SECR2033
Computer Organization and Architecture

Module 8
Performance Measurement and Analysis

Objectives:

- To understand the key performance issues that relate to computer design.
- Explain the reasons for the move to multicore organization, and understand the trade-off between cache and processor resources on a single chip.
- To summarize some of the issues in computer performance assessment.
- To discuss the benchmarks in general.

2

Module 8
Performance Measurement and Analysis

3

8.1 Introduction

8

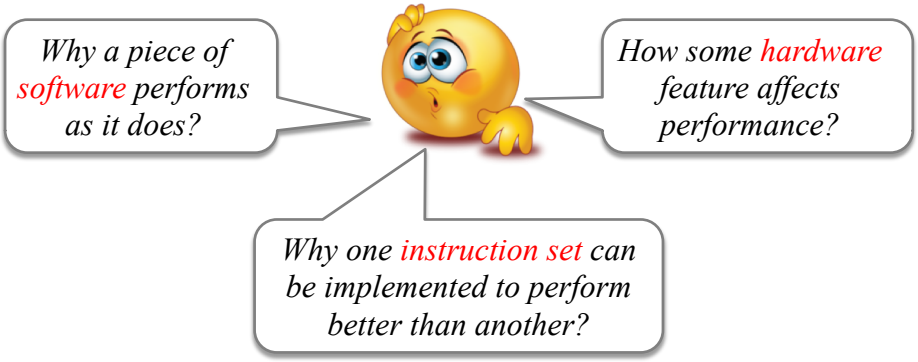
- Hardware performance is often key to the effectiveness of an entire system of _____ and _____.
- For different types of applications, different performance metrics may by appropriate, and different aspects of a computer systems may be the most significant factor in determining overall performance.

- Understanding how best to measure performance and limitations of performance is important when selecting a computer system.

4

8

- Assessing the performance of computers can be quite challenging, together with the wide range of performance improvement techniques employed by hardware designers, have made performance assessment much more difficult.
- To understand the issues of assessing performance:



Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.28. 5

Module 8
Performance Measurement and Analysis

- Overview
- Throughput and Response Time
- Relative Performance
- Performance Metrics

6

8

Overview

- When we say one computer has better performance than another, what do we mean?
- Although this question might seem simple, an analogy with passenger airplanes shows how subtle the question of performance can be (next slide).

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.29. 7

Which of these airplanes has the best performance?

The rate at which the airplane transports passengers, which is the capacity times the cruising speed.

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

- Considering different measures of performance, we see that :
  - the plane with the highest cruising speed was the Concorde (retired from service in 2003),
  - the plane with the longest range is the Douglas DC-8-50,
  - the plane with the largest capacity is the Boeing 747.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.29. 8

- Let's suppose we define performance in terms of speed.
- Two possible definitions.

Performance based on _____:	Performance based on _____:
You could define the fastest plane as the one with the highest cruising speed, taking a single passenger from one point to another in the least time. **Concord**	If you were interested in transporting 450 passengers from one point to another, in the fastest time (as the last column of the figure shows). **Boeing 747**

- Similarly, we can define computer performance in several different ways.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.29. 9

Throughput & Response Time

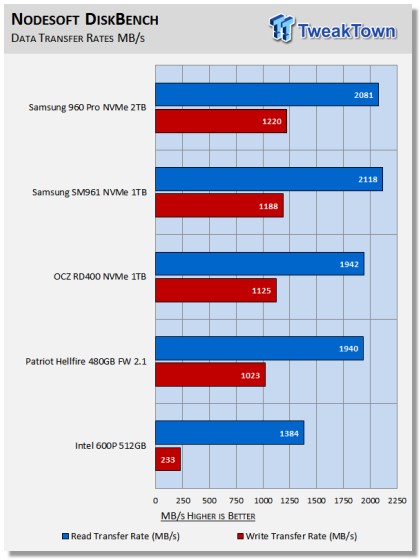
- We might interested in:

reducing *response time* the time between the start and completion of a task. → also referred to as _____.	increasing *throughput* or *bandwidth* → the total amount of work done in a given time.

