Otros modos de medir el rendimiento

5.1 MIPS

MIPS es el acrónimo de "millones de instrucciones por segundo". Es una forma alternativa de medir el rendimiento de los procesadores.. Para un programa dado, los MIPS son:

$$MIPS = \frac{Número\ de\ Instrucciones}{Tiempo\ de\ Ejecución * 10^6}$$

Sin embargo, esta medida sólo es útil para comparar procesadores con el mismo juego de instrucciones y usando programas que fueron compilados por el mismo compilador y con el mismo nivel de optimización. Esto es debido a que la misma tarea puede necesitar un número de instrucciones diferentes si los juegos de instrucciones también lo son; y por motivos similares en las otras dos situaciones descritas.

5.2 MFLOPS

MFLOPS es el acrónimo de "millones de instrucciones de punto flotante por segundo" (*floating point operations per second*). En informática, las operaciones de punto flotante por segundo son una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de coma flotante.

Para aplicaciones ordinarias (no científicas) las operaciones sobre enteros (medidos en MIPS) son mucho más comunes. De lo anterior se deduce que medir el rendimiento en FLOPS no predice con precisión la rapidez con la que un procesador realizará cualquier tarea. Sin embargo, para muchas aplicaciones científicas, como el análisis de datos, el rendimiento en FLOPS es una medida efectiva.

5.3 Benchmark

El **benchmark** (=*comparativa*) es una técnica utilizada para medir el rendimiento de los computadores. Un benchmark es el resultado de la ejecución de un programa informático o un conjunto de programas en una máquina, con el objetivo de estimar el rendimiento de un elemento concreto, y poder comparar los resultados con máquinas similares. Un benchmark podría ser realizado para cualquier componente del ordenador, ya sea CPU, RAM, tarjeta gráfica, etc.

La tarea de ejecutar un benchmark originalmente se reducía a estimar el tiempo de proceso que lleva la ejecución de un programa (medida por lo general en miles o millones de operaciones por segundo – MIPS). Con el paso del tiempo, la mejora en los compiladores y la gran variedad de arquitecturas y situaciones existentes convirtieron a esta técnica en toda una especialidad. La elección de las condiciones bajo la cual dos sistemas distintos pueden compararse entre sí es especialmente ardua.

Para llevar a cabo un benchmark se pueden usar **programas sintéticos**. Estos programas están especialmente diseñados para medir el rendimiento de un componente individual de un ordenador, normalmente llevando el componente escogido a su máxima capacidad. O también se pueden usar herramientas basadas en aplicaciones reales, que simulan una carga de trabajo para medir el comportamiento global del equipo.

Las técnicas benchmark pueden evaluar el rendimiento del computador en distintos niveles:

- Tests de Bajo nivel: miden directamente el rendimiento de los componentes. Ejemplo: el reloj de la CPU, los tiempos de la DRAM y de la caché SRAM, tiempo de acceso medio al disco duro, latencia, tiempo de cambio de pista, etc.

- Test de Alto nivel: están más enfocados a medir el rendimiento de la combinación componente/controlador/SO de un aspecto específico del sistema.

SPCE (System Performance Evaluation Corporation) es un intento de un gran número de proveedores para crear un conjunto estándar de benchmarks para ordenadores

modernos, que comenzó en 1989. Hoy en día ofrece una docena de benchmarks diferentes, incluyendo las prestaciones de:

- CPU
- Gráficos
- Cálculo intensivo
- Cálculo orientado a objetos
- Aplicaciones Java
- Modelos cliente servidor
- Sistemas de fichero
- Sistemas de correo
- Servidores Web

La última versión es SPEC CPU 2006, que consiste en 14 programas de punto flotante y 12 programas de enteros. Intenta medir el rendimiento de la CPU y también ofrece el tiempo de ejecución total. Ofrece un resumen separado para los programas de enteros que para los de punto flotante.

Benchmark	Language	Application Area
400.perlbench	C	Lenguaje prog.
401.bzip2	C	Compresión
403.gcc	C	Compilador
429.mcf	C	Optimización combinatoria
445.gobmk	C	Intelig. Artificial
456.hmmer	C	Busca secuencia genes
458.sjeng	C	Juego Ajedrez
462.libquantum	C	Cálculo física cuántica
464.h264ref	C	Compresión de vídeo
471.omnetpp	C++	Simulación de eventos discretos
473.astar	C++	Algoritmos de búsqueda de trayectorias
483.xalacbmk	C++	Procesamiento XML

Cuadro 2.1: Benchmarks SPEC CPU2006 de punto fijo

Benchmark	Lenguaje	Application area
410.bwaves	Fortran	Dinámica de fluidos
416.gamess	Fortran	Química cuántica
433.milc	C	Cromodinámica cuántica (Física)
434.zeusmp	Fortran	CFD (Física)
435.gromacs	C, Fortran	Dinámica molecular (Bioquímica)
436.cactusADM	C, Fortran	Relatividad general (Física)
437.leslie3d	Fortran	Dinámica de fluidos
444.namd	C++	Dinámica molecular (Biología)
447.dealII	C++	Análisis de elementos finitos
450.soplex	C++	Optimización, programación lineal
453.povray	C++	Imagen ray-tracing
454.calculix	C, Fortran	Mecánica estructural
459.GemsFDTD	Fortran	Electromagnetismo computacional
465.tonto	Fortran	Química cuántica

Cuadro 2.2: Benchmarks SPEC CPU2006 de punto flotante