Computer Performance

COE608: Computer Organization and Architecture

Dr. Gul N. Khan
http://www.ee.ryerson.ca/~gnkhan
Electrical and Computer Engineering
Ryerson University

Overview

- Introduction to Performance
- Aspects of Performance
  - Execution time, Elapsed time, user CPU time
  - CPI, MIPS and MFLOPS
  - Benchmarks
  - Performance Metrics
- Amdahl's Law

Part of Chapter 1 of the text (4th Edition)

Understanding Computer Performance

Algorithm

- Determines number of operations executed.

Programming language, compiler, architecture

- Determine number of machine instructions executed per operation.

Processor and memory system

- Determine how fast instructions are executed.

I/O system (including OS)

- Determines how fast I/O operations are executed.

Computer Performance

- Why some hardware is better than others for different programs?
- Which factors of system performance are hardware related?

- How does the machine's instruction set affect performance?

Purchasing perspective

Given a collection of machines, which has the best performance, least cost, best performance / cost?

Design perspective

Faced with design options, which has the best performance improvement, least cost, best performance / cost?

➢ Our goal is to understand cost/performance implications of architectural choices

Performance

Consider the following planes

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 777	375	4630	610
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

Which airplane has the best performance?

- How faster is the Concorde compared to B747?
- How much bigger is B747 than the DC-8?

Plane	DC to Paris	Speed	Passengers	Throughput (p.mph)
Boeing 747	**6.5 hours**	**610 mph**	**470**	**286,700**
Concorde	**3 hours**	**1350 mph**	**132**	**178,200**

Computer Performance

Computer Performance is related to TIME, TIME and TIME

Two notions of Performance

- Time to do the task:
  Execution time, Response time (latency)
  - How long does it take for a job to run?
  - How long does it take to execute a job?
  - How long must I wait for the database query?
- Tasks per day, hour, week, sec, nsec, etc.
  - How many jobs can a machine run at once?
  - What is the average execution rate?

- *When we upgrade a Pentium-IV PC with a new i7 quad core processor:*
  What do we increase?
- *When we add a new computer system to the lab:*
  What do we increase?

Response Time and Throughput

Response time

- How long it takes to do a task

Throughput

- Total work done per unit time

e.g. tasks/transactions/… per hour

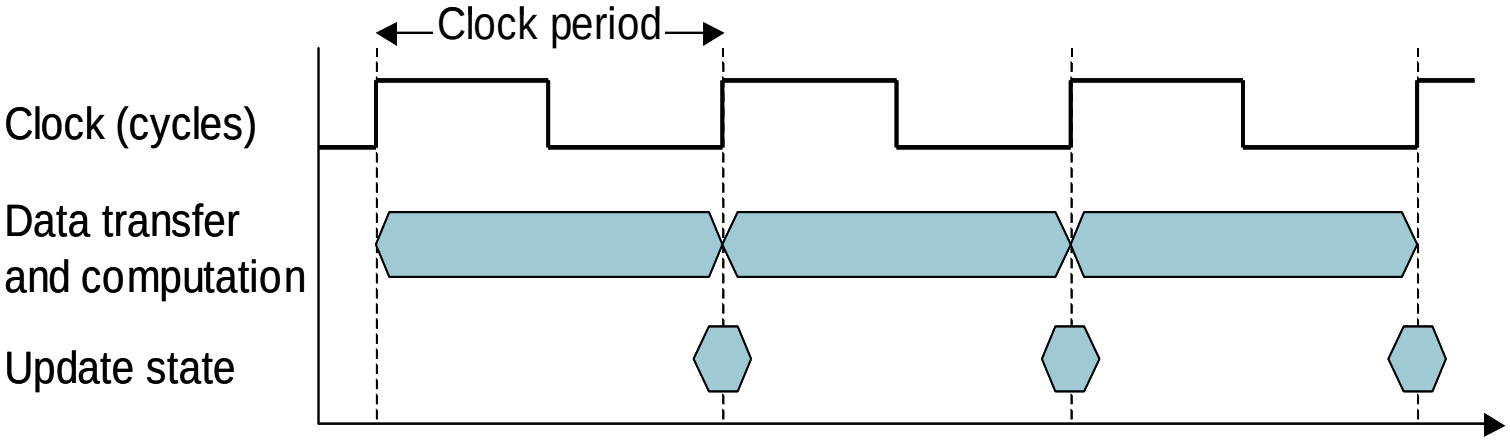
How are response time and throughput affected by:

- Replacing the processor with a faster version?
- Adding more processors?