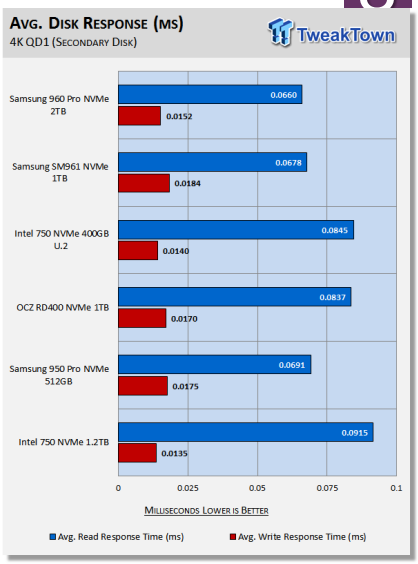
- Hence, in most cases, we need different performance metrics and sets of applications to benchmark personal mobile devices, which are more focused on _____, versus servers, which are more focused on _____.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.30. 10

- Computer performance measures:



(a) Throughput



(b) Response Time

https://www.tweaktown.com/articles/6008/toshiba-q-series-pro-256gb-raid-0-ssd-report/index.html 11

Example 1:

Do the following changes to a computer system increase throughput, decrease response time, or both?

(a) Replacing the processor in a computer with a faster version.

(b) Adding additional processors to a system that uses multiple processors for separate tasks, for example, searching the web

Solution :

(a) Decreasing response time almost always improves throughput. Hence, both response time and throughput are improved.

(b) No one task gets work done faster, so only throughput increases.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.30. 12

Relative Performance

- To maximize performance, the response time or execution time for some task should be minimized.
- Thus, the relationship between performance and execution time for a computer X:

$$Performance_X = \frac{1}{Execution\ time_X}$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.30. 13

- This means that for two computers X and Y, if the performance of X is greater than the performance of Y, we have:

$$Performance_X > Performance_Y$$
$$\frac{1}{Execution\ time_X} > \frac{1}{Execution\ time_Y}$$
$$Execution\ time_Y > Execution\ time_X$$

The execution time on Y is longer than X, if X is faster than Y.

- If X is n times as fast as Y, then the execution time on Y is n times as long as it is on X:

$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.31. 14

8

Example 2:

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?

Solution :

We know that A is n times as fast as B if:

$$\frac{Performance_A}{Performance_B} = \frac{Execution\ time_B}{Execution\ time_A} = n$$

A is therefore _____ times as fast as B.

Thus the performance ratio is $n = \frac{Execution\ time_B}{Execution\ time_A} =$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.31. 15

8

Performance Metrics

MB/s, Mb/s : Megabytes, Megabits Per Second

MIPS : _____

CPI : _____

IPC : Instructions Per Clock cycle

Hz : (processor clock frequency) cycles Per Second

LIPS : Logical Interference Per Second

FLOPS : Floating-Point arithmetic Operations Per Second

16

Module 8
Performance Measurement and Analysis

8.1 Introduction
8.2 Defining Performance
8.3 Basic Measures of Computer Performance
8.4 Comparing Performance
8.5 Benchmarks
8.6 Summary

- Overview
- Clock Speed
- Instruction Execution Rate
- CPU Performance
- Classic CPU – Instruction Count

17

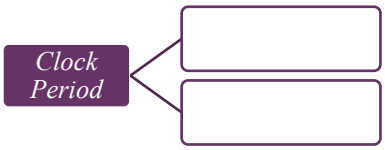
8 Overview

- In evaluating processor hardware and setting requirements for new systems, _____ is one of the key parameters to consider, along with cost, size, security, reliability, and, in some cases, power consumption.
- It is difficult to make meaningful performance comparisons among different processors, even among processors in the same family.
- Next we look at some traditional measures of processor speed.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.56. 18

8 Clock Speed

- *Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and so on, are governed by a system clock.
- These discrete time intervals are called _____. (or *ticks*, *clock ticks*, *clock periods*, *clocks, cycles*).
- *Clock period* is the length of each *clock cycle*, or *clock rate*, which inverse of the clock period.



(*e.g.*, 250 picoseconds, or 250 ps)

(*e.g.*, 4 Gigahertz, or 4 GHz),

* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.57.
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.33. 19

8 Instruction Execution Rate

- A processor is driven by a clock with a constant _____, f.
- Define the instruction count, I_c, for a program as the number of machine instructions executed for that program.
- An important parameter is the average *Cycles Per Instruction* (*CPI*) for a program.

