CMPE 411
Computer Architecture

Lecture 4

Performance Evaluation and Metrics

Lecture's Overview

- ***Previous Lecture***
  - ➔ Various styles of MIPS addressing

    (Register, displacement, immediate, PC-relative and pseudo-direct)
  - ➔ Program starting steps

    (Compiler, Assembler, Linker, Loader)
  - ➔ Architectural design guidelines

    (keep it *simple, focus on common cases, smart compromises*)
- ***This Lecture***
  - ➔ Why measuring computer performance is important
  - ➔ Different performance metrics
  - ➔ Performance comparison

Introduction

- Hardware performance is a key to the effectiveness of the entire system
- Performance has to be measured and compared to evaluate various design and technological approaches
- To optimize the performance, major affecting factors have to be known
- For different types of applications, different performance metrics may be appropriate and different aspects of a computer system may be most significant
- Instructions use and implementation, memory hierarchy and I/O handling are among the factors that affect the performance

Defining Performance

- Performance means different things to different people, therefore its assessment is subtle

Analogy from the airlines industry:

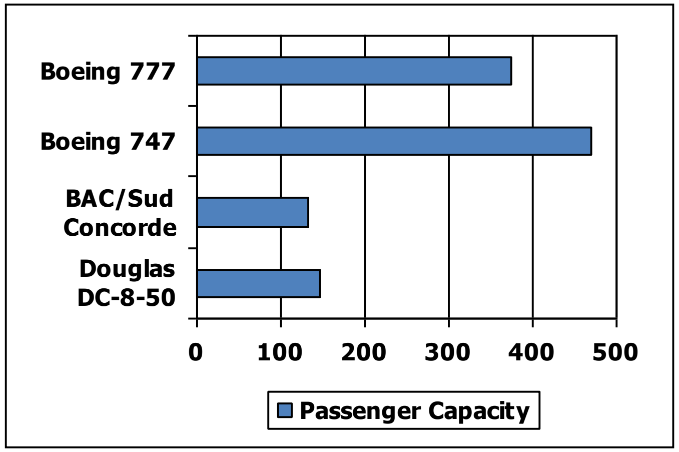
How to measure performance for a passenger airplane?

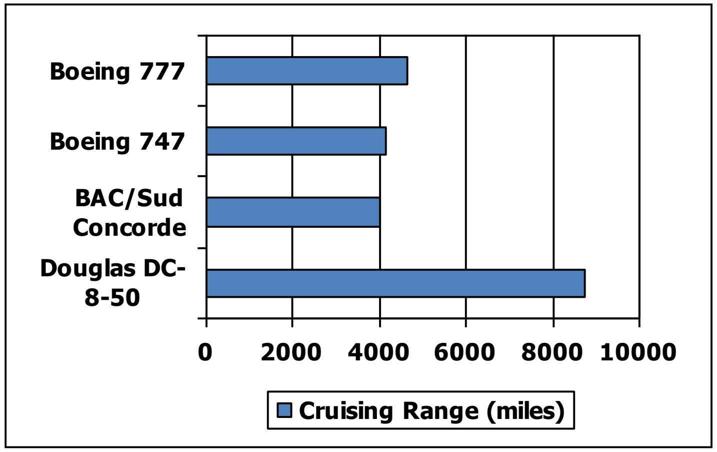
- Cruising speed *(How fast it gets to the destination)*
- Flight range *(How far it can reach)*
- Passenger capacity *(How many passenger it can carry)*
- All of the above