We'll focus on response time for now…

CPU Clocking

Operation of digital hardware governed by a constant-rate of clock



Clock period: duration of a clock cycle
e.g. 250ps = 0.25ns = 250×10^{-10}ns

Clock frequency (rate): cycles per second
e.g. 4.0GHz = 4000MHz = 4.0×10^{9}Hz

Execution Time

The execution time is defined in terms of:

Elapsed Time

Counts everything

A useful number, but often not good for comparison purposes.

CPU time

Doesn't count I/O or time spent running other programs.

The user CPU time

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

An 800 MHz. clock has a cycle time of

$$\frac{1}{800 \times 10^6} \times 10^9 = 1.25 \text{ nanosec}$$

Basic Definition of Performance

For some program running on machine X,

$(\text{Performance})_X = 1 / (\text{Execution time})_X$

When X is ***n*** times faster than Y machine

$(\text{Performance})_X / (\text{Performance})_Y = \boldsymbol{n}$

Problem:

Machine A runs a program in 20 seconds

Machine B runs the same program in 25 seconds

How to Improve Performance

Everything else being equal we can either:

- Reduce the number of required cycles for a program, or
- Reduce the clock cycle time or, said another way, the clock rate.

Hardware designer often trade off clock rate against cycle count

Can we assume: # of cycles = # of instructions?

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

CPU Time

Proportional to Instruction Count

(CPU-time/Program) =?? (Instructions/Program)

When ISA is set, what can influence instruction count?

Machine Instructions:
Static count?
or dynamic count?

Program:
What type of computer architect influences the number of instructions, a given program needs?

Any additional instruction you execute takes time.

CPU time: Proportional to Clock Period
How can architects reduce clock period?

Instruction's exe time in "number of cycles".
Short clock period => Short execution time.

What ultimately limits an architect's ability to reduce the clock period?

CPU Time Example

Computer ***A***: 2GHz clock, 10-sec CPU time

Designing Computer ***B***

- Aim for a 6-sec CPU time
- Can have faster clock, but causes 1.2×clock cycles

How fast must Computer ***B*** clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Aspects of CPU Performance

	Instruction_count	CPI	Clock_cycle
Algorithm	X	X	
Programming Language	X	X	
Compiler	X	X	
ISA	X	X	X
Core organization		X	X
Technology			X

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPI: Cycles per Instruction (average)

CPI = (CPU Time×Clock Rate)/Instruction Count

$$\text{CPU time} = \text{ClockCycleTime} \times \sum_{j=1}^{n} CPI_j * I_j$$

$$CPI = \sum_{j=1}^{n} CPI_j \times F_j$$ where F is instruction frequency

and $F_j = I_j/(\text{instruction count})$

Performance Equation

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Instruction Count for a program