- If all instructions required the same number of clock cycles, then *CPI* would be a constant value for a processor.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.58. 20

8

- However, on any given processor, the number of clock cycles required varies for different types of instructions, such as *load*, *store*, *branch*, and so on.
- Let CPI_i be the number of cycles required for instruction type i, and I_i be the number of executed instructions of type i for a given program. Then we can calculate an overall *CPI* as :

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

Note: I_c *is number of instruction executions, not the number of instructions in the object code of the program.*

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.58. 21

CPI (Cycles Per Instruction)

8

- A common measure of performance for a processor is the rate at which instructions are executed, expressed as *Millions of Instructions Per Second* (*MIPS*), referred to as the _____.
- We can express the MIPS rate in terms of the *clock rate* and *CPI* as follows:

$$MIPS\ rate = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

$T \rightarrow$ *Processor time*

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.58. 22

CPI (Cycles Per Instruction)

8

Example 3:

Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the *CPI* for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	*CPI*	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

Calculate the corresponding *MIPS rate* for the processor.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.59. 23

Instruction Type	*CPI*	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

CPI (Cycles Per Instruction)
MIPS (Millions of Instructions Per Second)

Solution :

The average *CPI* when the program is executed on a uniprocessor with the above trace results is:

$$CPI = (1 \times 60\%) + (2 \times 18\%) + (4 \times 12\%) + (8 \times 10\%)$$

$$=$$

$$=$$

$$=$$

The corresponding *MIPS* rate is:

$$MIPS\ rate = \frac{f}{CPI \times 10^6} =$$

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.59. 24

8

CPU Performance

- Users and designers often examine performance using different metrics.
- If we could relate these different metrics, we could determine the effect of a design change on the performance as experienced by the user.
- CPU performance as the bottom-line performance measure is *CPU execution time*.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.33. 25

$$Clock\ cycle\ time = \frac{1}{Clock\ rate}$$

- A simple formula relates the most basic metrics (clock cycles and clock cycle time) to CPU time for a program:

$$CPU\ execution\ time = CPU\ clock\ cycles \times Clock\ cycle\ time$$

- Alternatively, because *clock rate* and *clock cycle time* are inverses:

$$CPU\ execution\ time = \frac{CPU\ clock\ cycles}{Clock\ rate}$$

- This formula makes it clear that the performance can be improved by **reducing**:
  - the _____ of clock cycles required for a program, or
  - the _____ of the clock cycle.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34. 26

Example 4: Improving performance

Our favourite program runs in 10 seconds on computer A, which has a 2 GHz clock.

We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds.

The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program.

What *clock rate* should we tell the designer to target?

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34. 27

Solution : Improving performance

(Method 1)

- Let's first find the number of clock cycles required for the program on A:

$$CPU\ time_A = \frac{CPU\ clock\ cycles_A}{Clock\ rate_A}$$

$$CPU\ clock\ cycles_A =$$

$$=$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34. 28

- CPU time for B can be found using this equation:

$$CPU\ time_B = \frac{1.2 \times CPU\ clock\ cycles_A}{Clock\ rate_B}$$

$$Clock\ rate_B =$$

$$=$$

$$=$$

The clock rate we should tell the designer to target for computer B.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34. 29

Solution : Improving performance

(Method 2)

Computer A:	Computer B:
$Execution\ time_A = 10\ seconds$ $Clock\ rate_A = 2\ GHz = 2 \times 10^9\ Hz$	$Execution\ time_B = 6\ seconds$

- Let's first find the number of clock cycles required for the program on A:

$$CPU\ clock\ cycle_A = Execution\ time_A \times Clock\ rate_A$$

$$=$$

$$=$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34. 30

- Clock rate for B can be found using this equation with 1.2 times more clock cycle than A:

$$1.2\,(CPU\ clock\ cycle_A) = Execution\ time_B \times Clock\ rate_B$$

$$Clock\ rate_B =$$

$$=$$

$$=$$

$$=$$

The clock rate we should tell the designer to target for computer B.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.34. 31

CPI (Cycles Per Instruction)

Instruction Performance

- The performance equations previously did not include any reference to the number of instructions needed for the program.
- The execution time of a computer must depend on the number of instructions in a program.
- Therefore, the number of clock cycles required for a program can be written as:

$$CPU\ clock\ cycles = \underset{a\ program.}{Instruction\ for} \times \underset{per\ instruction}{Average\ clock\ cycles}$$

$$CPU\ clock\ cycles = I \times CPI$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.35. 32

Example 5: Using performance equation

Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program.

