Virginia Polytechnic Institute and State University

CS2504: Computer Organization

Lecture 4: Evaluating Performance

Instructor: Dimitris Nikolopoulos

Guest Lecturer: Matthew Curtis-Maury

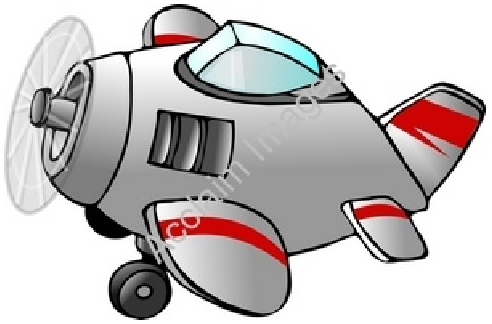
Understanding Performance

- Why do we study performance?
  - Evaluate during design
  - Evaluate before purchasing
  - Key to understanding underlying organizational motivation
- How can we (meaningfully) compare two machines?
  - Performance, Cost, Value, etc
- Main issue:
  - Need to understand what factors in the architecture contribute to overall system performance and the relative importance of these factors
    - Effects of ISA on performance
    - How will hardware change affect performance

Airplane Performance Analogy

Airplane	Passengers	Range	Speed
Boeing 777	375	4630	610
Boeing 747	470	4150	610
Concorde	132	4000	1250
Douglas DC-8-50	146	8720	544
Fighter Jet	4	2000	1500

- What metric do we use?
  - Concorde is 2.05 times faster than the 747
  - 747 has 1.74 times higher throughput
  - What about cost?
- And the winner is: *It Depends!*



Throughput vs. Response Time

- Response Time:
  - Execution time (e.g. seconds or clock ticks)
  - How long does the program take to execute?
  - How long do I have to wait for a result?
- Throughput:
  - Rate of completion (e.g. results per second/tick)
  - What is the average execution time of the program?
  - Measure of total work done

- Upgrading to a newer processor will improve: response time
- Adding processors to the system will improve: throughput

Example: Throughput vs. Response Time

- Suppose we know that an application that uses both a desktop client and a remote server is limited by network performance. For the following changes, which of throughput, response time, both, or neither is improved?
  - An extra network channel is added between the client and the server, increasing the total network throughput and reducing the delay to obtain network access.

    Throughput is improved directly and response time is improved by reducing delay.
  - The networking software is improved, thereby reducing the network communication delay, but not increasing throughput.

    Response time is improved directly.
  - More memory is added to the computer.

    Maybe neither. Maybe response time.

Design Goals

- **Performance** – maximum speed

- **Cost** – circuit size

- **Value** – best price-performance ratio

- **Energy** – least energy consumption

Definition of Performance

- Performance is *inversely* proportional to time:
  - To maximize performance, minimize execution time

$$\text{performance}_X = 1 / \text{execution\_time}_X$$

- "X is *n* times faster than Y"

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution\_time}_Y}{\text{execution\_time}_X} = \mathbf{n}$$

  - Execution time on Y is *n* times longer than on X

Example: Performance Calculation

- A particular multiprocessor server machine's performance is 4 times better than a given uniprocessor desktop system. If the desktop system runs an application in 28 seconds, how long will it take on the server?

$$\frac{\text{performance}_{\text{server}}}{\text{performance}_{\text{desktop}}} = \frac{28 \text{ seconds}}{\text{time}_{\text{server}}} = 4$$

$$\text{time}_{\text{server}} = 28 / 4 = 7 \text{ seconds}$$

Example: Relative Performance

- If a particular desktop runs a program in 60 seconds and a laptop runs the same program in 75 seconds, how much faster is the desktop than the laptop?

$\text{Performance}_{\text{desktop}} = 1/\text{execution\_time}_{\text{desktop}} = 1/60$

$\text{Performance}_{\text{laptop}} = 1/\text{execution\_time}_{\text{laptop}} = 1/75$

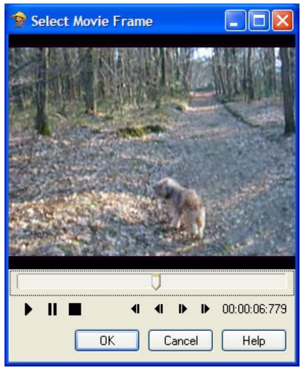
$\text{Performance}_{\text{desktop}} / \text{Performance}_{\text{laptop}} = (1/60)/(1/75) = 1.25$

Or simply: $\text{execution\_time}_{\text{laptop}} / \text{execution\_time}_{\text{desktop}} = 1.25$

So, the desktop is 1.25 times faster than the laptop

Real-time Constraints

- What are real-time systems?
  - Systems with "operational deadlines from event to system response"
- Examples of real-time systems:
  - Antilock brake system
    - Hard real-time
    - Must know immediately if the brakes have locked
  - Video playback
    - Soft real-time
    - Want to make most deadlines to avoid image jitter
- Real-time definition of performance:
  - "Are the deadlines met?"
  - Consequences on design: minimize cost to meet deadlines



