CENG 3420
Computer Organization and Design

Lecture 04: Performance

Bei Yu

The Chinese University of Hong Kong

Performance Metrics

- ❑ Purchasing perspective
  - ● given a collection of machines, which has the
    - best performance ?
    - least cost ?
    - best cost/performance?

- ❑ Design perspective
  - ● faced with design options, which has the
    - best performance improvement ?
    - least cost ?
    - best cost/performance?

- ❑ Both require
  - ● basis for comparison
  - ● metric for evaluation

- ❑ Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

Throughput versus Response Time

❑ Response time (execution time) – the time between the start and the completion of a task

- Important to individual users

❑ Throughput (bandwidth) – the total amount of work done in a given time

- Important to data center managers

❑ Will need different performance metrics as well as a different set of applications to benchmark embedded and desktop computers, which are more focused on response time, versus servers, which are more focused on throughput
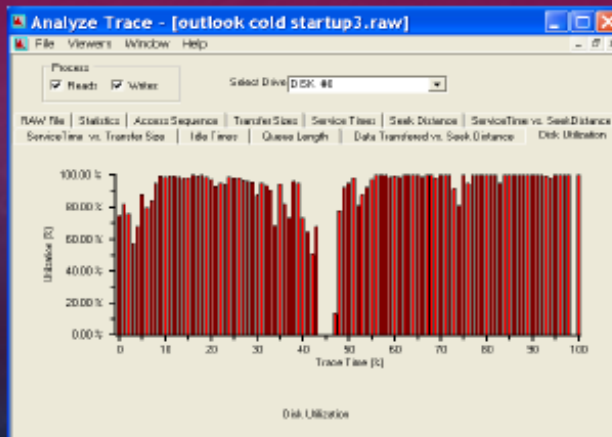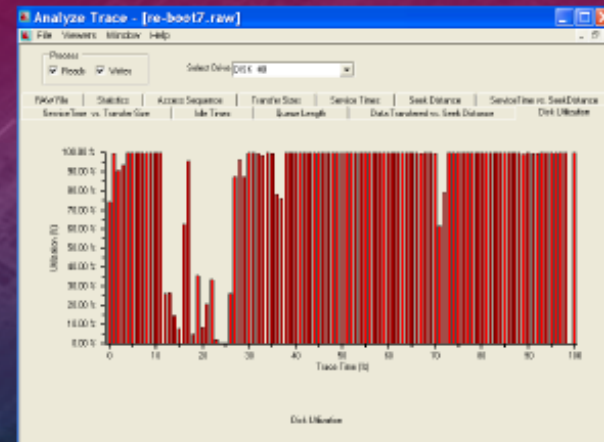
Defining (Speed) Performance

❑ To maximize performance, need to minimize execution time

$$performance_X = 1 / execution\_time_X$$

If X is n times faster than Y, then

$$\frac{performance_X}{performance_Y} = \frac{execution\_time_Y}{execution\_time_X} = n$$

❑ Decreasing response time almost always improves throughput

A Relative Performance Example

❑ If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

Relative Performance Example

❑ If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times faster than B if

$$\frac{performance_A}{performance_B} = \frac{execution\_time_B}{execution\_time_A} = n$$

The performance ratio is $\frac{15}{10} = 1.5$

So A is 1.5 times faster than B

Performance Factors

❑ CPU execution time (CPU time) – time the CPU spends working on a task

- Does not include time waiting for I/O or running other programs

$$\text{CPU execution time for a program} = \text{\# CPU clock cycles for a program} \times \text{clock cycle time}$$

or
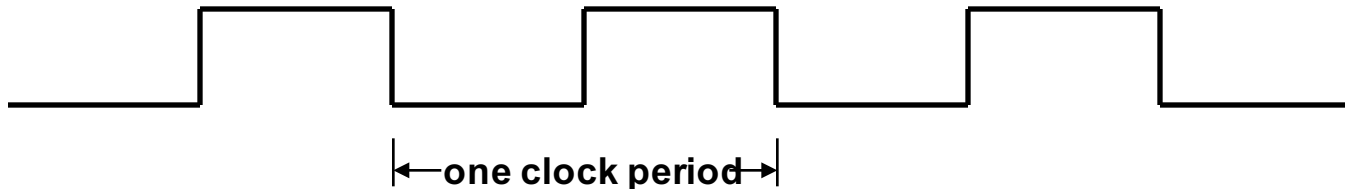
$$\text{CPU execution time for a program} = \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

❑ Can improve performance by reducing either the length of the clock cycle or the number of clock cycles required for a program

Review: Machine Clock Rate

❑ Clock rate (clock cycles per second in MHz or GHz) is inverse of clock cycle time (clock period)

$$CC = 1 / CR$$

←one clock period→

10 nsec clock cycle => 100 MHz clock rate

5 nsec clock cycle => 200 MHz clock rate

2 nsec clock cycle => 500 MHz clock rate

1 nsec (10^{-9}) clock cycle => 1 GHz (10^9) clock rate

500 psec clock cycle =>   2 GHz clock rate

250 psec clock cycle =>   4 GHz clock rate

200 psec clock cycle =>   5 GHz clock rate

Improving Performance Example

❑ A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must a computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

Improving Performance Example

❑ A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must a computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{clock rate}_A}$$

$$\text{CPU clock cycles}_A = 10 \text{ sec} \times 2 \times 10^9 \text{ cycles/sec}$$
$$= 20 \times 10^9 \text{ cycles}$$

$$\text{CPU time}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{clock rate}_B}$$

$$\text{clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = 4 \text{ GHz}$$

Clock Cycles per Instruction

❑ Not all instructions take the same amount of time to execute

- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

$$\begin{matrix} \text{\# CPU clock cycles} \\ \text{for a program} \end{matrix} = \begin{matrix} \text{\# Instructions} \\ \text{for a program} \end{matrix} \text{ x } \begin{matrix} \text{Average clock cycles} \\ \text{per instruction} \end{matrix}$$