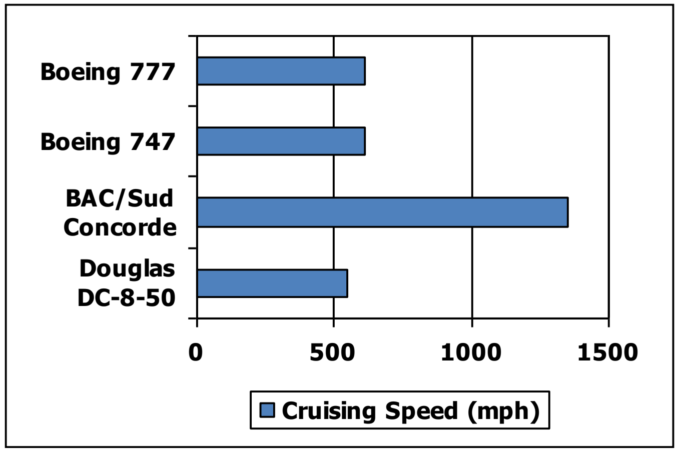
Airplane	**Passenger capacity**	**Cruising range (miles)**	**Cruising speed (m.p.h)**	**Passenger throughput (Passenger × m.p.h)**
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

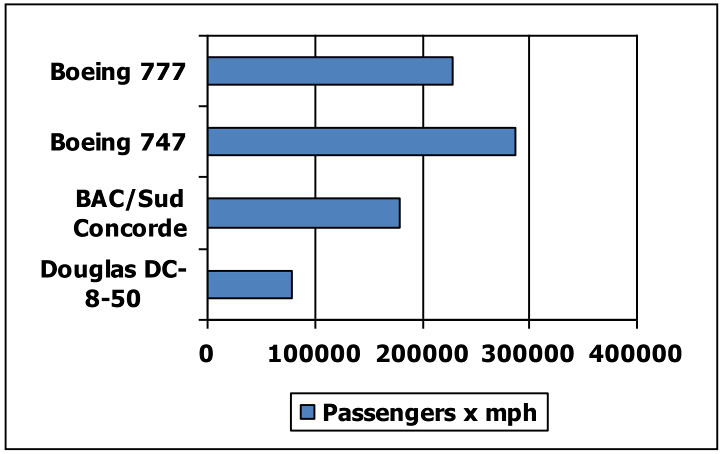
Criteria of performance evaluation differs among users and designers

Which airplane has the best performance?









Performance Metrics

- ❑ **Response (execution) time:**
  - *The time between the start and the completion of a task*
  - ➔ Measures user perception of the system speed
  - ➔ Common in reactive and time critical systems, single-user computer, etc.
- ❑ **Throughput:**
  - *The total number of tasks done in a given time*
  - ➔ Most relevant to batch processing (billing, credit card processing, etc.)
  - ➔ Mainly used for input/output systems (disk access, printer, etc.)
- ❑ ***Examples:***
  - Replacing the processor of a computer with a faster version
    - ➢ Enhances BOTH response time and performance
  - Adding additional processors to a system that uses multiple processors for separate tasks (e.g. handling of airline reservations system)
    - ➢ Improves ONLY throughput

Decreasing response time always improve throughput

Response-time Metric

- Maximizing performance means minimizing response (execution) time

$$\text{Performance} = \frac{1}{\text{Execution time}}$$

- Performance of Processor P_1 is better than P_2 is, for a given work load L, P_1 takes less time to execute L than P_2 does

$$\text{Performance}\,(P_1) > \text{Performance}\,(P_2)\ \text{w.r.t}\ L$$

$$\Rightarrow \text{Exection time}\,(P_1, L) < \text{Execution time}\,(P_2, L)$$

- Relative performance capture the performance ratio of of Processor P_1 compared to P_2 is, for the same work load

$$\frac{\text{CPUPerformance}\,(P_2)}{\text{CPUPerformance}\,(P_1)} = \frac{\text{Totel execution time}\,(P_1)}{\text{Totel execution time}\,(P_2)}$$