- Determined by program, ISA and compiler

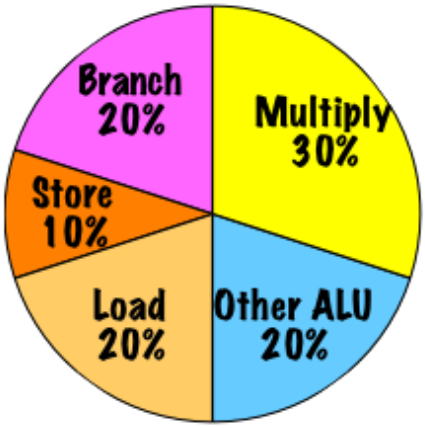
Average cycles per instruction

- Determined by CPU hardware
- If different instructions have different CPI

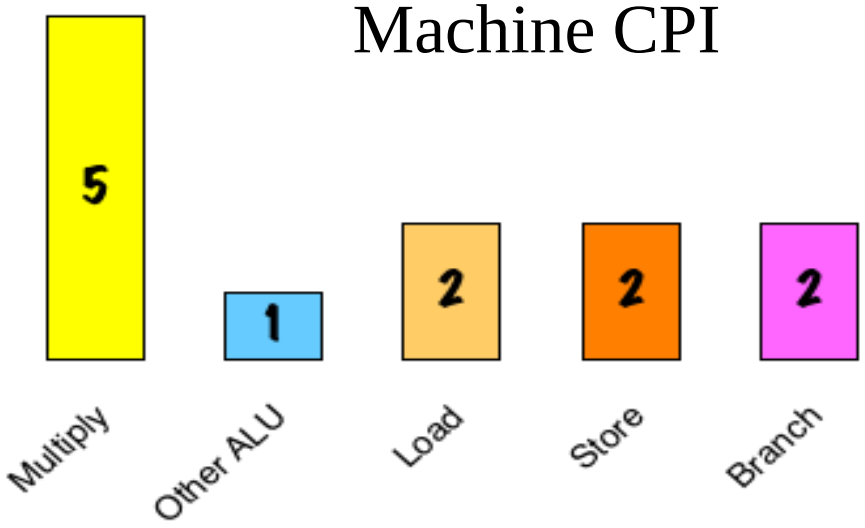
  Average CPI affected by instruction mix

CPI: Analytical Tool to Design

Program Instruction

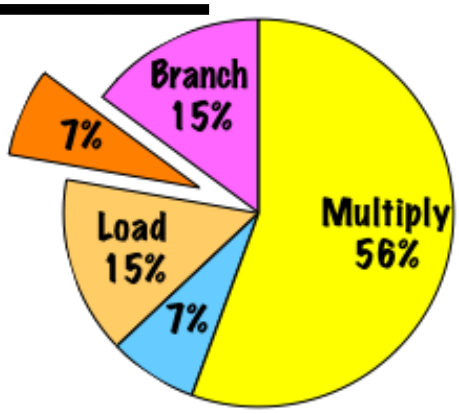


Machine CPI



$$\frac{5 \times 30 + 1 \times 20 + 2 \times 20 + 2 \times 10 + 2 \times 20}{100}$$

=



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 psec and average CPI of 2.0
Machine B has a clock cycle time of 400 psec and average CPI of 1.2

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

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

Number 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?

MIPS and MFLOPS

MIPS is often used as an alternative to time for indicating performance.

MIPS = Instruction count /(Execution time x 10^6)

This is also called native MIPS.
Faster machine will have higher MIPS

Mainly three problems with MIPS

- It does not take into account the capabilities of instructions. You cannot compare two computers with different instruction sets.
- MIPS will vary for different programs on the same machine.
- MIPS can vary inversely with performance.

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- CPI varies between programs on a given CPI

An Example

Two different compilers are tested for a 1 GHz computer with three classes of instructions:

- Class A instructions require one cycle
- Class B instructions have two cycle
- Class C require three cycles

Both compilers are used to produce a code for large piece of software. First compiler's code uses:

- 5 million Class A instructions
- 1 million Class B instructions
- 1 million Class C instructions.

The second compiler's code uses:

- 10 million Class A instructions
- 1 million Class B instructions
- 1 million Class C instructions.

Which sequence will be faster?

Benchmarks

Performance best determined by running a real application

- Use programs typical of expected workload.
- Or, typical of expected class of applications.

Small Benchmarks

- Nice for architects and designers
- Easy to standardize
- Can be abused

SPEC

(System Performance Evaluation Corporation)

- System/CPU Manufacturers and others have agreed on a set of real program and inputs
- Can still be abused (Intel's "other" bug)

  Intel compiler generated wrong code for Pentium showing huge performance gain.

- Valuable indicator of performance.

Benchmark Games

Saturday, January 6, 1996 New York Times

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…

SPEC CPU Benchmark

Programs used to measure performance

- Supposedly typical of actual workload

Standard Performance Evaluation Corp:SPEC

Develops benchmarks for CPU, I/O, Web, …

SPEC ’95: Based on real programs

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
ijpeg	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 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

Main Sources for CPU performance improvement.