❑ Clock cycles per instruction (CPI) – the average number of clock cycles each instruction takes to execute

- A way to compare two different implementations of the same ISA

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

Effective (Average) CPI

❑ Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging

$$\text{Overall effective CPI} = \sum_{i=1}^{n} CPI_i \times IC_i$$

- Where IC_i is the percentage of the number of instructions of class i executed
- CPI_i is the (average) number of clock cycles per instruction for that instruction class
- n is the number of instruction classes

❑ The overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs

Using the Performance Equation

❑ Computers A and B implement the same ISA. Computer A has a clock cycle time of 250 ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500 ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?

Using the Performance Equation

❑ Computers A and B implement the same ISA. Computer A has a clock cycle time of 250 ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500 ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?

Each computer executes the same number of instructions, I, so

$$\text{CPU time}_A = I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}$$

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

Clearly, A is faster … by the ratio of execution times

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{execution\_time}_B}{\text{execution\_time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

THE Performance Equation

❑ Our basic performance equation is then

CPU time = Instruction_count x CPI x clock_cycle

or

$$\text{CPU time} = \frac{\text{Instruction\_count} \ x \ \text{CPI}}{\text{clock\_rate}}$$

❑ These equations separate the three key factors that affect performance

- Can measure the CPU execution time by running the program

- The clock rate is usually given

- Can measure overall instruction count by using profilers/ simulators without knowing all of the implementation details

- CPI varies by instruction type and ISA implementation for which we must know the implementation details

Determinates of CPU Performance

CPU time   = Instruction_count x CPI x   clock_cycle

	Instruction_count	CPI	clock_cycle
Algorithm			
Programming language			
Compiler			
ISA			
Core organization			
Technology			

Determinates of CPU Performance

CPU time = Instruction_count x CPI x clock_cycle

	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

A Simple Example

Op	Freq	CPI_i	Freq x CPI_i
ALU	50%	1	.
Load	20%	5	
Store	10%	3	
Branch	20%	2	
			$\sum =$

❏ How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

❏ How does this compare with using branch prediction to shave a cycle off the branch time?

❏ What if two ALU instructions could be executed at once?

A Simple Example

Op	Freq	CPI_i	$Freq \times CPI_i$			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			$\sum =$ 2.2	1.6	2.0	1.95

❑ How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

   CPU time new = 1.6 x IC x CC   so   2.2/1.6 means 37.5% faster

❑ How does this compare with using branch prediction to shave a cycle off the branch time?

   CPU time new = 2.0 x IC x CC   so   2.2/2.0 means 10% faster

❑ What if two ALU instructions could be executed at once?

   CPU time new = 1.95 x IC x CC   so   2.2/1.95 means 12.8% faster

Workloads and Benchmarks

❑ Benchmarks – a set of programs that form a "workload" specifically chosen to measure performance

❑ SPEC (System Performance Evaluation Cooperative) creates standard sets of benchmarks starting with SPEC89. The latest is SPEC CPU2006 which consists of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006).

<p style="text-align:center">www.spec.org</p>

❑ There are also benchmark collections for power workloads (SPECpower_ssj2008), for mail workloads (SPECmail2008), for multimedia workloads (mediabench), …

Old SPEC Benchmarks

Integer benchmarks		FP benchmarks	
gzip	compression	wupwise	Quantum chromodynamics
vpr	FPGA place & route	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D fields
mcf	Combinatorial optimization	applu	Parabolic/elliptic pde
crafty	Chess program	mesa	3D graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition (NN)
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory interpreter	facerec	Facial image recognition
vortex	Object oriented database	ammp	Computational chemistry
bzip2	compression	lucas	Primality testing
twolf	Circuit place & route	fma3d	Crash simulation fem
		sixtrack	Nuclear physics accel
		apsi	Pollutant distribution

SPEC CINT2006 on Barcelona (CC = 0.4 x 10^9)

Name	ICx10^9	CPI	ExTime	RefTime	SPEC ratio
perl	2,1118	0.75	637	9,770	15.3
bzip2	2,389	0.85	817	9,650	11.8
gcc	1,050	1.72	724	8,050	11.1
mcf	336	10.00	1,345	9,120	6.8
go	1,658	1.09	721	10,490	14.6
hmmer	2,783	0.80	890	9,330	10.5
sjeng	2,176	0.96	837	12,100	14.5
libquantum	1,623	1.61	1,047	20,720	19.8
h264avc	3,102	0.80	993	22,130	22.3
omnetpp	587	2.94	690	6,250	9.1
astar	1,082	1.79	773	7,020	9.1
xalancbmk	1,058	2.70	1,143	6,900	6.0
Geometric Mean					11.7

Comparing and Summarizing Performance

❑ How do we summarize the performance for benchmark set with a single number?

- First the execution times are normalized given the "SPEC ratio" (bigger is faster, i.e., SPEC ratio is the inverse of execution time)

- The SPEC ratios are then "averaged" using the geometric mean (GM)

$$GM = \sqrt[n]{\prod_{i=1}^{n} SPEC\ ratio_i}$$

❑ Guiding principle in reporting performance measurements is reproducibility – list everything another experimenter would need to duplicate the experiment (version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

Other Performance Metrics

❑ Power consumption – especially in the embedded market where battery life is important

● For power-limited applications, the most important metric is energy efficiency