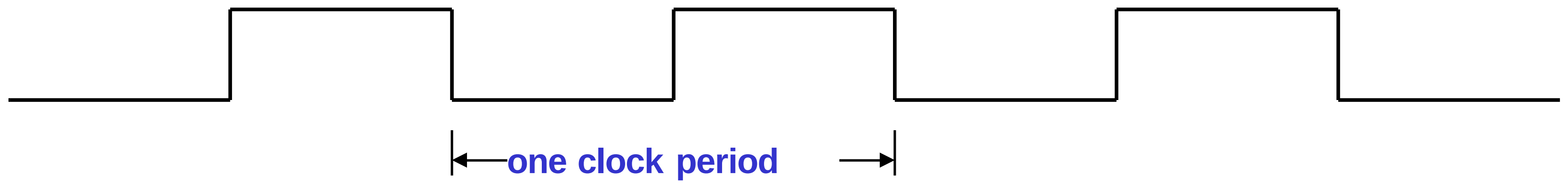
Measuring Performance (user's view)

- There are many user's interpretation of execution time:
  - Elapsed (response or wall-clock) time: meaning the total time spent until the completion of the task including I/O activities, operating system overhead, memory access, etc.
  - CPU time: meaning the time for which the task received service from the CPU until completion and could be divided to:
    - *User CPU time*:
      meaning the time the processor spent serving this task excluding blocking time waiting for an I/O or re-seize the CPU after preemption
    - *System CPU time*:
      meaning the time spent by the operating system to manage the task (operating system overhead)
- Unix *time* command reports all components, e.g. 90.7u 12.9s 2:39 65%,
  - User CPU time: 90.7 seconds
  - System CPU time: 12.9 seconds
  - Elapsed time: 2 minutes and 39 seconds (159 seconds)
  - Ratio of CPU to elapsed time = (90.7 + 12.9)/159 = 0.65
- Computer designers like to measure performance in the speed of basic functions or the clock speed

Review: Machine Clock Rate

- Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

CC = 1 / CR



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 clock cycle =>	1 GHz 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

* Slide is courtesy of Mary Jane Irwin

Designer's Performance Metrics

- ❑ Users and designers measure performance using different metrics
- ❑ Designers looks at the bottom line of the program execution

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

$$= \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

- ❑ To enhance the hardware performance, designers focuses on reducing the clock cycle time and the number of cycles per program
- ❑ Many techniques to decrease the number of clock cycles also increase the clock cycle time or the average number of cycles per instruction (CPI)

Example

A program runs in 10 seconds on a computer "A" that has 2 GHz clock. It is desirable to design a faster computer "B" that could run the program in 6 seconds. The designer has determined that a substantial increase in the clock speed is possible, however it would cause computer "B" to require 1.2 times as many clock cycles as computer "A". What should be the clock rate of computer "B"?

$$\text{CPU time (A)} = \frac{\text{CPU clock cycles}}{\text{Clock rate (A)}}$$

$$\text{10 seconds} = \frac{\text{CPU clock cycles of the program}}{2 \times 10^9 \text{ cycles/second}}$$

$$\text{CPU clock cycles of the program} = \text{10 seconds} \times 2 \times 10^9 \text{ cycles/second}$$

$$= 20 \times 10^9 \text{ cycles}$$

To get the clock rate of the faster computer, we use the same formula

$$\text{6 seconds} = \frac{1.2 \times \text{CPU clock cycles of the program}}{\text{clock rate (B)}} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{clock rate (B)}}$$

$$\text{clock rate (B)} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{6 second}} = 4 \times 10^9 \text{ cycles/second (4 GHz)}$$

Calculation of CPU Time

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

Or

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

Component of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instructions (CPI)	Average number of clock cycles/instruction
Clock cycle time	Seconds per clock cycle

Example

Suppose we have two implementation of the same instruction set architecture. Machine "A" has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and machine "B" has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which machine is faster for this program and by how much?

