微算機系統 第四章
Accessing and Understanding Performance

陳伯寧 教授
電信工程學系
國立交通大學

Performance

- *Why study Performance?*
  - ◆*See through the marketing hype*
  - ◆*Make intelligent choice of a computer system to suit your need*
  - ◆*Key to understand the motivation behind the design of particular aspects of a machine*

Performance

- *You may ask:*
  - ◆ *Why is some hardware better than others for different programs?*
  - ◆ *What factors of system performance are hardware related?*
    - *(e.g., Do we need a new machine, or a new operating system?)*
  - ◆ *How does the machine's instruction set affect performance?*
- *Or more directly, you will be concerned with*:
  - ◆ *If I upgrade a machine with a new processor or a new component, what do I gain?*
  - ◆ *If we add a new machine to the lab, what do we increase?*

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- *How much faster is the Concorde compared to the 747?*
- *How much bigger is the 747 than the Douglas DC-8?*

Computer performance = Time

- *Response Time (also named wall-clock time, elapsed time, latency, or execution time)*
  - *How long does it take to execute my program?*
  - ***Definition:** The time between the start and completion of a task, including disk accesses, memory accesses, input/output activities, operating system overhead … everything.*
- *Throughput*
  - *How many jobs can the machine run at once?*
  - *What is their average execution rate?*
  - ***Definition:** The total amount of work done in a given time*

Computer performance = Time

- *In a personal computer system, perhaps **response time (elapsed time)** is more essential than **throughput**.*
  - *This indicates that we may define the computer performance as:*

$$\text{Performance} = \frac{1}{\text{Response Time}}$$

  - *Then, "X is n times faster than Y" if*

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

Response time

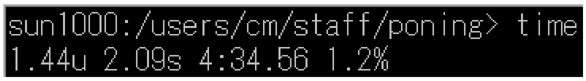
- *Response time (elapsed time)*
  - ◆ *A useful number, but often not good for comparison purposes. Why?*
    - – *The time-shared nature of operating system (OS) may result in varied response time for a task, especially when some other tasks are executed simultaneously.*
    - – *Often, the OS is designed to optimize throughput rather to minimize the response time for a specific task.*
  - ◆ *Hence, alternative performance index as **CPU time** is introduced.*
    - – ***Definition:** The time the CPU spends computing for this task, where the time waiting for I/O and used by other tasks is excluded.*

CPU time

- *CPU time = user CPU time + system CPU time*
  - ◆ *User CPU time = The CPU time spent executing the lines of code that are "**in**" the program*
  - ◆ *System CPU time = The CPU time spent in the operating system performing on behalf of the program*
  - ◆ *Note: Differentiating between **system CPU time** and **user CPU time** is difficult to do accurately, because it is hard to assign responsibility for operating system activities to one user program rather than another.*

CPU time

- *Example*



```
sun1000:/users/cm/staff/poning> time
1.44u 2.09s 4:34.56 1.2%
```

  - ◆ *User CPU time = 1.44 seconds*
  - ◆ *System CPU time = 2.09 seconds*
  - ◆ *Response time = 4 minutes and 34 seconds = 274 seconds*
  - ◆ *1.2% = Percentage of CPU time against response time*

$$\frac{1.44+2.09}{274}=1.29\%$$

  - ◆ *98.2% of the response time was spent waiting for I/O, running other programs, or both.*

Performance indices

- *Two performance indices are used in text*
  - ◆ *System performance = Reciprocal of response time*
  - ◆ *CPU performance = Reciprocal of user CPU time*

- *Instead of reporting* ***response time*** *or* ***execution time*** *in* *<u>seconds</u>, it can be measured in <u>clock cycles</u>. Hence, it is named* ***CPU clock cycles****.*

$$\frac{\text{seconds}}{\text{program}}=\frac{\text{cycles}}{\text{program}}\times\frac{\text{seconds}}{\text{cycle}}=\frac{\frac{\text{cycles}}{\text{program}}}{\frac{\text{cycles}}{\text{second}}}$$

Clock cycles

- *Example*

$$\frac{\text{cycles}}{\text{second}} = 200 \text{ MHz} \Rightarrow \frac{\text{seconds}}{\text{cycle}} = 5 \text{ nanoseconds}$$