Which computer is faster for this program and by how much?

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.35. 33

Solution : Using performance equation

- We know that each computer executes the same number of instructions for the program; let's call this number I.
- First, find the number of processor clock cycles for each computer:

$$CPU\ clock\ cycle_A = I \times 2.0$$
$$CPU\ clock\ cycle_B = I \times 1.2$$

- Now we can compute the CPU time for each computer:

$$CPU\ time_A = CPU\ clock\ cycle_A \times Clock\ cycle\ time_A$$
$$=$$
$$=$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.36. 34

- The CPU time for computer B:

$$\begin{aligned} CPU\ time_B &= CPU\ clock\ cycle_B \times Clock\ cycle\ time_B \\ &= (I \times 1.2) \times 500ps \\ &= 600 \times I\ ps \end{aligned}$$

- Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{CPU\ performance_A}{CPU\ performance_B} = \frac{Execution\ timeB}{Execution\ timeA}$$

$$=$$

$$=$$

Computer A is ___ times as fast as computer B for this program.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.36. 35

CPI (Cycles Per Instruction)

Classic CPU – Instruction Count

- We can now write this basic performance equation in terms of instruction count, I_i (the number of instructions executed by the program), CPI, and clock cycle time:

$$CPU\ time = Instruction\ count \times CPI \times Clock\ cycle\ time$$

- or, since the clock rate is the inverse of clock cycle time:

$$CPU\ time = \frac{Instruction\ count \times CPI}{Clock\ rate}$$

The three key factors that affect performance.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.36. 36

Example 6: Comparing code segments

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers has supplied the following facts:

	CPI for each instruction class		
CPI	1	2	3

The compiler writer is considering two code sequences the require the following instruction counts:

Code Sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

(a) Which code sequence execute the most instructions?
(b) Which will be faster?
(c) What is the CPI for each sequence?

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.37. 37

Solution : Comparing code segments

Code sequence 2 executes the most instruction count.

(a) $Code\ sequence_1 = 2 + 1 + 2 = 5\ instructions\ count_1$

$Code\ sequence_2 = 4 + 1 + 1 = 6\ instructions\ count_2$

(b) Get the CPU clock cycle for each code sequence using the equation:

$$CPU\ clock\ cycle = \sum_{i=1}^{n} (CPI_i \times I_i)$$

Code sequence 2 executes the instructions faster.

$CPU\ clock\ cycle_1 =$

$=$

$CPU\ clock\ cycle_2 =$

$=$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.37. 38

(c) The *CPI* values for each code sequence can be computed by:

$$CPI = \frac{CPU\ clock\ cycle}{Instruction\ count}$$

$$CPI_1 = \frac{CPU\ clock\ cycle_1}{Instruction\ count_1} =$$

$$CPI_2 = \frac{CPU\ clock\ cycle_2}{Instruction\ count_2} =$$

Since code sequence 2 takes fewer overall clock cycle but more instructions, it must has a lower CPI .

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.37. 39

40

The BIG Picture

- The table shows the basic measurement at different levels in the computer and what is being measured in each case.

Components of performance	Unit of measure
CPU execution time	Seconds for the program.
Instruction count	Instructions executed for the program.
Clock cycle per instruction (CPI)	Average number of clock cycles per instruction.
Clock cycle time	Seconds per clock cycle.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.38. 41

The BIG Picture

- We can see how these factors are combined to yield execution time in seconds per program:

$$Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} \times \frac{Clock\ cycles}{Instruction} \times \frac{Seconds}{Clock\ cycle}$$

- Lower instruction count → slower clock cycle time or higher CPI

CPI depends on type of instructions executed, less instructions may not be the fastest.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.38. 42

Module 8
Performance Measurement and Analysis

43

8

Overview

- Computer performance assessment is a quantitative science.
- Mathematical and statistical tools give us many ways in which to rate the overall performance of a system and the performance of its constituent components.
- There are so many ways to quantify system performance.
- In this section, we describe the most common measures of "average" computer performance and then provide situations where the use of each is appropriate and inappropriate.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture*. United States: Jones and Bartlett Publishers. p.453. 44

- In comparing the performance of two systems, we measure the _____ that it takes for each system to perform the same amount of work.
- If the same program is run on two different systems, System A and System B:
  - System A is *n times as fast* as System B if:

$$\frac{Running\ time\ on\ system\ B}{Running\ time\ on\ system\ A} = n$$

  - System A is *x% faster* than System B if:

$$\left(\frac{Running\ time\ on\ system\ B}{Running\ time\ on\ system\ A} - 1\right) \times 100 = x$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.453. 45

46

Example 7:

Consider the performance of two race cars. Car A completes a 10-mile run in 3 minutes, and car B completes the same 10-mile course in 4 minutes.