Both machines execute the same instructions for the program. Assume the number of instructions is "I",

CPU clock cycles (A) = $I \times 2.0$

CPU clock cycles (B) = $I \times 1.2$

The CPU time required for each machine is as follows:

$$\begin{aligned}\text{CPU time (A)} &= \text{CPU clock cycles (A)} \times \text{Clock cycle time (A)}\\ &= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

$$\begin{aligned}\text{CPU time (B)} &= \text{CPU clock cycles (B)} \times \text{Clock cycle time (B)}\\ &= I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}\end{aligned}$$

Therefore machine A will be faster by the following ratio:

$$\frac{\text{CPUPerformance (A)}}{\text{CPUPerformance (B)}} = \frac{\text{CPU time (B)}}{\text{CPU time (A)}} = \frac{500 \times I \text{ ps}}{600 \times I \text{ ps}} = 1.2$$

CPU Time (Cont.)

- CPU execution time can be measured by running the program
- The clock cycle is usually published by the manufacturer
- Measuring the CPI and instruction count is not trivial
- Instruction counts can be measured by: a software profiling, using an architecture simulator, using hardware counters on some architecture
- The CPI depends on many factors including: processor structure, memory system, the mix of instruction types and the implementation of these instructions
- Designers sometimes uses the following formula:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i$$

Where: ***C_i*** is the number of instructions of class *i* that gets executed

CPI_i is the average number of cycles per instruction for that instruction class

n is the number of different instruction classes

Comparing Code Segments

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

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

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code sequence	Instruction count for instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Answer:

Sequence 1: executes 2 + 1 + 2 = 5 instructions

Sequence 2: executes 4 + 1 + 1 = 6 instructions ✓

Comparing Code Segments (Cont.)

Using the formula: $$\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i$$

Sequence 1: CPU clock cycles = $(2 \times 1) + (1 \times 2) + (2 \times 3) = 10$ cycles

Sequence 2: CPU clock cycles = $(4 \times 1) + (1 \times 2) + (1 \times 3) = 9$ cycles

➢ Therefore Sequence 2 is faster although it executes more instructions

Using the formula: $$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

Sequence 1: CPI = 10/5 = 2

Sequence 2: CPI = 9/6 = 1.5

➢ Since Sequence 2 takes fewer overall clock cycles but has more instructions it must have a lower CPI

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	
Processor organization		X	X
Technology			X

* Slide is courtesy of Mary Jane Irwin

An Example

Op	Freq	CPI_i	Freq x 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
			Σ = 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

* Slide is courtesy of Mary Jane Irwin

Power Effect on Performance

- For CMOS chips, dominant energy consumption has been in switching transistors, called dynamic power (*static power depends on # transistors even if they are off)*

$$Power_{dynamic} \propto 1/2 \times Capacitive\ Load \times Voltage^2 \times Frequency\ switched$$

- For mobile devices, energy is a better metric

$$Energy_{dynamic} \propto Capacitive\ Load \times Voltage^2$$

- For a fixed task, slowing clock rate (frequency switched) reduces power, but not energy
- Capacitive load is a function of the number of transistors connected to output and technology, which determines capacitance of wires and transistors
- Dropping voltage helps both, so went from 5V to little over 1V
- To save energy & dynamic power, most CPUs now turn off clock of inactive modules (e.g. Fl. Pt. Unit)
- Suppose 15% reduction in voltage results in a 15% reduction in frequency. What is impact on dynamic power?

$$Power_{dynamic} \approx 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$

$$\approx 1/2 \times .85 \times CapacitiveLoad \times (.85 \times Voltage)^2 \times FrequencySwitched$$