So, to improve CPU performance (everything else being equal) one can either

___$\downarrow$___ *the # of required cycles for a program, or*
___$\downarrow$___ *the clock cycle time or, said another way,*
___$\uparrow$___ *the clock rate.*

CPU clock cycles

- *A compiler compiles a program in high-level programming language into Machine Instructions.*
- *Example.*
  - ◆*Instruction* MOV BP, SP *for 8086 CPU = Binary machine codes 10001011 11101100*
  - ◆*Instruction* MOV [1000H], DL *for 8086 CPU = Binary machine codes 10001000 00010110 00000000 00000001*
  - ◆*Notably, as shown in the previous two examples, instructions may vary in length.*

Instructions for a program

- *Some may use "number of instructions for a program" as an index.*
- *Notably, "number of instructions for a program" is by no means a performance index.*

$$\frac{\text{cycles}}{\text{program}} \approx \#\text{ of instructions for a program} \times \text{CPI}$$

- ◆ *CPI = Average clock cycles per instruction*
- ◆ *CPI is a way of comparing two different implementations of the same instruction set architecture.*

CPU time revisited

- *Summary*

$$\begin{aligned} \text{CPU clock cycles} &= \text{Instruction count} \times \text{CPI} \\ \text{CPU time} &= \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time} \\ &= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \end{aligned}$$

- *The above formula is useful because it separates the three key factors that affect performance.*

Classification of Instructions

- *It is sometimes classified the instructions into several classes so that a more accurate CPI for each class can be obtained.*

$$\text{CPU clock cycles} = \sum_i \text{Instruction count for class } i \times \text{CPI}_i$$

$$\text{CPU time} = \sum_i \frac{\text{Instruction count for class } i \times \text{CPI}_i}{\text{Clock rate}}$$

Classification of Instructions

- *Multiplication takes more time than addition*
- *Floating point operations take longer than integer ones*
- *Accessing memory takes more time than accessing registers*

- *Partial summary*
  - ◆*A program may be measured by*
    - – *CPU time*
    - – *CPU cycles*
    - – *Instruction count*

CPU time revisited

- *Example*
  - *The hardware designer provides*
    - *Instruction class A: CPI = 1*
    - *Instruction class B: CPI = 2*
    - *Instruction class C: CPI = 3*
  - *Two macro codes are written, respective using*

Macro code	*Instruction counts for instruction class*		
	A	*B*	*C*
1	*2*	*1*	*2*
2	*4*	*1*	*1*

CPU time revisited

- *Then*

$$\text{CPU clock cycle}_1 = (2\times 1)+(1\times 2)+(2\times 3) = 10 \text{ cycles}$$
$$\text{CPU clock cycle}_2 = (4\times 1)+(1\times 2)+(1\times 3) = 9 \text{ cycles}$$

$$\text{Instruction count}_1 = 2+1+2 = 5 \text{ instructions}$$
$$\text{Instruction count}_2 = 4+1+1 = 6 \text{ instructions}$$

- *We conclude that macro code 2 is faster, even if it uses more instructions!*
- *The morale of the example is that it is danger to use the* ***instruction count*** *as a major performance index (for certain types of machines).*

Now that we understand cycles

- *A given program will require*
  - *some number of instructions (machine instructions)*
  - *some number of cycles*
  - *some number of seconds*
- *We have a vocabulary that relates these quantities:*
  - *cycle time (seconds per cycle)*
  - *clock rate (cycles per second)*
  - *CPI (cycles per instruction)*
  - ***MIPS (millions of instructions per second)***

CPI Example

- *Suppose we have two implementations of the same instruction set architecture (ISA)*

  For some program,

  Machine A has a clock cycle time of 250 ps and a CPI of 2.0
  Machine B has a clock cycle time of 500 ps and a CPI of 1.2

  What machine is faster for this program, and by how much?

$$\frac{Time_A}{Time_B} = \frac{I \times 2.0 \times 250}{I \times 1.2 \times 500} = \frac{5}{6}$$

MIPS example

- *Two different compilers are being tested for a 4 GHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.*

