Arquitectura y Organización de Computadoras II

Rendimiento

MSc. Ing. Ticiano J. Torres Peralta
Ing. Pablo Toledo

2023 – Segundo Cuatrimestre

Universidad Nacional de Tucumán
Facultad de Ciencias Exactas y Tecnología

Ecuación Clásica de Rendimiento

$$CPU\ time = Instruction\ Count\ x\ CPI\ x\ Clock\ Period$$

$$CPU\ time = \frac{Instruction\ Count\ x\ CPI}{Clock\ Rate}$$

Parametro	Significado	UoM
CPU time	Tiempo de ejecución en CPU	Segundos
Instruction Count	Total instructiones ejecutadas	Instrucciones
CPI (Cycles Per Instruction)	Promedio de ciclos que toma una instruccion en completarse	Ciclos / Instrucciones
Clock Period Clock Rate o Frecuency	Tiempo para completar un ciclo de reloj Frecuencia del reloj	Segundos (Hoy típicamente en 10E-9, osea, ns) Hz (Hoy típicamente en 10E9, osea, GHz)

Ecuación Clásica de Rendimiento

Es importantes notar que hay parámetros que son mas fácil de medir que otros. Podemos medir el tiempo de ejecución ejecutando nuestro programa, y la frecuencia de reloj es publicada como parte de la documentación del CPU. El total de instrucciones puede ser medido con herramientas de software que perfilan la ejecución de un programa. Esto usan contadores (registros contadores) que se encuentran en el CPU y se pueden utilizar para medir una serie de métricas. El CPI, sin embargo, depende de muchos detales como el sistema de memoria, la estructura y organización del procesador, como también la mezcla de los tipos de instrucciones que se encuentran ejecutando.

En fin, el rendimiento de una maquina termina dependiendo del: algoritmo, el lenguaje de programación, el compilador, la arquitectura y la implementación del hardware (microarquitectura).

Factores Que Afectan Rendimiento

Componente	Afecta que?	Como?
Algoritmo	Total Instrucciones, CPI	Determina la cantidad de instrucciones del programa fuente y por ende la cantidad de instrucciones ejecutadas a nivel procesador. Tambien puede afectar el CPI, favoreciendo instrucciones mas lenta o mas rápidas Por ejemplo, si el algoritmo mas divisiones, tendrá un CPI mas alto.
Lenguaje de Programación	Total Instrucciones, CPI	Afecta el total de instrucciones porque las frases escritas en lenguaje de alto nivel tienen que ser traducidas a su equivalente a nivel procesador. Puede afectar CPI porque un lenguaje, por ejemplo, que usa mucho la abstracción de datos (C++, Java) usaran mucho el llamado indirecto, que incrementa el CPI.
Compilador	Total Instrucciones, CPI	La eficiencia del compilador afecta tanto el total de instrucciones como el CPI, dado que el mismo gobierna como se efectuá la traducción desde un lenguaje de alto nivel al del procesador. El rol del compilador puede ser muy complexo y también su efecto en el CPI.
ISA	Total Instrucciones, CPI, Frecuencia de Reloj	Afecta los tres aspectos del rendimiento del CPU, dado que define las instrucciones disponibles, la complejidad del la instrucción y por ende la limitaciones en la frecuencia de reloj.

A tener en cuenta

Parece lógico pensar, o esperar, que el CPI mínimo posible es 1, pero veremos que algunos procesadores (la mayoría de hoy) buscan y ejecutan multiples instrucciones por ciclo de reloj. Para reflejar esto, algunos diseñadores invierten la medida de CPI para utilizar IPC (Instructions Per Clock/Cycle). Por ejemplo, si un procesador ejecuta en promedio 2 instrucciones por ciclo de reloj, entonces tiene un IPC de 2 y por consecuencia un CPI de 0.5. (Aun mas importante, tendencias de industria causan que el IPC/CPI ya no sea medido en termino de la ejecución completa de la instrucción, sino hasta que la instrucción es despachada a la unidad de ejecución.)

Tambien a considerar, tradicionalmente el ciclo de reloj era una medida fija, pero para ser mas eficientes o aumentar el rendimiento, CPUs modernos pueden variar la velocidad del ciclo. Uno tienen que usar el promedio de ciclo de reloj para un programa. Por ejemplo, Inter Cores CPUs tienen una tecnología llamada Turbo Boost Technology.

