
Chapter 4

Why Performance ?

- **Purchasing perspective (given a collection of machines)**
 - best performance ?
 - least cost ?
 - best performance / cost ?
- **Design perspective (faced with design options)**
 - best performance improvement ?
 - least cost ?
 - best performance / cost ?
- **Both require**
 - basis for comparison
 - metric for evaluation

- **Our goal is to understand cost & performance implications of architectural choices**

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation

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?

Which of these airplanes has the best performance?



<u>Airplane</u>	<u>Passengers</u>	<u>Range (mi)</u>	<u>Speed (mph)</u>
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 ?
 - Concord is $1350 \text{ mph} / 610 \text{ mph} = 2.2$ times faster
- How much bigger is the 747 than the Douglas DC-8 ?

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 more throughput is the Concorde compared to the 747 ?

- Concord is $1350 \text{ mph} * 132 \text{ passengers} = 178,200$
- 747 is $610 \text{ mph} * 470 \text{ passengers} = 286,700$

→ 747 is 1.6 times more throughput than Concorde

Computer Performance: TIME, TIME, TIME

- Response Time (latency) <- user's perspective
 - How long does it take for my job to run?
 - How long does it take to execute a job?
 - How long must I wait for the database query?
- Throughput <- system manager's perspective
 - How many jobs can the machine run at once?
 - What is the average execution rate?
 - How much work is getting done?
- *If we upgrade a machine with a new processor what do we increase?*
If we add a new machine to the lab what do we increase?

Execution Time

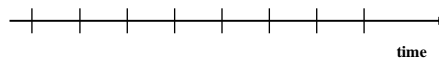
- **Elapsed Time**
 - counts everything (*disk and memory accesses, I/O, etc.*)
 - a useful number, but often not good for comparison purposes
- **CPU time**
 - doesn't count I/O or time spent running other programs
 - can be broken up into system time, and user time
- **Our focus: user CPU time**
 - time spent executing the lines of code that are "in" our program

Clock Cycles

- **Instead of reporting execution time in seconds, we often use cycles**

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

- **Clock "ticks" indicate when to start activities (one abstraction):**



- **cycle time = time between ticks = seconds per cycle**
- **clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)**

A 200 Mhz. clock has a $\frac{1}{200 \times 10^6} \times 10^9 = 5$ nanoseconds **cycle time**

How to Improve Performance

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

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

_____ the # of required cycles for a program, or

_____ the clock cycle time or, said another way,

_____ the clock rate.

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?"

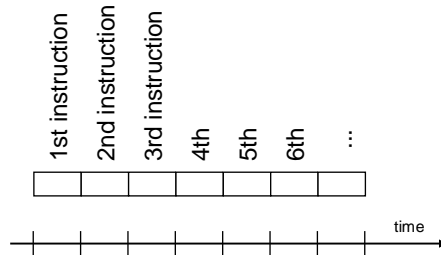
- A: 10 sec. 400Mhz C cycles/prog.
- B: 6 sec. ??? 1.2 C cycles/prog.

$$10 = C / 400 \times 10^6$$

$$6 = 1.2C / ??? \quad ??? \Rightarrow 800\text{Mhz}$$

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

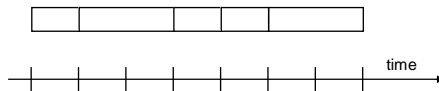


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



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

- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

How to Improve Performance (revisited)

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

IC (instruction count)
fixed for a given ISA

CPI (cycles per instruction)
“varied” for each instruction
will use average CPI for a given ISA

CT (cycle time)
fixed for a given technology

Example

Instruction	IC	CPI	IC*CPI	% Time
ALU	50	1	50	23%
Load	20	5	100	45%
Store	10	3	30	14%
Branch	20	2	40	18%
			<u>220</u>	

Typical Mix

With this mix of instructions, average CPI is $220 / 100 = 2.2$

I.e., average CPI can be obtained from $(\sum \text{CPI}_i * \text{IC}_i) / \text{ICt} = \sum (\text{CPI}_i * \text{IC}_i / \text{ICt})$

* ICt : total instruction count

* IC_i/ICt : frequency of the instruction

How to Improve Performance (revisited)

$$\frac{\text{seconds}}{\text{program}} = IC \times CPI \times CT$$

So, to improve performance you can either

	inst. count	CPI	cycle time
Program			
Compiler			
Inst. Set			
Organization			
Technology			

