Performance

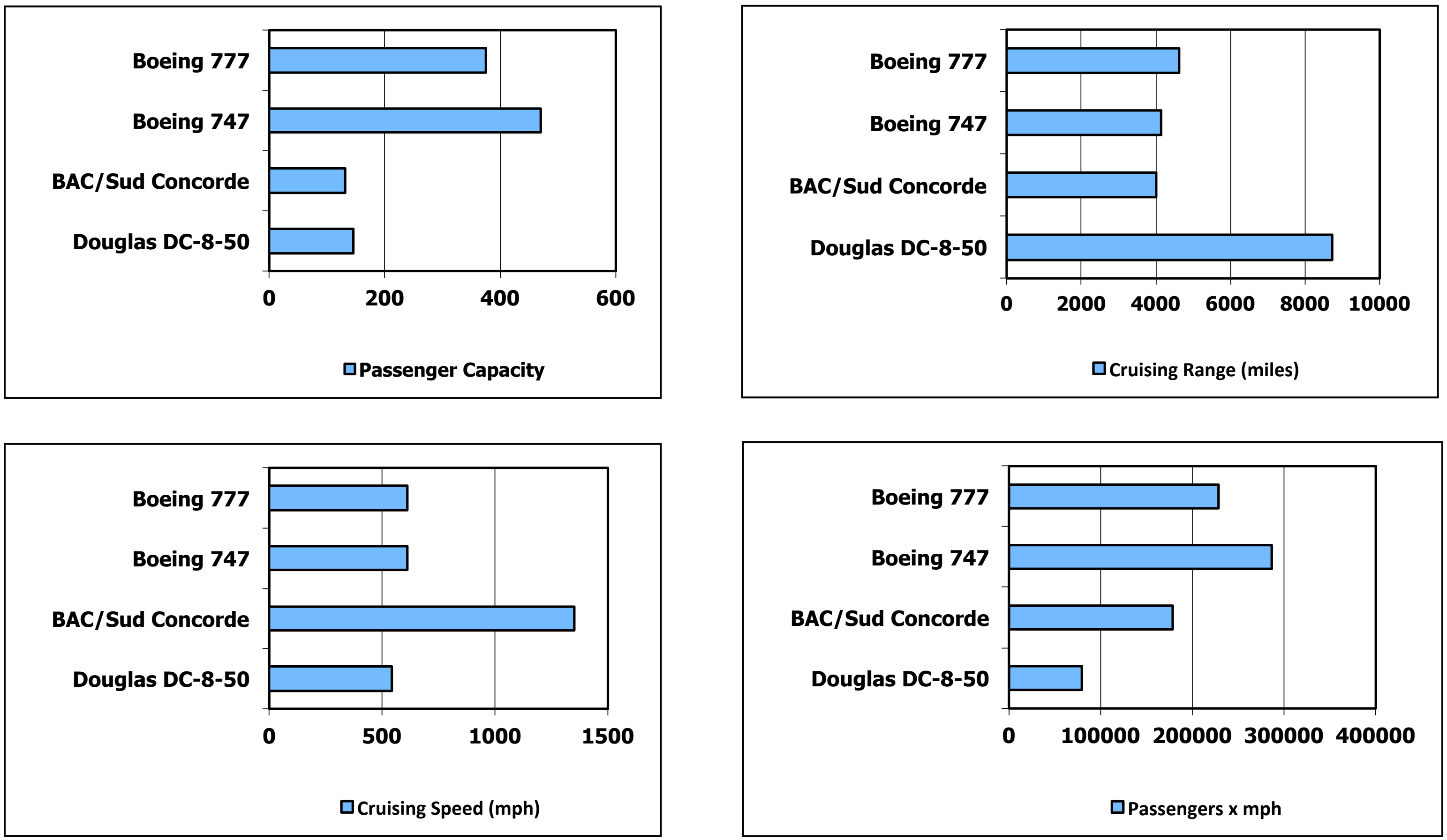
Lecture 19
December 5th, 2017

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: [CS:APP3e] slides from CMU; [COD5e] slides from Elsevier Inc.*

Performance Example

- Question: Which aircraft performs the best??