Application-Specific Metrics

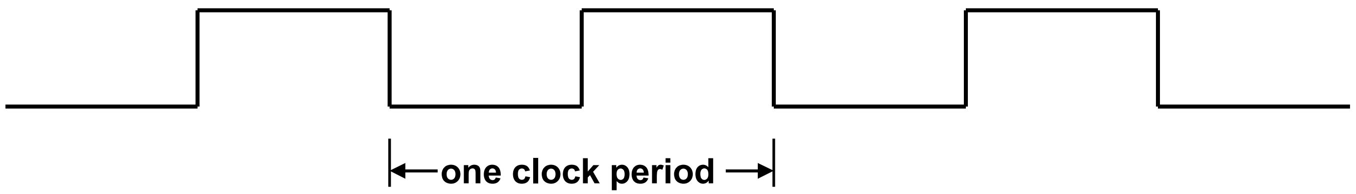
- Applications depend on different parts of computer
  - Scientific applications: CPU and memory
  - Server applications: I/O
- Also, need to find the right corresponding metric
  - Wall-clock time
  - Throughput
  - Both (maximum throughput with worst case response time)
- So, need to keep metric in mind for optimization
  - Identify bottleneck in terms of the target metric

Measuring Execution Time

- Time is the *ultimate* measure of performance
  - Same work in less time = better performance
- Most straight-forward definition of time
  - Wall-clock time, response time, elapsed time
  - Total time to completion of a task
- CPU time
  - Amount of time the task was actually executing
  - Doesn’t include I/O or running other tasks
- We will generally use CPU time

Clock Cycles

- *Clock Cycles* are a direct measure of time
  - Measures how fast the computer can perform basic functions
  - Discrete time interval in the CPU



- *Clock period* is the time for one clock cycle (seconds)
- *Clock rate* is the inverse of clock period (cycles/second)
  - 5 nsec clock cycle => 200 MHz clock rate
  - 500 psec clock cycle => 2 GHz clock rate
  - 200 psec clock cycle => 5 GHz clock rate

Execution Time Formula

- Relating cycles to seconds:

CPU_time = CPU_cycles * cycle_time

or

CPU_time = CPU_cycles / clock_rate

- So to improve performance we have two options
  - Decrease number of cycles in a program
  - Increase the clock rate (decrease cycle time)
  - However, these are often at odds with each other

Example: Improving Performance

Our favorite program runs in 10 seconds on computer A (4GHz). We are designing a computer to run the same program in 6 seconds. If increasing the clock rate will require 1.2 times as many cycles for computer B, what clock rate do we need?

Number of clock cycles executed by A:

$$10 \text{ seconds} = \frac{\text{clock\_cycles}_A}{4 * 10^9 \text{ cycles/second}}$$

clock_cycles_A = 10 seconds * (4*10^9 cycles/second) = 40*10^9 cycles

Then we find the clock rate needed on computer B:

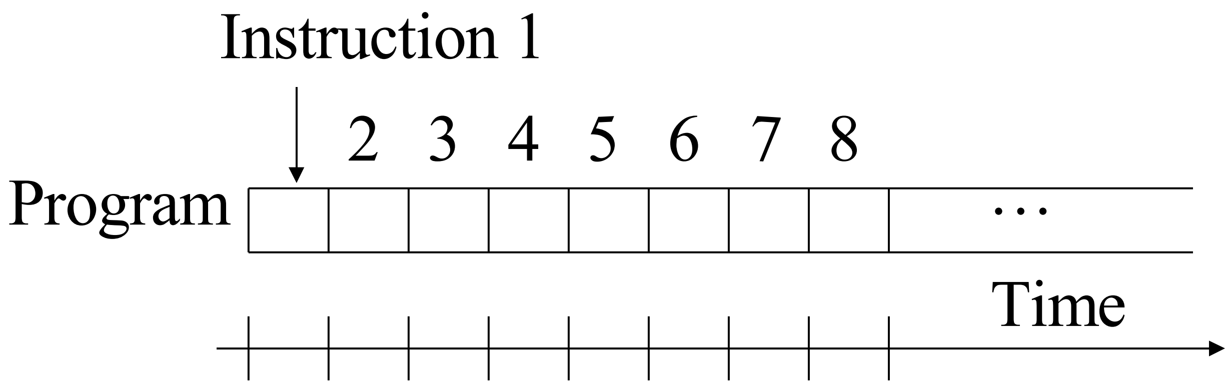
$6 \text{ seconds} = 1.2*(4*10^9 \text{ cycles}) / \text{clock\_rate}_B$

$\text{clock\_rate}_B = 1.2*(4*10^9 \text{ cycles}) / 6 \text{ seconds} = 8*10^9 \text{ cyc/sec}$ = 8 GHz

Determining Clock Cycles

- So what determines the number of cycles required to execute an application?