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 10 ns. and a CPI of 2.0

Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

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

- *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

Example

- Percentage of instructions of a program (CPU A)
 - compare 20%, branch 20%, others 60%
 - compare 1 cycle, branch 2 cycles, others 1 cycle (CPI_i)
 - CPI_A =
 - CPU_A =
- Percentage of instructions of the same program (CPU B)
 - combine “compare & branch” in one instruction
 - compare & branch 2 cycles, others 1 cycle (CPI_i)
 - CPI_B =
 - CPU_B =

of Instructions Example

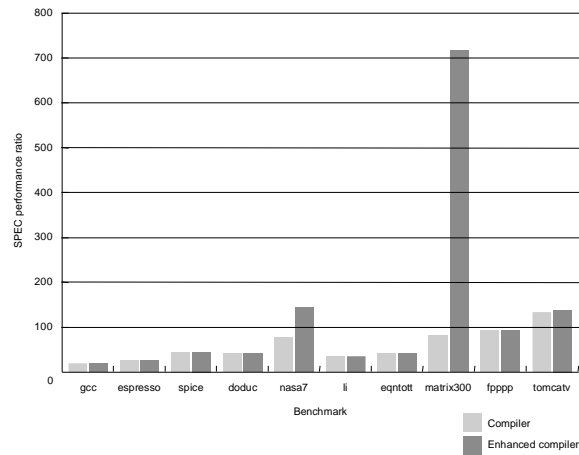
- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

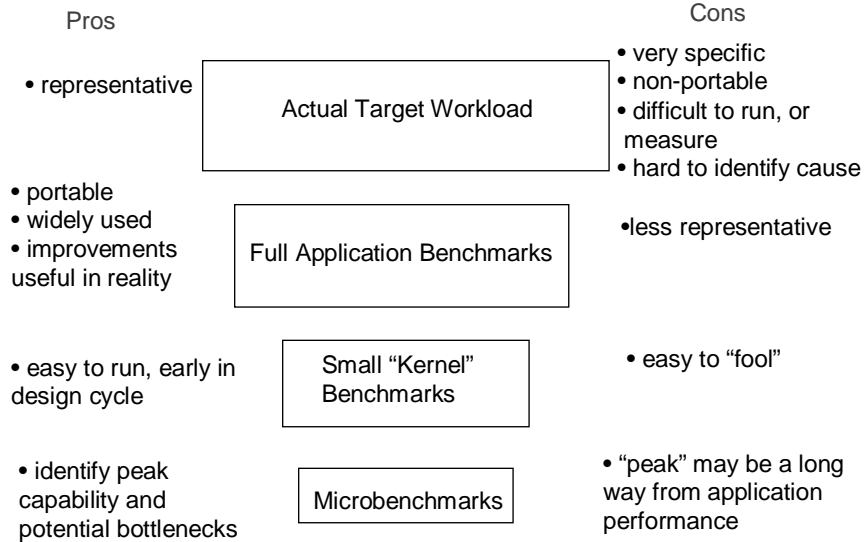
Which sequence will be faster? How much?
What is the CPI for each sequence?

SPEC '89

- Compiler “enhancements” and performance



Basis of Performance Evaluation



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
 - can still be abused
 - valuable indicator of performance (and compiler technology)

SPEC

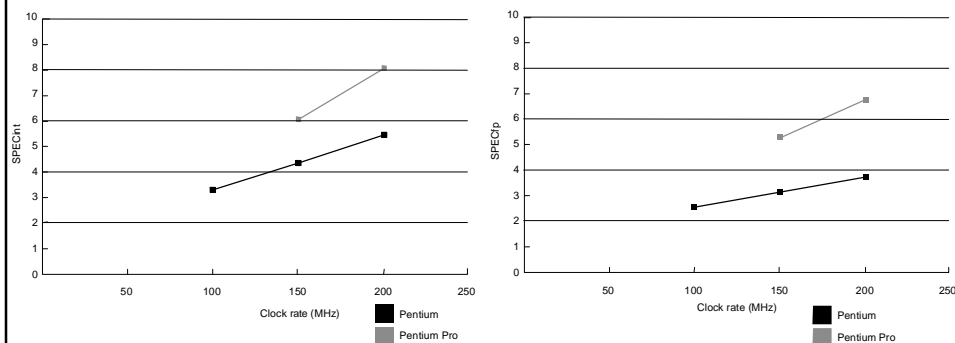
- SPEC89 - reference system is VAX-11/780 (geometric mean for 10 benchmark programs)
- SPECint92, SPECfp92 : Integer and floating-point programs
- SPECint95 (8 integer programs) + SPECfp95 (10 fp programs)

SPEC '95

Benchmark	Description
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
ljpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
su2cor	quantum physics: Monte Carlo simulation
hydro2d	Astrophysics: Hydrodynamic Navier 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
fpopp	Quantum chemistry
wave5	Plasma physics: electromagnetic particle simulation

SPEC '95

*Does doubling the clock rate double the performance?
Can a machine with a slower clock rate have better performance?*