(a) How many time car A fast as car B?

(b) What is % car A faster than car B?

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.453. 47

Solution :

Car A is _____ times as fast as Car B.

(a) How many time car A fast as car B?

$$\frac{Time\ for\ Car\ B\ to\ travel\ 10\ miles}{Time\ for\ Car\ A\ to\ travel\ 10\ miles} =$$

(b) What is % car A faster than car B?

$$= \left(\frac{Time\ for\ Car\ B\ to\ travel\ 10\ miles}{Time\ for\ Car\ A\ to\ travel\ 10\ miles} - 1 \right) \times 100$$

$=$

$=$

$=$

Car A is ____% faster than Car B.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.453. 48

Average Performance

- The previous formulas are useful in comparing the average performance of one system with the average performance of another.

- When we are evaluating system performance, we are most interested in its expected performance under a given workload.
- We use _____, forming a concise *measures of central tendency*.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.453. 49

(a) The Arithmetic Mean

- The *arithmetic mean* (average) is the one with which everyone is most familiar.
- If we have n measurements, and we add them together and divide by n, then the result is the *arithmetic mean* (average), given by:

$$\frac{\sum_{i=1}^{n} x_i}{n}$$

- The arithmetic mean can be misleading if the data are skewed or scattered.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.454. 50

8

Example 8:

Consider the arithmetic average running time in seconds of five programs (*v, w, x, y, z*) on three systems (*A, B, C*) in the table below.

Program	System A Execution Time	System B Execution Time	System C Execution Time
v	50	100	500
w	200	400	600
x	250	500	500
y	400	800	800
z	5000	4100	3500
Average	1180	1180	1180

The performance differences are hidden by the simple average.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.454. 51

Example 9:

If execution frequencies (expected workloads) are known, a *weighted average* can be revealing. The running times for System *A* and System *C* are restated in the following table.

- Weighted average for System *A* =

Program	Execution Frequency	System A Execution Time	System C Execution Time
v	50%	50	500
w	30%	200	600
x	10%	250	500
y	5%	400	800
z	5%	5000	3500
Weighted Average		380 seconds	695 seconds

- Similar weighted average calculation for System *C*.
- System *A* is about 83% faster than System *C* for this particular workload.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.454-455. 52

$$= \left(\frac{Weighted\ Average\ System\ C}{Weighted\ Average\ System\ A} - 1 \right) \times 100$$

Program	Execution Frequency	System A Execution Time	System C Execution Time
v	50%	50	500
w	30%	200	600
x	10%	250	500
y	5%	400	800
z	5%	5000	3500
Weighted Average		380 seconds	695 seconds

- System A is about 83% faster than System C for this particular workload.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.454-455. 53

- However, workloads can change over time.
- A system optimized for one workload may perform poorly when the workload changes (Execution Frequency), as illustrated in Example 10 below:

Example 10:

Program	System A Execution Time	Execution Frequency #1	Execution Frequency #2
v	50	50%	25%
w	200	30%	5%
x	250	10%	10%
y	400	5%	5%
z	5000	5%	55%
Weighted Average		380 seconds	2817.5 seconds

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.455. 54

(b) The Geometric Mean

(_____)

- The previous discussion cannot use the arithmetic mean if the measurements exhibit a great deal of variability.
- The *geometric mean* gives us a consistent number with which to perform comparisons regardless of the distribution of the data.
- The geometric mean, G is defined as the nth root of the product, x_i of the n measurements, and represented by the following formula:

$$G = \sqrt[n]{x_1 \times x_2 \times x_3 \times \ldots \times x_n}$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.456. 55

- Unlike the *arithmetic means*, the *geometric mean* does not give us a real expectation of system performance. It serves only as a tool for comparison.
- The geometric mean is more helpful than the arithmetic average when comparing the relative performance of two systems.
- The systems under evaluation are _____ to the reference machine when we take the ratio of the run time of a program on the reference machine to the run time of the same program on the system being evaluated.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.456. 56

Example 11: System A and System C normalized to System B.