  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

- *Which sequence will be faster according to MIPS?*
- *Which sequence will be faster according to execution time?*

Answer to MIPS example

$$\begin{cases} 1st = 5\times 1 + 1\times 2 + 1\times 3 = 10 \text{ million cycles} \\ 2nd = 10\times 1 + 1\times 2 + 1\times 3 = 15 \text{ million cycles} \end{cases}$$

$$\begin{cases} MIPS_1 = \dfrac{(5+1+1)\text{million instructions}}{10 \text{ million cycles}} \times \dfrac{\text{4G cycles}}{\text{second}} = 2.8G \\ MIPS_1 = \dfrac{(10+1+1)\text{million instructions}}{15 \text{ million cycles}} \times \dfrac{\text{4G cycles}}{\text{second}} = 3.2G \end{cases}$$

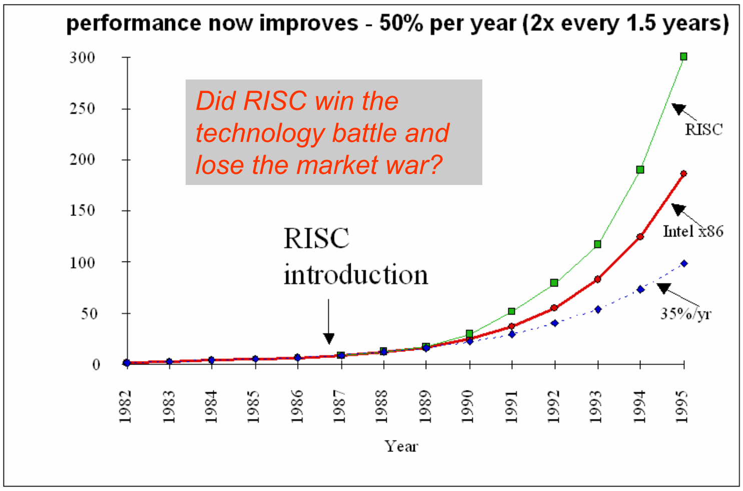
CISC versus RISC

- *Complex Instruction Set Computer Architecture*
  - ***Philosophy:*** *Language-oriented design*
  - ***Example:*** *Intel x86 series, VAX*
    - *VAX: minimize code size, make assembly language easy (but machine code complex)*
      - Instructions from 1 to 54 bytes long!

CISC versus RISC

- *Reduced Instruction Set Computer Architecture*
  - ***Philosophy:*** *Less programming was being done at the assembly level (more at the compiler level) due to advances in compiler technology, memory cost, and programming language.*
  - ***Characteristics:*** *Fixed instruction lengths, load-store instruction sets, limited addressing modes, and limited operations.*
  - ***Examples:*** *MIPS, Sun SPARC, Hewlett-Packard PA-RISC, IBM PowerPC, and DEC Alpha.*
  - *Virtually all new instruction sets since 1982 have been RISC*

Processor performance (SPEC)



Benchmarks

- *Performance best determined by running a real application*
  - *Use programs typical of expected workload*
  - *Or, typical of expected class of applications e.g., compilers/editors, scientific applications, graphics, etc.*
- *SPEC (System Performance Evaluation Cooperative)*
  - *companies have agreed on a set of real program and inputs*
  - *valuable indicator of performance (and compiler technology)*
  - *can be abused*

Benchmarks

- *Benchmark abuse*
  - ◆*A designer may try to make some sequence of instructions run especially fast because the sequence occurs in a benchmark.*
  - ◆*E.g. compilers with special-purpose optimizations targeted at specific benchmarks.*

Misleading by benchmark

- *Example of misleading by benchmark*
  - ◆*SPEC (System Performance Evaluation Cooperative) is a well-known benchmark, which will be introduced later.*
  - ◆*In late 1995, Intel published a new performance rating for the integer SPEC benchmarks running on a Pentium processor and using an* ***internal compile****,* ***not used outside of Intel****.*
  - ◆*Unfortunately, the code produced for one of the benchmarks was wrong, a fact that was discovered when a competitor read through the binary to understand how Intel had sped up one of the programs in the benchmark suite so dramatically.*
  - ◆*In January 1996, Intel admitted the error and restated the performance.*