SPEC CPU2000

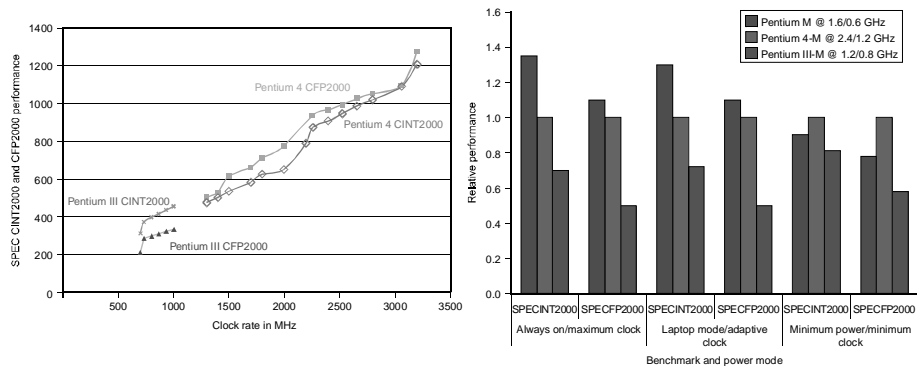
Integer benchmarks		FP benchmarks	
Name	Description	Name	Type
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial 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
perlbnk	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.

SPEC 2000

Does doubling the clock rate double the performance?

Can a machine with a slower clock rate have better performance?



How to make “ONE REPRESENTATIVE VALUE” ?

SPECint92 Programs	Execution Time (sec)	
	System A	System B
espresso	10	15
li	20	25
eqntott	15	10
compress	17	12
sc	3	5
gcc	55	45

Which system is better ?
How to compare System A to System B ?

	Computer A	Computer B	Computer C
Program P1 (secs)	1	10	20
Program P2 (secs)	1000	100	20
Total time (secs)	1001	110	40

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
Program P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
Program P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Arithmetic mean	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
Geometric mean	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
Total time	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

Comparison

- Means
 - Arithmetic mean $A=(s+f)/2$... execution time (sec)
 - Geometric mean $G=\sqrt{sf}$
 - Harmonic mean $1/H=(1/s+1/f)/2$... ???
- Depending on the reference system,
 - Arithmetic mean gives varied conclusions
 - Geometric mean gives consistent results
- If you are a manufacturer of Computer A,
 - What will you do ?
 - Which program will you try to improve, s or f ?
(s: execution time of a slow program>f: ET of a fast program)

G.Mean vs H.Mean

- Geometric mean improvement
 - when slower program becomes improved “k” times
 - $G'(s'=s/k) = \sqrt{sf} / \sqrt{k}$
 - when faster program is improved “k” times
 - $G'(f'=f/k) = \sqrt{sf} / \sqrt{k} = G'(s'=s/k)$
 - As a manufacturer, you want to show a better number (mean) but with little investment
 - ====> Improve easier
 - since either way results in the same effect !!!

G.Mean vs H.Mean (cont'd)

- Harmonic mean improvement
 - when slower program becomes improved “k” times
 - $H'(s'=s/k) = 2(s/k)f/(s/k+f) = 2sf/(s+kf)$
 - when faster program is improved “k” times
 - $H'(f'=f/k) = 2sf/(ks+f) < 2sf/(s+kf) = H'(s'=s/k)$
 - Should improve the slow program since it results in more improvement
 - “Improve the bottleneck !!!” is what H.Mean says

G.Mean vs H.Mean (cont'd)

- H.Mean means more sense than G.Mean ???
- SPEC uses G.Mean. Why ???
- What do you say about A.Mean ???
 - Solve equations with “s” and “f” ?
- “Mean” seems simple but it is NOT

Design Principles

- **Amdahl's Law**
 - Performance improvement with an accelerator is limited by the fraction of the time that the accelerator can be used
 - **Example**
 - Graphic accelerator makes 2 times faster of all graphic operations which is 30% of all operations : New Execution Time = Old ET * ???
 - **Speedup = Old ET/New ET = $1/((1-\text{Fraction})+\text{Fraction}/\text{Part_Speedup})$**
 - Cache access time is 10 times faster, hit ratio is 90% : Speedup = ???

Design Principles (cont'd)

- **Amdahl's Law**
 - Speedup < $1/(1-\text{Fraction})$ even if Part_Speedup = ∞
 - “Better find to apply accelerator at all time”

(Example) A parallel program has some amount of sequential part ($s \leq 1$), which cannot be parallelized.
=> Speedup equation ???
=> What does it imply if $s = 5\%$???
- **Make the common case fast**
- **Locality of reference**

Summary: Evaluating Instruction Sets?

Design-time metrics:

- Can it be implemented, in how long, at what cost?
- Can it be programmed? Ease of compilation?

Static Metrics:

- How many bytes does the program occupy in memory?

Dynamic Metrics:

- How many instructions are executed?
- How many bytes does the processor fetch to execute the program?
- How many clocks are required per instruction?
- How "lean" a clock is practical?

Best Metric: Time to execute the program!