- Clock rate
- CPI due to processor organization
- Compiler enhancement

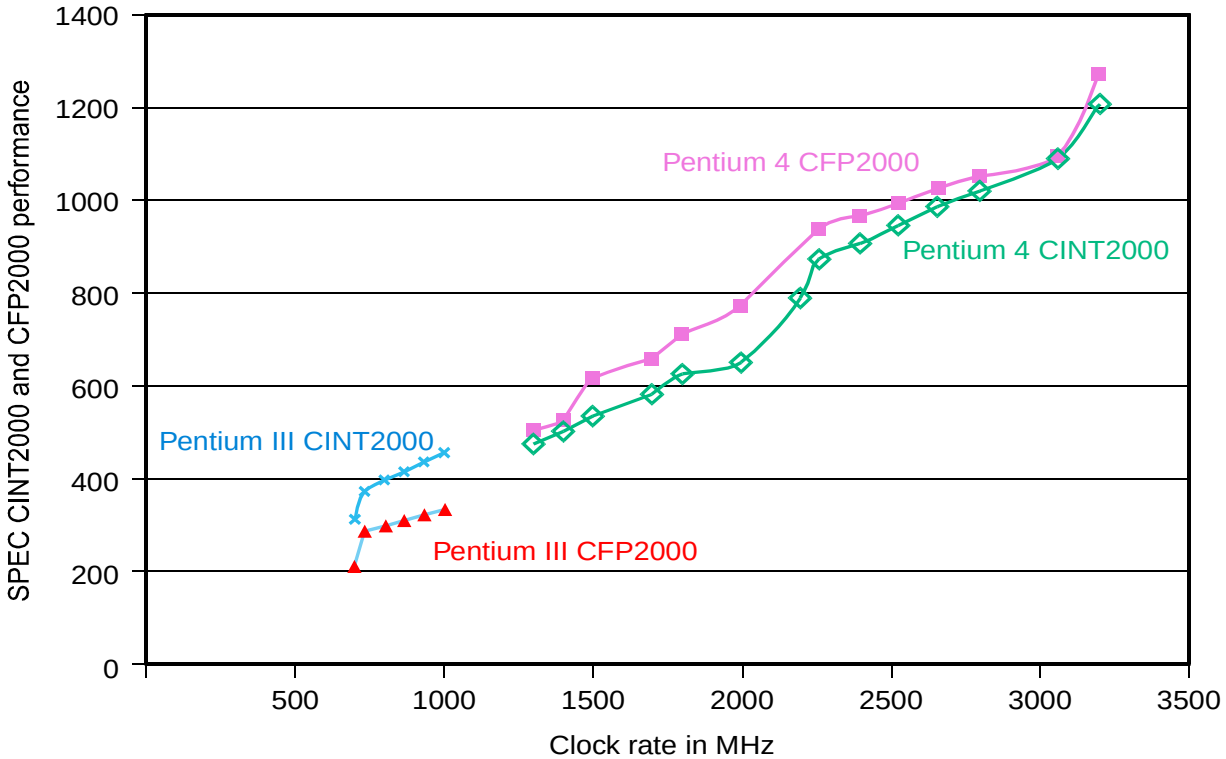
SPEC CPU2000

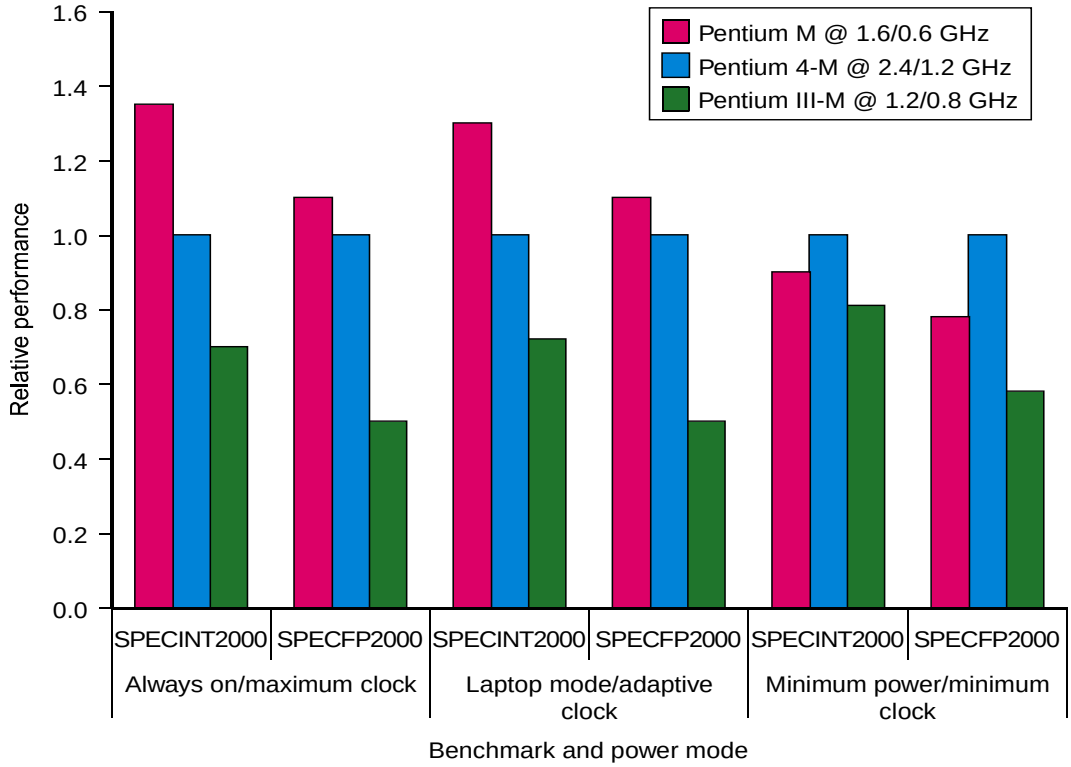
Integer benchmarks		FP benchmarks	
Name	**Description**	**Name**	**Type**
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic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