Today

Textbook: [P&H] 1.6

- **Performance Metrics: Time and Rate**
- **Summarizing Performance**

Performance Metrics #1: Time

- **Wall-clock time, response time, or elapsed time**
  - Actual time from start to completion
  - Includes everything: CPU time for other programs as well as for itself, I/O, operating system overheads, etc

- **CPU (execution) time**
  - CPU time spent for a given program
  - user CPU time + system CPU time
  - e.g., results of UNIX `time` command

```
90.7u  12.9s  2:39  65%
```

Performance Metrics #1: Time

- **Decomposition of CPU (Execution) Time**

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$

$$= \frac{\text{Cycles}}{\text{Program}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

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

This equation is called "Iron Law of CPU Performance."

Performance Metrics #1: Time

- **More on CPI (Clocks or Cycles Per Instruction)**
  - $$\text{CPI} = \frac{\sum_{i=1}^{n} (\text{CPI}_i \times \text{I}_i)}{\text{Instruction Count}}$$
  - CPI Example

Instruction Class	Frequency	CPI_i
ALU operations	43%	1
Loads	21%	2
Stores	12%	2
Branches	24%	2

CPI = 0.43 x 1 + 0.21 x 2 + 0.12 x 2 + 0.24 x 2

Performance Metrics #1: Time

- **Comparing CPIs of two CPUs**
  - Example question: What is the CPI of CPU_S and CPU_Q?

Instruction Type	Instr. count (millions)	Cycles per Instr. (CPI)	
		CPU_S	CPU_Q
Arithmetic & Logic	10	1	1
Load & Store	5	4	2
Branch	4	2	3
Miscellaneous (기타)	1	4	4

$\mathbf{CPI_S}$ = (10 x 1 + 5 x 4 + 4 x 2 + 1 x 4) / (10 + 5 + 4 + 1) = 2.1

$\mathbf{CPI_Q}$ = (10 x 1 + 5 x 2 + 4 x 3 + 1 x 4) / (10 + 5 + 4 + 1) = 1.8

Question: So, CPU_Q always performs better?