Misleading by benchmark

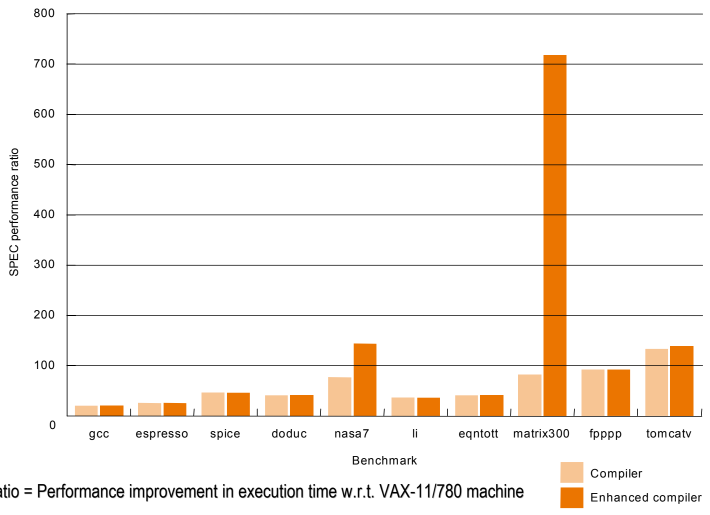
An embarrassed Intel Corp. acknowledged Friday that a bug in a software program known as a compiler had led the company to overstate the speed of its microprocessor chips on an industry benchmark by 10 percent. However, industry analysts said the coding error…was a sad commentary on a common industry practice of "cheating" on standardized performance tests…The error was pointed out to Intel two days ago by a competitor, Motorola …came in a test known as SPECint92…Intel acknowledged that it had "optimized" its compiler to improve its test scores. The company had also said that it did not like the practice but felt to compelled to make the optimizations because its competitors were doing the same thing…At the heart of Intel's problem is the practice of "tuning" compiler programs to recognize certain computing problems in the test and then substituting special handwritten pieces of code…

Saturday, January 6, 1996 New York Times

Misleading by benchmark

- *Example of misleading by benchmark*
  - ◆*The first release of the SPEC suite in 1989 included a benchmark called matrix300.*
  - ◆*Matrix300 consists solely of a series of matrix multiplications, in which 99% of the execution time is in a single line.*
  - ◆*Hence, a designer can easily enhance a compiler to mislead the customer by showing "dramatic" performance improve in matrix operations.*
  - ◆*Notably, special compiler is not the general compiler used by the users; so, a user may be disappointed when using this machine.*

SPEC'89 on IBM Powerstation 550



Performance ratio = Performance improvement in execution time w.r.t. VAX-11/780 machine

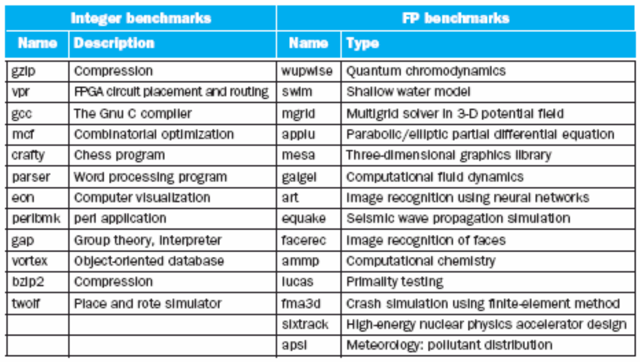
SPEC'92

- *In 1992 release of the SPEC benchmark suite, matrix300 was dropped!*

SPEC'95

	Benchmark	Description
C integer benchmarks	go	Artificial intelligence; plays the game of Go
	m88ksim	Motorola 88k chip simulator; runs test program
	gcc	The Gnu C compiler generating SPARC code
	compress	Compresses and decompresses file in memory
	li	Lisp interpreter
	ijpeg	Graphic compression and decompression
	perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
	vortex	A database program
Fortran 77 floating-point benchmarks	tomcatv	A mesh generation program
	swim	Shallow water model with 513 x 513 grid
	su2cor	quantum physics; Monte Carlo simulation
	hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
	mgrid	Multigrid solver in 3-D potential field
	applu	Parabolic/elliptic partial differential equations
	trub3d	Simulates isotropic, homogeneous turbulence in a cube
	apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
	fpppp	Quantum chemistry
	wave5	Plasma physics; electromagnetic particle simulation