SPEC CPU2000

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





SPEC CPU2006

Elapsed time to execute a set of programs

- Negligible I/O, so focuses on CPU performance

Normalize relative to reference machine.

Summarize as geometric mean of performance ratios:

CINT2006 (integer), CFP2006 (floating-point)

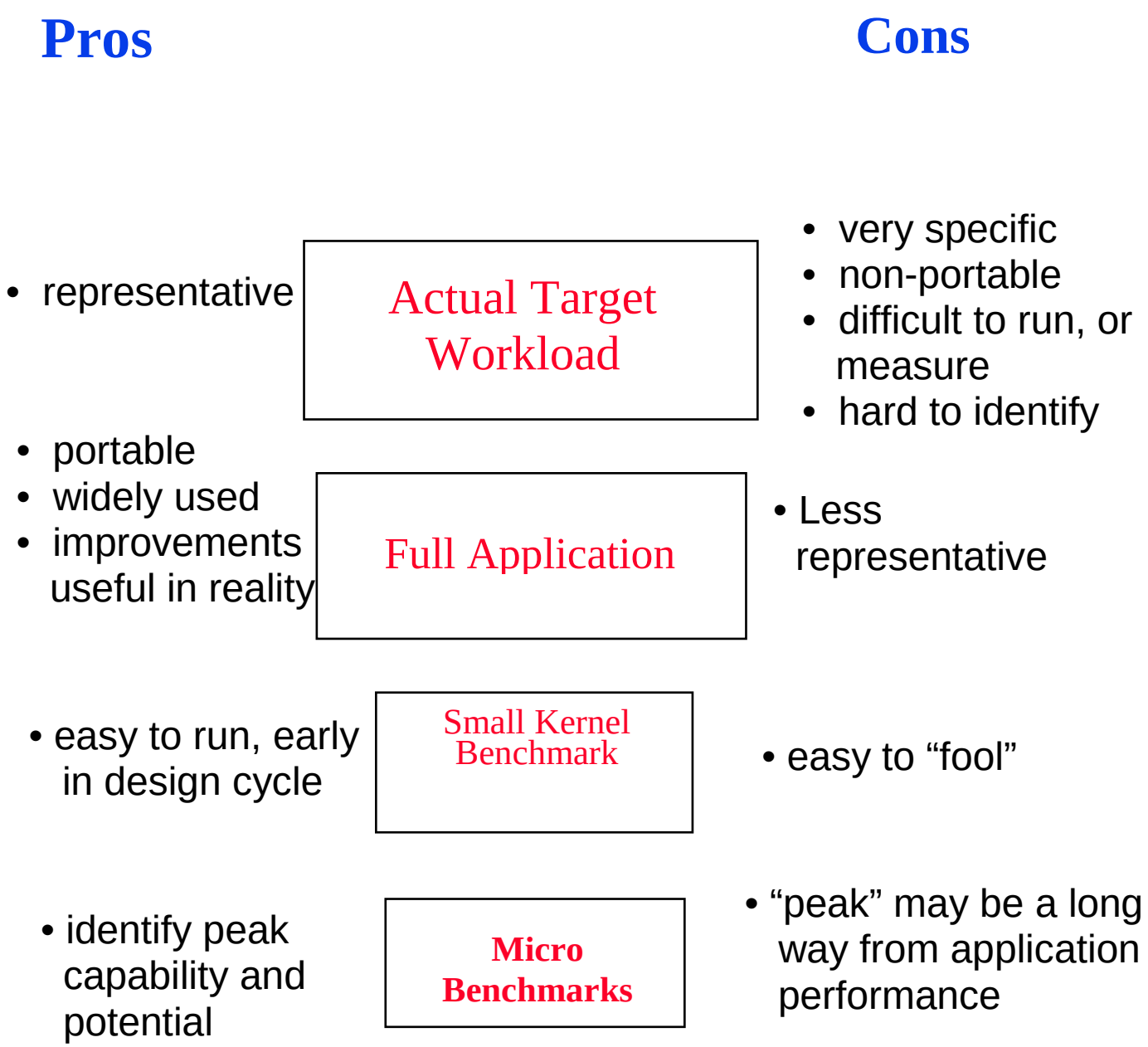
$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