Performance Metrics #1: Time

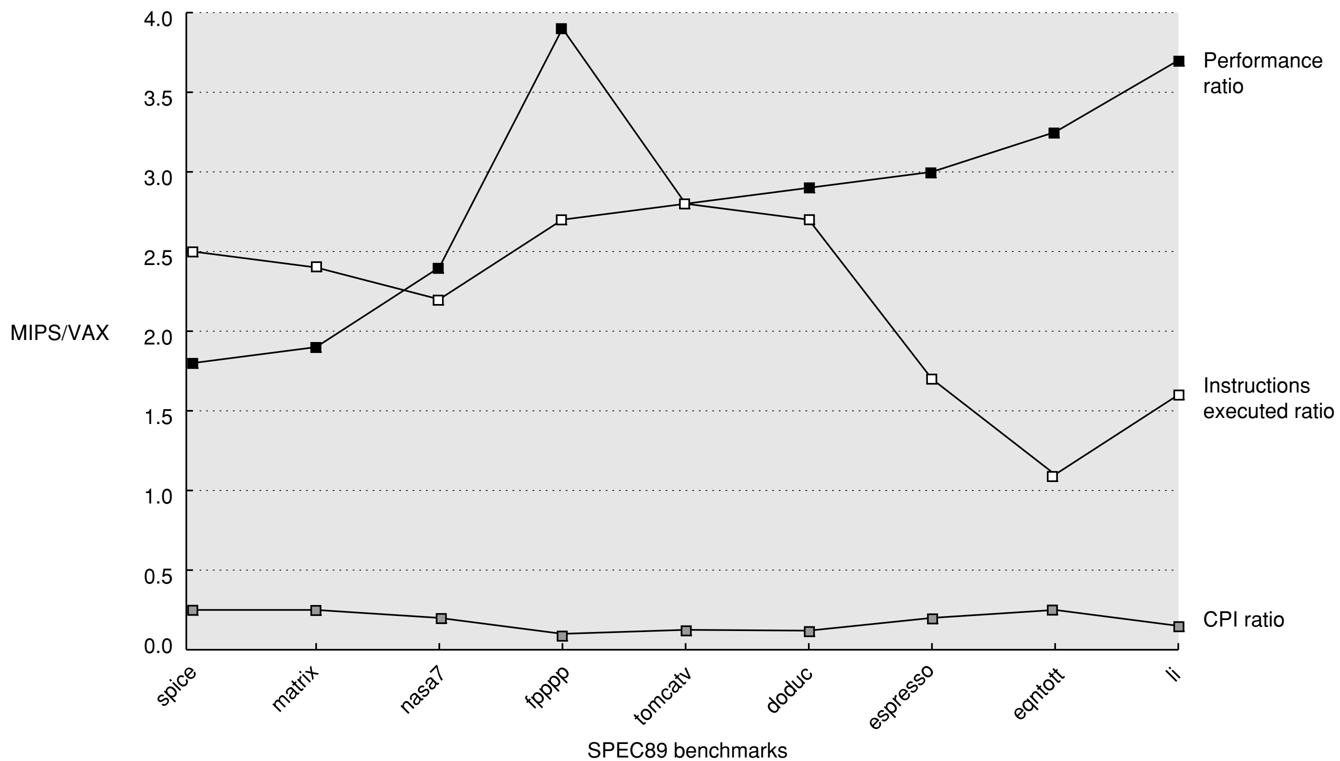
- **Factors involved in the CPU Time**

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

	Instructions / Program	Cycles / Instruction	Seconds / Cycle
Program	v		
Compiler	v		
ISA	v	v	
Organization		v	v
Technology			v

Performance Metrics #1: Time

- **RISC vs. CISC arguments**
  - **MIPS (typical RISC) vs. VAX8700 (typical CISC)**



Source : Hennessy & Patterson *Computer Architecture: A Quantitative Approach, 5th Ed.(Appencix L)*, Morgan Kaufmann, 2012

Performance Metrics #2: Rate

- **MIPS (million instructions per second)**
  - $$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$
  - Specifies performance (roughly) inversely to execution time
  - Easy to understand; faster machines means bigger MIPS
  - Problems
    - It does not take into account the capabilities of the instructions.
    - It varies between programs on the same computer.
    - It can even vary inversely with performance!!

- **MFLOPS (million floating-point operations per second)**

Performance Metrics: Ratio

- "X is n times faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

- "X is n% faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 1 + \frac{n}{100}$$

- "X is n order of magnitude faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 10^n$$

Summarizing Performance

- **Arithmetic mean** (Time) $$\frac{1}{n}\sum_{i=1}^{n} T_i$$
- **Harmonic mean** (Rate) $$\frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}}$$
- **Geometric mean** (Ratio) $$\sqrt[n]{\prod_{i=1}^{n} Ratio_i}$$

Summarizing Performance: Arithmetic Mean

- **Used to summarize performance given in times**
  - Average Execution Time= ($\sum_{i=1}^{n}$ Execution Times) / n
  - Assumes each benchmark is run an equal no. of times

- **Weighted Arithmetic Mean**
  - Weighted Average Execution Time = $\sum_{i=1}^{n}$ (W_i X Execution Times) / $\sum_{i=1}^{n} W_i$
  - One possible weight assignment: equal execution time on some machine

Summarizing Performance: Harmonic Mean

- **Used to summarize performance in rates (e.g., MIPS, FLOPS):**
  - Harmonic Mean = $n \;/\; \sum_{i=1}^{n} (1 / R_i)$
  - Example
    - Four programs execute at 10, 100, 50 and 20 MFLOPS, respectively
    - Harmonic mean is 4 / (1/10 + 1/100 + 1/50 + 1/20) = 22.2 MFLOPS

- **Weighted Harmonic Mean**
  - Weighted Harmonic Mean = $\sum_{i=1}^{n} W_i \;/\; \sum_{i=1}^{n} (W_i \;/\; R_i)$

Summary

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

- **"Execution time is the only and unimpeachable measure of performance"**
  - CPU time equation can predict performance by estimating the effects of changing features.
- **Measuring performance requires good care**
  - Good ways to summarize performance
  - Good workloads (benchmarks)