Given the geometric means for a sample of five programs, found by taking the 5th root of the products of the normalized execution times for each system.

Program	System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
v	50	2	100	1	500	0.2
w	200	2	400	1	600	0.6667
x	250	2	500	1	500	1
y	400	2	800	1	800	1
z	5000	0.82	4100	1	3500	1.1714
Geometric Mean		1.6733		1		0.6898

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.456. 57

- The *geometric means* for System A and System C normalized to System B are calculated as follows:

$$G_A = \sqrt[5]{\frac{100}{50} \times \frac{400}{200} \times \frac{500}{250} \times \frac{800}{400} \times \frac{4100}{5000}}$$
$$=$$
$$=$$
$$=$$

$$G_C = \sqrt[5]{\frac{100}{500} \times \frac{400}{600} \times \frac{500}{500} \times \frac{800}{800} \times \frac{4100}{3500}}$$
$$=$$
$$=$$
$$=$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.456. 58

Example 12: System A and System B normalized to System C.

When another system is used for a reference machine, we get a different set of numbers.

Program	System A Execution Time	Execution Time Normalized to C	System B Execution Time	Execution Time Normalized to C	System C Execution Time	Execution Time Normalized to C
v	50	10	100	5	500	1
w	200	3	400	1.5	600	1
x	250	2	500	1	500	1
y	400	2	800	1	800	1
z	5000	0.7	4100	0.8537	3500	1
Geometric Mean		2.4258		1.4497		1

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.457. 59

- The *geometric means* for System A and System B normalized to System C are calculated as follows:

$$G_A = \sqrt[5]{\frac{500}{50} \times \frac{600}{200} \times \frac{500}{250} \times \frac{800}{400} \times \frac{3500}{5000}}$$
$$=$$
$$=$$
$$=$$

$$G_B = \sqrt[5]{\frac{500}{100} \times \frac{600}{400} \times \frac{500}{500} \times \frac{800}{800} \times \frac{3500}{4100}}$$
$$=$$
$$=$$
$$=$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.457. 60

... (Properties)

- One of the nice properties of the *geometric mean* is that we get the same results regardless of which system we pick for a reference.
- Based on previous 2 examples, we can notice that the ratio of the *geometric means* is consistent no matter which system we choose for the reference machine:

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.457. 61

... (Properties)

System → *A* *B* *C*

Program	Execution Time Normalized to B	Execution Time Normalized to B	Execution Time Normalized to B
Geometric Mean	1.6733	1	0.6898

Program	Execution Time Normalized to C	Execution Time Normalized to C	Execution Time Normalized to C
Geometric Mean	2.4258	1.4497	1

Reference machine: *B*

$$\frac{Geometric\ mean\ A}{Geometric\ mean\ B} \approx \frac{Geometric\ mean\ A}{Geometric\ mean\ B}$$

$$\frac{1.6733}{1} \approx \frac{2.4258}{1.4497}$$

Reference machine: *C*

62

Reference machine: *B*

$$\frac{Geometric\ mean\ C}{Geometric\ mean\ B} \approx \frac{0.6898}{1}$$

Reference machine: *C*

$$\frac{Geometric\ mean\ C}{Geometric\ mean\ B} \approx \frac{1}{1.4497}$$

Reference machine: *B*

$$\frac{Geometric\ mean\ A}{Geometric\ mean\ C} \approx \frac{1.6733}{0.6898}$$

Reference machine: *C*

$$\frac{Geometric\ mean\ A}{Geometric\ mean\ C} \approx \frac{2.4258}{1}$$

Reference machine: *B*

Reference machine: *C*

63

Self-test

Get the ratio of the geometric means for system B to A.

Reference machine: *B*

Reference machine: *C*

64

... (Problem with Geometric Mean)

- The inherent problem with using the geometric mean to demonstrate machine performance is that all execution times contribute equally to the result.
- So shortening the execution time of a small program by 10% has the same effect as shortening the execution time of a large program by 10%.
  - Shorter programs are generally easier to optimize, but in the real world, we want to shorten the execution time of longer programs.
- Also, the geometric mean is not proportionate → A system giving a geometric mean 50% smaller than another is not necessarily twice as fast!