MIPS

MIPS (Millions Instructions Per Second) es una medida de taza de ejecución, computadoras mas rápidas tienen un MIPS mas alto. Es una medida fácil de entender, un MIPS mas grande, una computadora mas rápida.

$$MIPS = \frac{Instruction\ Count}{Execution\ Time\ x\ 10^6}$$

Pero hay problemas obvios en utilizar esta métrica. MIPS especifica una taza de ejecución de instrucciones pero no tiene información sobre la complejidad de esas instrucciones. Esto significa que no puede ser utilizada cuando comparando dos ISA distintas. En segundo lugar, el MIPS varia entre programas en la misma computadora, entonces una computadora teóricamente no tiene una sola métrica de MIPS.

Trampas en Conceptos

Trampa: Usando un subconjunto de la ecuación clásica de rendimiento como una métrica de rendimiento.

El claro que no se puede predecir el rendimiento basado en simplemente la frecuencia de reloj, el total de instrucciones, or el CPI. Otro error común es utilizar solo dos de los tres parámetros. Por ejemplo, comparando dos algoritmos que hacen la misma cosa con solo el total de instrucciones y la velocidad de reloj, cuando uno puede tener un CPI sustancialmente diferente al otro.

Una alternativa que vimos a la ecuación clásica de rendimiento es la MIPS.

Ley de Amdahl

Uno puede obtener mejoras en rendimiento si decide mejorar una porción de la computadora. La mejora global puede ser calculada usando la Ley de Amdahl. La misma dice:

La mejora en rendimiento utilizando algún modo mas rápido de ejecución esta limitada por la fracción de tiempo (del total) donde se aplica ese modo mas rápido.

Ley de Amdahl

$$Speedup = \frac{Performance\ for\ entire\ task\ using\ the\ enhancement\ when\ possible}{Performance\ for\ entire\ task\ without\ using\ the\ enhancement}$$

alternativamente,

$$Speedup = \frac{Execution\ time\ for\ entire\ task\ without\ using\ the\ enhancement}{Execution\ time\ for\ entire\ task\ using\ the\ enhancement\ when\ possible}$$

Ley de Amdahl

La mejora de tiempo de ejecución nos dice cuando mas rápido una tarea va a ejecutarse en términos relativos. Depende de dos factores:

1. La fracción del tiempo de ejecución original en la computadora original que puede ser convertido para tomar ventaja de la mejora. Por ejemplo, si 50 segundos de un programa que corre por 100 segundos puede ser mejorado, la fracción es 50/100. Este valor siempre es menor que 1.
2. El factor de mejora que se puede aplicar a esta fracción mejorada. Por ejemplo, si el tiempo de ejecución original de la fracción es 50 y el nuevo tiempo de ejecución es 25, entonces el factor de mejora es de 50/25. Este valor siempre es major que 1.

Ley de Amdahl

$$ExecutionTime_{new} = ExecutionTime_{old}\, x\,(1 - Fraction) + ExecutionTime_{old}\, x\,(\frac{Fraction_{enhancement}}{Speedup_{enhancement}})$$

$$ExecutionTime_{new} = ExecutionTime_{old}\, x\,((1 - Fraction) + (\frac{Fraction_{enhancement}}{Speedup_{enhancement}}))$$

$$Speedup_{overall} = \frac{Execution\,time_{old}}{Execution\,time_{new}} = \frac{1}{(1 - Fraction_{enhancement}) + \frac{Fraction_{enhancement}}{Speedup_{enhancement}}}$$

Ley de Amdahl

La ley de Amdahl respeta la Ley de Retornos Disminuidos. La mejora global de rendimiento obtenida por mejorar una porción de la tarea de computo disminuye mientras mas se mejora la porción en cuestión.

Como un corolario importante de esta ley, si una mejora solo se puede aplicar a una fracción de la tarea completa, entonces no se puede obtener una mejora global mas que el reciproco de 1 menos esta fracción.

Trampas

Trampa: Esperando que una mejora de un aspecto en particular de una computadora tenga un impacto de forma proporcional en la mejora global del sistema.

Esto puede ser ilustrado con la Ley de Amdahl. Supongamos que tenemos un programa que toma 100 segundos en ejecutar, con operaciones de multiplicación tomando 80 segundos del tiempo total. Cuanta mejora global se obtiene si se aplica un factor de mejora de 5 a las multiplicaciones.

$$Speedup_{overall} = \frac{1}{(1-0.8) + \frac{0.8}{5}} = 2.78$$

Trampas

Cuanto tengo que mejorar el factor de mejora de multiplicaciones si quiero que mi programa corra 5 veces mas rápido (una mejora global de 5)?

$$20 = 100\,x\left((1-0.8)+\left(\frac{0.8}{Speedup_{enhancement}}\right)\right)$$

$$20 = 20 + \frac{80}{Speedup_{enhancement}}$$

$$0 = \frac{80}{Speedup_{enhancement}}$$

No hay factor de mejora a las multiplicaciones que me provee una mejora global de 5.