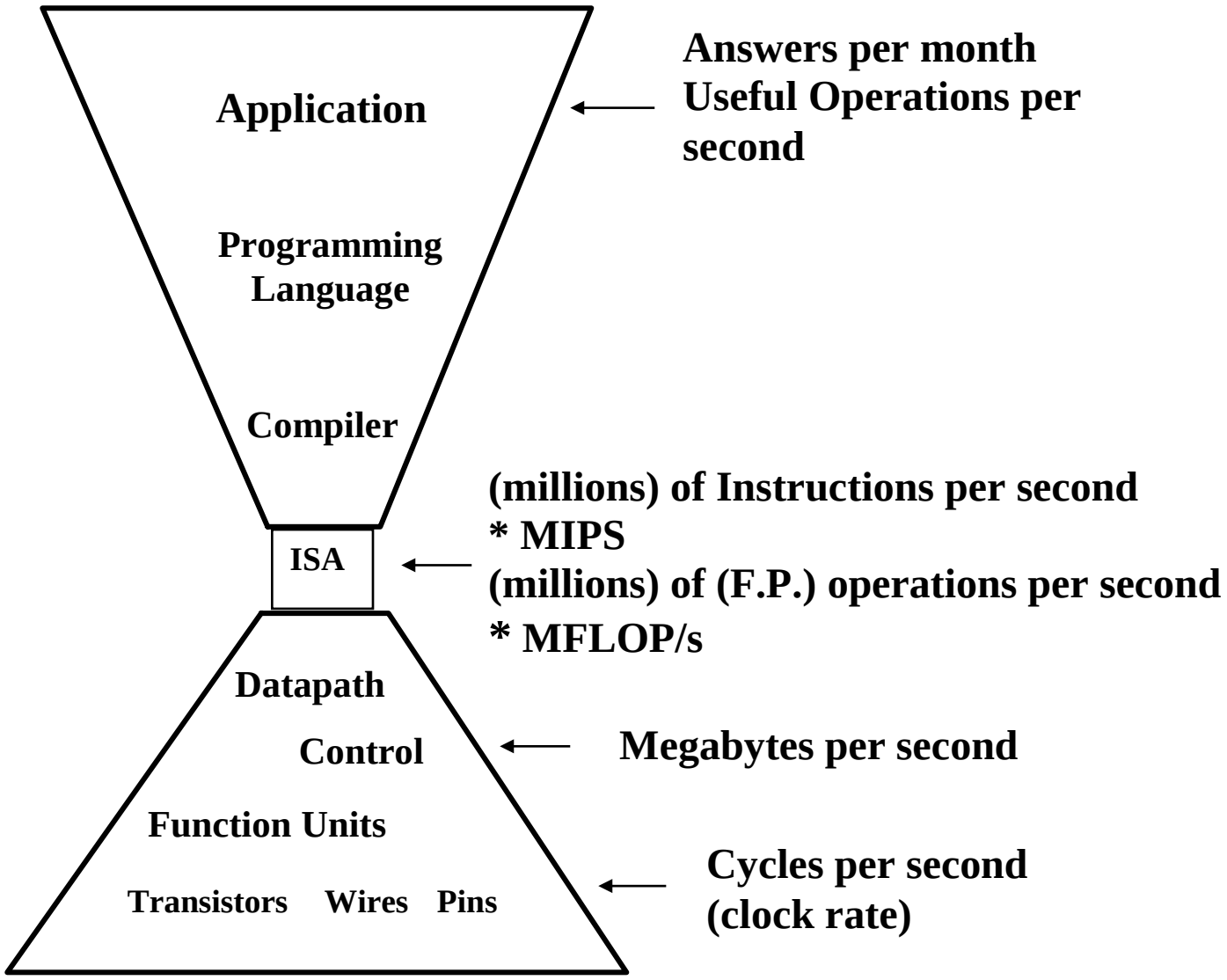
High cache-miss rates

Processor Evaluation Basis



Performance Metrics

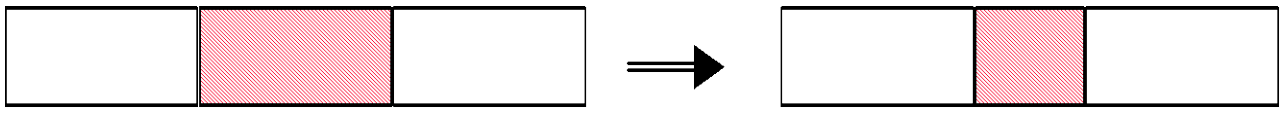
Each metric has a place and a purpose, and each can be misused



Amdahl's Law

Speedup due to enhancement E:

$$\textbf{Speedup(E)} = \frac{\text{ExeTime w/o E}}{\text{ExeTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$



Suppose that enhancement **E** accelerates a fraction **F** of the task by a factor **S** and the remainder of the task is unaffected then,

ExTime(with E) Š

= ((1-F) +F/S) * ExTime(without E)

Speedup(with E) Š = 1/ ((1-F) + F/S)

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?"

Amdahl's Law

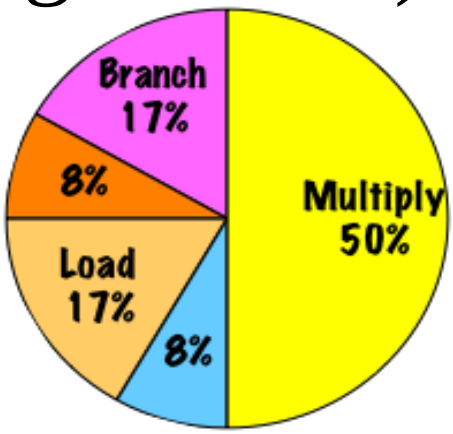
Example

Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

Amdahl's Law
(of Diminishing Returns)

Where a program spends its time during execution



If enhancement "E" speeds up multiply, but other instructions are unchanged, what is the maximum speedup S?

Speedup(with E) Š = 1/ ((1-F) + F/S)

Speedup(with E) Š = 1/ ((1-0.5) + 0.5/Max)

=

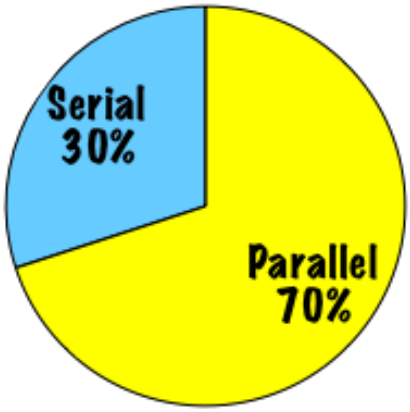
=

=

What is the lesson of Amdahl's Law?

Enhancement by Multiple CPUs

Program We Wish to Run on ***n*** CPUs



The program spends 30% of its time running code that can not be recoded to run in parallel.

Compute speedup for N = 2, 3, 4, 5, and ∞
Speedup(with E) Š = 1/ ((1-F) + F/S)
Speedup(with E) Š = 1/ ((1-0.7) + 0.7/2)
Speedup(with E) Š =

= 1.54

CPUs	2	3	4	5	∞
Speedup	1.54				

Experimental Example

Phone a major computer retailer like Dell or MDG and tell them you are having trouble deciding between two different computers, specifically you are confused about the processors strengths and weaknesses

e.g.,
(Pentium 4 at 2Ghz *vs.*
Celeron M at 1.4 Ghz)

- What kind of responses are you likely to get?
- What kind of response could you give a friend with the same question?

Points to Remember

Performance is specific to particular program(s)

- Execution time is a consistent summary of performance.

For a given architecture, performance increases due to:

- Increases in clock rate (without adverse CPI)
- Improvements in processor organization for lowering CPI.
- Compiler enhancements that lower CPI and/or instruction count.