SPEC CPU 2000 benchmark



Integer benchmarks		FP benchmarks	
Name	**Description**	**Name**	**Type**
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution

FIGURE 4.5 The SPEC CPU2000 benchmarks. The 12 integer benchmarks in the left half of the table are written in C and C++, while the floating-point benchmarks in the right half are written in Fortran (77 or 90) and C. For more information on SPEC and on the SPEC benchmarks, see www.spec.org. The SPEC CPU benchmarks use wall clock time as the metric, but because there is little I/O, they measure CPU performance.

Normalized benchmark

- *The execution time measurements for SPEC CPU 2000 benchmark are normalized by* ***dividing the executing time*** *on a Sun Ultra 5_10 with a 300 MHz processor* ***by the executing time*** *on the measured computer, which yields a measure called the* ***SPEC ratio****.*
- *A CINT2000 or CFP2000 summary measurement is obtained by taking the* ***geometric mean*** *of the SPEC ratios.*

Why geometric mean?

- *Geometric mean.*

$$\text{geometric mean} = \sqrt[n]{\prod_{i=1}^{n} \text{SPEC ratio}_i}$$

Question: Why geometric mean? Why not arithmetic mean?
Answer: Arithmetic mean will depend on the choice of the reference machine.

Why geometric mean?

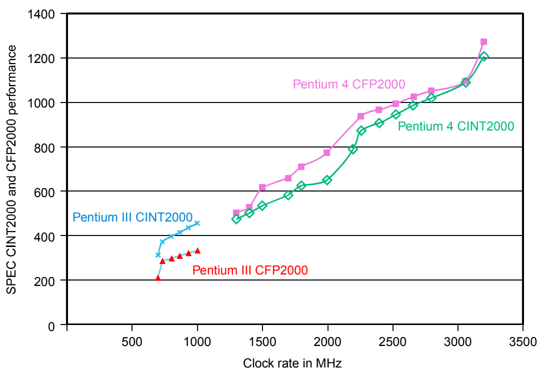
- *Example: Under arithmetic mean,*
  - ◆ *A is 5.05 times faster than B, when A is the reference.*
  - ◆ *B is 5.05 times faster than A, when B is the reference.*

	Time on A	Time on B	Normalized to A		Normalized to B	
			A	B	A	B
Program 1	1	10	1	10	0.1	1
Program 2	1000	100	1	0.1	10	1
Arithmetic mean of time or normalized time	500.5	55	1	5.05	5.05	1
Geometric mean of time or normalized time	31.6	31.6	1	1	1	1

When execution times are normalized, only a geometric mean can be used to consistently summarize the normalized result. However, geometric mean does not predict execution time.

Clock rate versus performance

- *Does doubling the clock rate double the performance?*
  - ◆ *Not quite "memory" performance could be a factor other than CPU clock rates..*



Measured using Dell Precision Desktop Computers

Clock rate versus performance

- *The performance loss is mostly due to the speed of the main memory remains the same, while the system clock rates increase.*
- ***Think:** Can a machine with a slower clock rate have better performance?*

Things to remember

- *Performance is specific to a particular program!*
- *For a given computer architecture, performance improvement jointly comes from a better balance among*
  - *clock rate*
  - *CPI*
  - *instruction count*

Amdahl's law

Execution Time After Improvement
= Execution Time Unaffected
+ (Execution Time Affected /Amount of Improvement)

$$\mathbf{Exe_{after} = Exe_{unaffected} + Exe_{affected} / Amount_{improvement}}$$

Amdahl's law

- *Example:*

  *"Suppose a program runs in 100 seconds on a machine, with **multiply** responsible for 80 seconds of this time. How much do we have to improve the **speed of multiplication** if we want the program to run **4 times** faster?"*