Benchmarking

Para definir, hacer un benchmark es el acto de ejecutar un programa, un set de programas, u otras operaciones en la computadora, con el propósito de medir el rendimiento relativo entre sistemas completos o entre características especificas del sistemas (por ejemplo, su habilidad de procesar puntos flotantes).

Benchmarking

La mejor manera de evaluar el rendimiento es en las manos de usuarios quienes de forma rutinaria corren los mismos programas; para evaluar un sistema nuevo, simplemente compararías el tiempo de ejecución de su carga de trabajo entre ese y el anterior. Desafortunadamente, lo mencionado no es algo que se pueda hacer de forma practica.

En consecuencia, al final un usuario va a tener que confiar en otros métodos y/o evaluadores.

Benchmarking

Como ya fue mencionado, la mejor manera de medir el rendimiento es usando programas reales. Usar programas que son mas simples que alguno real puede producir malos resultados. Ejemplos de estos pueden incluir:

- Kernels - pequeña, partes claves de un programa real.
- Programas juguetes - programas de 100 líneas de código típicos cuando uno esta aprendiendo a programar (e.g. quicksort).
- Benchmarks sintéticos - programas falsos generados para tratar de simular el perfil y comportamiento de un programa real.

Hoy, en general esta puesto bastante en duda el uso de estos tres tipos de benchmarks pero desafortunadamente su uso y reportaje de los resultados siguen siendo difundidos de forma amplia.

Benchmarking

Una razón muy importante de ponerlos en duda es que escritores de compiladores y arquitectos de computadoras pueden conspirar para hacer que la computadora aparenta ser mas rápida en estos programas falsos. Esto implica también que es importante tener en cuenta las condiciones bajo la cuales uno hace un benchmark.

Por ejemplo, una forma de mejorar el rendimiento de un benchmark es compilar las aplicaciones con opciones específicamente para que rinda para ese benchmark. Para evitar esto y mejorar la estadística de los resultados (menos sesgada), los que desarrollan suites de benchmarks requieren que los vendedores usen un set especifico de opciones para cada compilador para cada lenguaje de programación.

Benchmarking

Otra pregunta a preguntar es si modificaciones del código fuente están permitidas. Hay esencialmente tres formas de tratar este tema:

- No se permiten modificaciones del código fuente.
- Modificaciones son permitidas pero son esencialmente imposibles. Por ejemplo, benchmarks para bases de datos dependen de estándares muy meticulosamente definidos que terminan como decenas de millones de líneas de código. Por consecuencia, es muy poco probable que lo tuneen para algún hardware especifico.
- Modificaciones son permitidas mientras la version modificada produzca la misma salida.

Un diseñador de benchmarks tienen la difícil decisión de ver que permitir, y si esa posible modificación reflejaría practicas reales que proveen buena información al usuario o si simplemente ofuscan la realidad.

Benchmarking

Buenas practicas cuando benchmarking es usar un benchmark suite, una collection de programas de diversos usos y características. La clara ventaja de hacer esto es que un sistema que tienen debilidad por algún lado puede compensar por otro. Al final del día, el objetivo del benchmark es dar una buena caracterización del rendimiento relativo entre dos sistemas, en especial una que se pueda extender a los programas no presentes en el suite.

Reportando Benchmarking

El principio principal en el momento de informar métricas de rendimiento debe ser la reproductibilidad: anotar todo lo que otro experimentador necesitaría para duplicar los resultados. Esto podría incluir una descripción extensa de la computadora y las opciones del compilador, así como la publicación de los resultados de referencia y optimizados. También puede incluir información de auditoría y costos; y lo más importante, los tiempos de ejecución.

Estos informes son una excelente fuente para encontrar el costo real de los sistemas informáticos, ya que los fabricantes compiten con respecto a rendimiento y costo-rendimiento.

SPEC (Standard Performance Evaluation Corporation)

SPEC creó originalmente un conjunto benchmarks centrado en el rendimiento del procesador, el SPEC89, y ha evolucionado hasta constar de un conjunto de 12 benchmarks de números enteros y 17 benchmarks de punto flotante. La suite se compone de programas reales modificados para ser portátiles y minimizar los efectos de la E/S. El siguientes diagramas muestran los programas utilizados y cómo evolucionaron con el tiempo.