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.458.

65

66

Module 8
Performance Measurement and Analysis

8.1 Introduction
8.2 Defining Performance
8.3 Basic Measures of Cor
8.4 Comparing Performanc
8.5 Benchmarking
8.6 Summary

- Overview
- Clock Rate, MIPS and FLOPS
- Synthetic Benchmarks
- SPEC Benchmarks
- TPC Benchmarks
- System Simulation

67

8 Overview

- * _____ is the most common approach to assessing processor and computer system performance.
- *Performance benchmarking* is the science of making objective assessments of the performance of one system over another.
- Benchmarks are also useful for assessing performance improvements obtained by upgrading a computer or its components.

- Good benchmarks:
  - enable to cut through advertising hype and statistical tricks.
  - identify the systems that provide good performance at the most reasonable cost.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture*. United States: Jones and Bartlett Publishers. p.461. 68

ISA (Instruction Set Architecture)
MIPS (Millions of Instructions Per Second)

8

(a) Clock Rate, MIPS, and FLOPS

- CPU speed is a misleading metric that is most often used by computer vendors touting their systems' alleged superiority to all others.
- A widely metric related to clock rate is the _____.

Example 13:

- Saying that System A (1.4 GHz) is faster than System B (900 MHz) is valid only when the ISAs of System A and B are identical.
- With different ISAs, it is possible that both of these systems could obtain identical results within the same amount of wall clock time.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.462. 69

FLOPS (Floating-point Operations Per Second)

8

- There is a similar problem with the $FLOPS$ metric.
- The FLOPS metric is even more difficult than the MIPS metric because there is no agreement as to what constitutes a floating-point operation.
- Despite their shortcomings, clock speed, MIPS, and FLOPS can be useful metrics in comparing relative performance across a line of similar computers offered by the same vendor.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.463. 70

8

(b) Synthetic Benchmarks

(*Whetstone, Linpack, Dhrystone*)

- The resulting execution time would lead to a single performance metric across all of the systems tested → *synthetic benchmarks*, because they don't necessarily represent any particular workload or application.
- Three of the better-known synthetic benchmarks are the _____, _____, and _____ metrics.

These programs are much too small to be useful in evaluating the performance of today's systems.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.464. 71

8

(c) SPEC Benchmarks

(*Standard Performance Evaluation Corporation*)

- The science of computer performance measurement benefited greatly by the contributions of the *Whetstone*, *Linpack*, and *Dhrystone* benchmarks.
- For one thing, these programs gave merit to the idea of having a common standard benchmark suite by which all systems could be compared → *SPEC benchmarks*.
- Their most widely known benchmark suite is the _____.
- The latest version of this benchmark is CPU2000 → consists of two parts, CINT2000, which measures integer arithmetic operations, and CFP2000, which measures floating-point processing.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.465. 72

8 (d) TPC Benchmarks

(_____)

- SPEC's benchmarks are helpful to computer buyers whose principal concern is CPU performance.
- When the performance of the entire system under high transaction loads is a greater concern, the *TPC benchmarks* are more suitable.

- The current version of this suite is the *TPC-C benchmark*.
- TPC-C models the transactions typical of a warehousing and distribution business using terminal emulation software.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.469. 73

8 (e) System Simulation

- The TPC benchmarks differ from the SPEC benchmarks in that they try to simulate a complete computing environment.
- Although the purpose of the TPC benchmarks is to measure performance, their simulated environment may also be useful to predict, and hence optimize, performance under various conditions.

- In general, *simulations* give us tools that we can use to model and predict aspects of system behaviour without the use of the exact live environment that the simulator is _____.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.476. 74

8.6 Summary

- Computer performance assessment relies upon measures of central tendency that include the *arithmetic mean*, *weighted arithmetic mean*, the *geometric mean*, and the *harmonic mean*.
- Each of these is applicable under different circumstances.
- *Benchmark suites* have been designed to provide objective performance assessment.
  - The most well respected of these are the *SPEC* and *TPC* benchmarks.

75

- *CPU performance* depends upon many factors.
  - These include pipelining, parallel execution units, integrated floating-point units, and effective branch prediction.
  - User code optimization affords the greatest opportunity for performance improvement.
  - Code optimization methods include loop manipulation and good algorithm design.

76