  $100/4 = 80/n + 20 \rightarrow n = 16$

Amdahl's law

- *How about making it 5 times faster?*

  $100/5 = 80/n + 20 \rightarrow n = \infty$

- **Principle: Make the common case fast**

$$\text{Speedup} = \frac{\text{Performance After Improvement}}{\text{Performance Before Improvement}} = \frac{\text{Execution Time Before Improvement}}{\text{Execution Time After Improvement}}$$

MIPS revisited

- *MIPS : Million instructions per second*
- *Problems of MIPS as a performance measure, rather than Execution Time*
  - *Cannot compare computers with different instruction sets using MIPS.*
  - *A machine cannot have a single MIPS rating for all programs.*
  - *MIPS can possibly vary inversely with performance due to variable-length-nature of CIST instructions!*

MIPS revisited

- *Example. To compile a program with two different compilers over a machine with clock rate 500MHz*

Code from	*Instruction counts (in billions) for each instruction class*		
	A (CPI=1)	*B (CPI=2)*	*C (CPI=3)*
Compiler 1	*5*	*1*	*1*
Compiler 2	*10*	*1*	*1*

$$\text{Execution Time}_1 = \frac{(5\times 1 + 1\times 2 + 1\times 3)\times 10^9}{500\times 10^6} = 20\text{ seconds}$$

$$\text{Execution Time}_2 = \frac{(10\times 1 + 1\times 2 + 1\times 3)\times 10^9}{500\times 10^6} = 30\text{ seconds}$$

$$\left.\begin{array}{l}\text{MIPS}_1 = \dfrac{(5+1+1)\times 10^9}{20\times 10^6} = 350 \\ \text{MIPS}_2 = \dfrac{(10+1+1)\times 10^9}{30\times 10^6} = 400\end{array}\right\}$$

MIPS fails to give a true picture of performance!

MIPS revisited

- *Peak MIPS : Obtained by choosing an instruction mix that minimizes CPI.*

- *Sometimes, the computer manufacturers will intentionally neglecting the word "peak" in their performance announcement.*
  - *With a 50MHz clock rate, Intel i860 was announced to achieve 100 **MFLOPS** (millions of floating-point operations per second) and 150 **MOPS** (millions of operations per second), in which "peak" is neglected.*
  - *Although a 33-MHz R3000 processor that appeared at about the same time as Intel i860 only yielded peak performance of 16 **MFLOPS** and 33 **MOPS**, the SPEC benchmarks showed R3000 was actually about 15% faster than i860.*

MIPS revisited

- *"Difference between MFLOPS and MIPS" or "Difference between floating-point operation and instruction"*
  - *Floating-point operation = mathematical operations, such as +, −, ×, ÷, etc, applied onto floating-point numbers; may consist of a few instructions*

MIPS revisited

- *Relative MIPS :*

$$\text{Relative MIPS} = \frac{\text{Time}_{\text{reference}}}{\text{Time}_{\text{unrated}}} \times \text{MIPS}_{\text{reference}}$$

where

$\text{Time}_{\text{reference}}$=*Execution time of a program on the reference machine,*

$\text{Time}_{\text{unrated}}$=*Execution time of the same program on the machine to be rated,*

$\text{MIPS}_{\text{reference}}$=*Agreed-upon MIPS rating of the reference machine.*

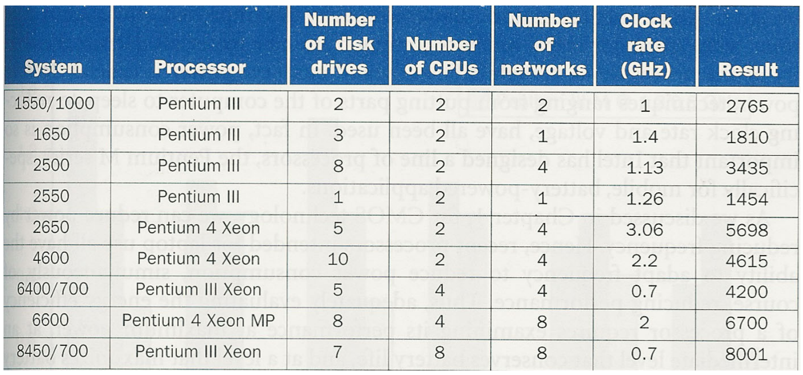
Some notes on SPEC histories

- ◆*The common reference machine was used to be the VAX-11/780, a 1-MIPS machine, due to its popularity at its time (1977~1981).*
- ◆*Such a 1-MIPS rating was widely believed until Joel Emer of DEC measured the VAX-11/780 under a timesharing load, and found that the actual VAX-11/780 MIPS rate was 0.5!*
- ◆*Ironically, subsequent VAXs that run 3 million VAX instructions per second were therefore called 6-MIPS machines because they run six times faster than the VAX-11/780!*