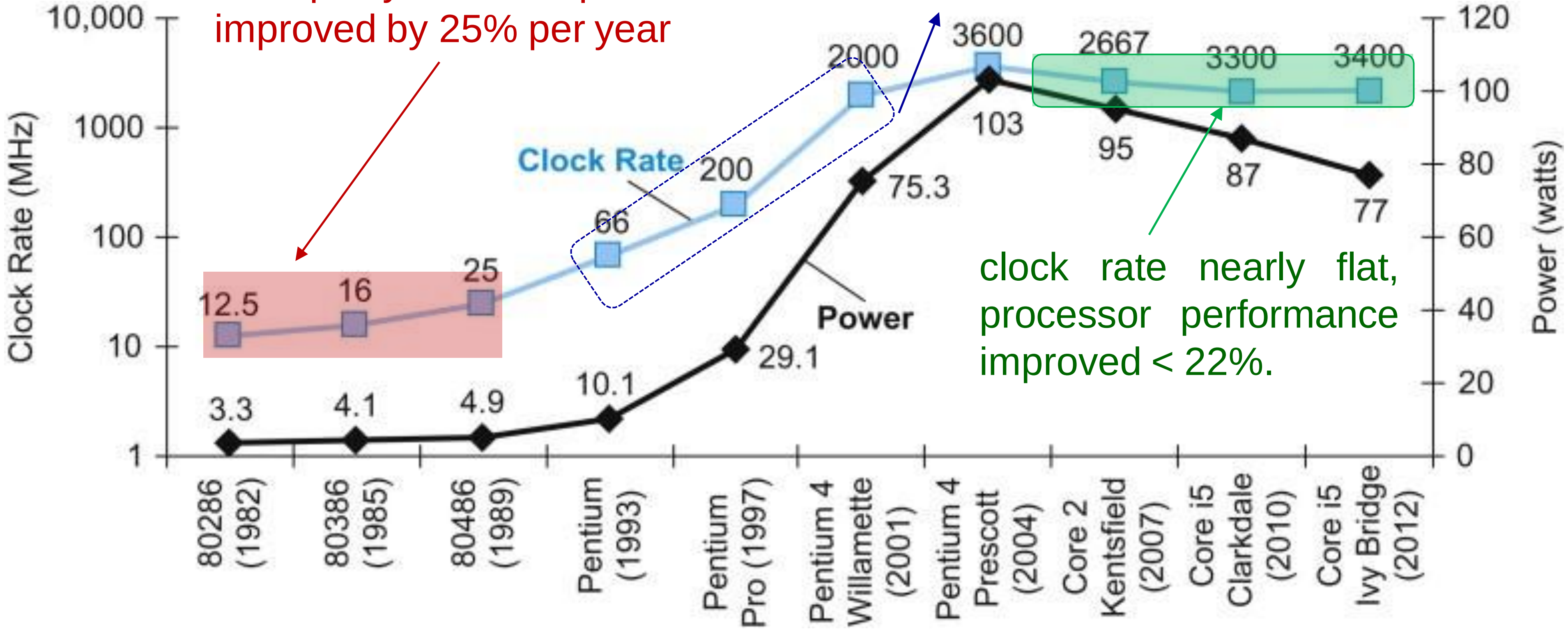
$$\approx (.85)^3 \times OldPower_{dynamic} \approx 0.6 \times OldPower_{dynamic}$$

* Slide is courtesy of Dave Patterson

Clock Rate and Power

Clock rate improved less than 15% per year while performance improved by 25% per year

"renaissance period" 52% performance and about 40% clock rates growth per year



clock rate nearly flat, processor performance improved < 22%.

- ❑ Power provides a limit to what we can cool, it has become a metric by itself in recent years
- ❑ Green computing and communication have become a major focus in recent years

Power Effect on Performance

- The power wall
    - We can’t reduce voltage further
    - We can’t remove more heat
- How else can we improve performance?

* Slide is courtesy of Dave Patterson

Conclusion

- ***Summary***
  - ➔ Why measuring performance is important and can be subtle
    (Execution time is the only unimpeachable measure of performance)
  - ➔ Different performance metrics
    (response time, throughput, CPU time)
  - ➔ Performance comparison
  - ➔ Effect on power on execution time (clock rate)

- ***Next Lecture***
  - ➔ Performance reports and summary
  - ➔ Selection of programs for performance evaluation
  - ➔ Widely used benchmark programs
  - ➔ Example industry metrics (e.g. MIPS, MFLOP, etc.)

Read sections 1.6-1.7 of the textbook