SPEC

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			

SPEC

SPEC2006 benchmark description	Benchmark name by SPEC generation				
	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
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			

SPEC

Hoy, SPEC admite muchas suites para diferentes contextos informáticos:

- SPEC CPU: suite útil para realizar benchmarks de procesadores tanto para sistemas de escritorio como para servidores de un solo procesador.
  - SPEC INT: conjunto de benchmarks de números enteros.
  - SPEC FP: conjunto de benchmarks de punto flotante.
- SPEC SFS: mide el rendimiento de NFS; prueba el rendimiento del sistema de E/S (tanto de disco como de red), así como del procesador.
- SPEC Web: simula múltiples clientes que solicitan páginas estáticas y dinámicas de un servidor, así como clientes que publican datos a un servidor.
- SPEC jbb: mide el rendimiento del servidor para aplicaciones web escritas en java.
- SPEC virt_Sc2010: benchmarks de punto a punto de servidores de centros de datos virtualizados; Incluye el hardware, la capa de virtualización y el sistema operativo virtualizado.
- SPEC power: compara la CPU, el caché, el sistema de memoria e incluso el sistema de interconexión multiprocesador para calcular una clasificación de rendimiento de energía.

Resultados del SPEC

Idealmente, la suite se debería asemejar a una muestra estadísticamente válida del espacio de la aplicación; desafortunadamente, dicha muestra requiere más puntos de referencia de los que normalmente se encuentran en la mayoría de las suites y requiere un muestreo aleatorio, que prácticamente ningún suite hace. No obstante, es importante resumir adecuadamente un conjunto de benchmarks que se consideran relevantes.

Al final del día, en el diseño práctico de computadoras, se deben evaluar innumerables opciones de diseño para determinar sus beneficios cuantitativos relativos a través de un conjunto de benchmarks. Por otro lado, los consumidores que intentan elegir una computadora dependerán de las mediciones de rendimiento de los benchmarks.

Resultados del SPEC

Queremos resumir los resultados de rendimiento de la suite en un solo número. Un posible enfoque sencillo sería comparar las medias aritméticas de los tiempos de ejecución de los programas de la suite. Este método tiene claros inconvenientes, ya que favorece programas donde el tiempo de ejecución es mayor. Una alternativa sería agregar un factor de ponderación a cada benchmark y utilizar la media aritmética ponderada. Ahora surge el problema de cómo elegir las pesas; cada empresa puede tener un conjuto favorito de pesas.

Resultados del SPEC

En lugar de elegir pesos, podríamos normalizar los tiempos de ejecución a una computadora de referencia dividiendo el tiempo de la computadora de referencia con el tiempo de la computadora que se está calificando, lo que produciría una relation proporcional al rendimiento.

SPEC utiliza este enfoque y lo llama SPECRatio. Tiene la calidad de que coincide con la forma en que comparamos un rendimiento relativo.

Resultados del SPEC

Por ejemplo, supongamos que comparamos las siguientes ESPECRatios:

$$1.25 = \frac{\text{SPECRatio}_A}{\text{SPECRatio}_B} = \frac{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_A}}{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_B}} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

Resultados del SPEC

Debido a que SPECRatio es una relación en lugar de un tiempo de ejecución absoluto, la media debe calcularse utilizando una media geométrica. La fórmula es:

$$\text{Geometric mean} = \sqrt[n]{\prod_{i=1}^{n} sample_i}$$

donde sample *i* es la SPECRatio del programa *i*.

Resultados del SPEC

El uso de la media geométrica garantiza propiedades importantes.

1. La media geométrica de las relaciones es la misma que la relación de las medias geométricas.
2. La relación de las medias geométricas es igual a la media geométrica de las relaciones de rendimiento, lo que implica que la elección del ordenador de referencia es irrelevante.

$$\frac{\text{Geometric mean}_A}{\text{Geometric mean}_B} = \frac{\sqrt[n]{\prod_{i=1}^{n} \text{SPECRatio A}_i}}{\sqrt[n]{\prod_{i=1}^{n} \text{SPECRatio B}_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{SPECRatio A}_i}{\text{SPECRatio B}_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\dfrac{\text{Execution time}_{\text{reference}_i}}{\text{Execution time}_{A_i}}}{\dfrac{\text{Execution time}_{\text{reference}_i}}{\text{Execution time}_{B_i}}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Execution time}_{B_i}}{\text{Execution time}_{A_i}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Performance}_{A_i}}{\text{Performance}_{B_i}}}$$