Some notes on SPEC histories

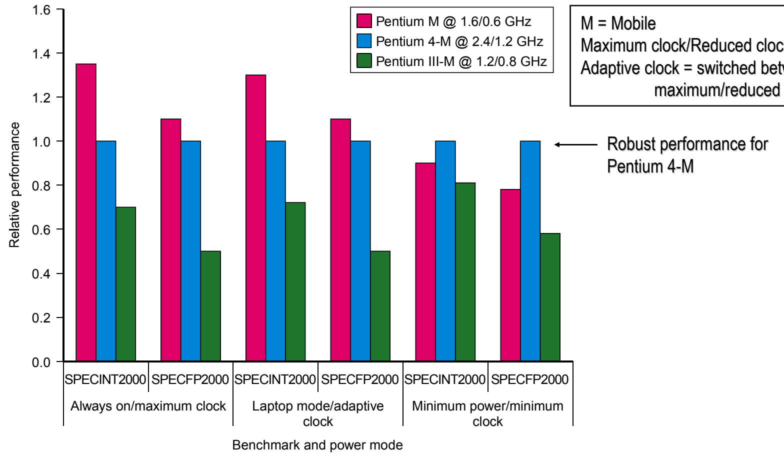
- ◆*SPEC'95 changed the base machine for normalization from VAX11-780 to Sun SPARCstation 10/40 because the original base machine were becoming difficult to find.*
- ◆*In 1996, SPEC introduced its first benchmark designed to measure Web server performance, which was later superseded by SEPCweb99 announced in 1999.*

SPECweb99



System	Processor	Number of disk drives	Number of CPUs	Number of networks	Clock rate (GHz)	Result
1550/1000	Pentium III	2	2	2	1	2765
1650	Pentium III	3	2	1	1.4	1810
2500	Pentium III	8	2	4	1.13	3435
2550	Pentium III	1	2	1	1.26	1454
2650	Pentium 4 Xeon	5	2	4	3.06	5698
4600	Pentium 4 Xeon	10	2	4	2.2	4615
6400/700	Pentium III Xeon	5	4	4	0.7	4200
6600	Pentium 4 Xeon MP	8	4	8	2	6700
8450/700	Pentium III Xeon	7	8	8	0.7	8001

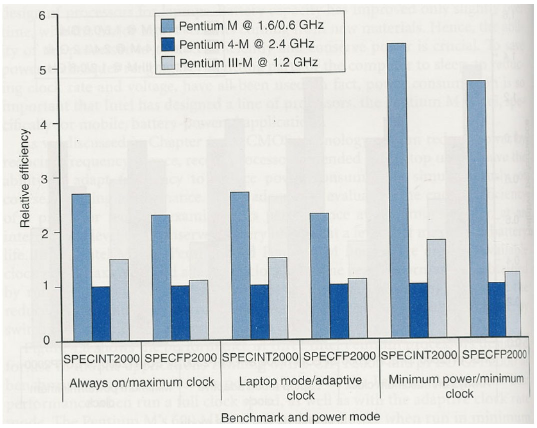
Power and energy efficiency



M = Mobile
Maximum clock/Reduced clock rates
Adaptive clock = switched between maximum/reduced clocks

Power and energy efficiency

- *Energy efficiency = Benchmark per joule*



Final notes

- ***Performance** is just one of the consideration of computer designs.*
- *Another important factor is **cost**!*
- *Commercial computer designs should balance between cost and performance.*

Suggestive exercises

- *4.1~4.9, 4.11~4.14, 4.17~4.18, 4.22~4.23, 4.25, 4.30, 4.33, 4.35~4.46, 4.49*
- *Should you have questions on exercises, you can ask the teaching assistants (助教).*