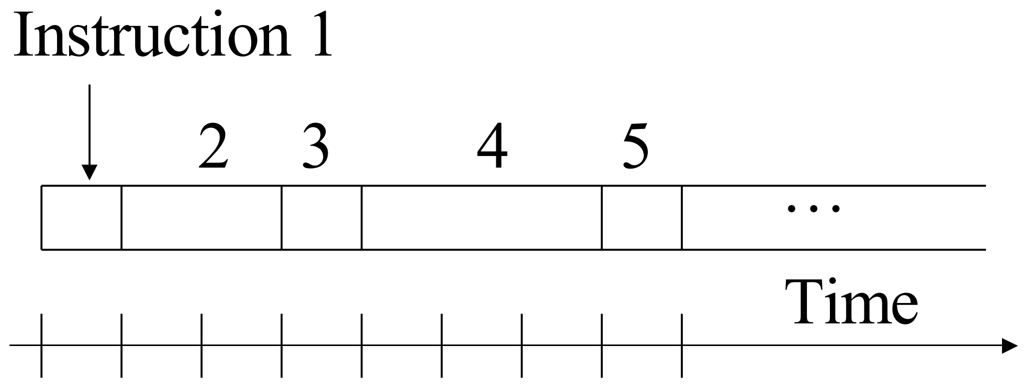
One possibility: #Cycles = #Instructions



However, this is NOT true because different instructions take different amounts of time

Determining Clock Cycles

- A more realistic picture of what's happening…



  - Floating point operations can take longer than integer
  - Multiplication takes longer than addition
  - Memory accesses can take many cycles to complete

Clock cycles = Instructions * Avg Cycles Per Instruction

Example: Calculating Time

- Suppose we have two implementations of the same ISA. Computer A has a cycle time of 250 ps and a CPI (cycles per instruction) of 2.0 for some program, and computer B has a cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program?

Note: A constant number of instructions will be executed: I

$\text{clock\_cycles}_A = I * 2.0$ and $\text{clock\_cycles}_B = I * 1.2$

$\text{time}_A = I * 2.0 * 250 \text{ ps} = 500 * I \text{ ps}$ and $\text{time}_B = I * 1.2 * 500 \text{ ps} = 600 * I \text{ ps}$

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{time}_B}{\text{time}_A} = \frac{600 * I \text{ ps}}{500 * I \text{ ps}} = \mathbf{1.2}$$

Computer A is 1.2 times faster

Clock Cycles per Instruction

- Then what is average Cycles Per Instruction?
  - The average number of cycles each instruction takes
  - A way to compare two implementations of one ISA
- CPI is dependent on the *instruction mix*
  - This is the composition of different types of instructions in an application
- Aware of variation in CPI by instruction type
  - Averages across all instructions executed
  - Specific to a given instruction sequence

Effective CPI

- Maximum CPI
  - CPI with instruction mix of exclusively the shortest instruction
- Calculating clock cycles

$$\text{CPU Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)$$

  - IC_i is the number of total instructions of class i
  - CPI_i is the average CPI for instruction class i
  - n is the number of instruction classes
  - Accounts for the weight and CPI of each instruction type
- Effective CPI

  CPI = Clock cycles / Number of instructions

Example: Calculating CPI

- Given the following CPIs for each instruction class and instruction mixes, which code sequence executes fewer instructions? Which is faster? Which has the lower CPI?

Class	A	B	C
CPI	1	2	3

Sequence	A	B	C
1	2	1	2
2	4	1	1

- **Instructions:** Seq 1: 2+1+2 = 5 ins ← Sequence 1 has fewer instructions
  Seq 2: 4+1+1 = 6ins

- **Cycles:** Seq 1: (2*1)+(1*2)+(2*3) = 10 cycles
  Seq 2: (4*1)+(1*2)+(1*3) = 9 cycles ← Sequence 2 is faster

- **CPI:** Seq 1: 10/5 = 2.0
  Seq 2: 9/6 = 1.5 ← Sequence 2 has the lower CPI

"THE" Performance Equation

- Combining the formulas we have seen:

Time = Instruction Count * CPI * Cycle time

Or

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

“THE” Performance Equation

- Separates the three key performance factors
  - Instructions, CPI, Clock rate
- Can help evaluate design decisions
  - Known effects on these terms can be translated into the overall effect on performance
- How can the values of these terms be found?
  - Time: by running the program
  - Clock rate: published by computer manufacturer
  - Instructions and CPI:
    - Hardware performance counters – CPU logic to record events
    - Simulation of the system

Example: Performance Equation

- A given application written in Java runs for 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 times as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler?

A: 15 seconds = Ins_A * CPI_A * ClockRate

ClockRate = 15 seconds / (Ins_A * CPI_A)

B: Time_B = (0.6*Ins_A) * (1.1*CPI_A) * ClockRate

Time_B = (0.6*Ins_A) * (1.1*CPI_A) * 15 seconds / (Ins_A * CPI_A)

Time_B = 0.6*1.1*15 seconds = 9.9 seconds

MIPS Performance

- MIPS is an alternative metric for performance
  - Million instructions per second

MIPS = Instructions / (Time * $10^{6)}$ = Clock Rate / CPI

- Inversely proportional to execution time
  - Bigger numbers indicate better performance
  - Intuitive representation

- 3 significant problems with MIPS usage
  - Doesn't consider what the instructions actually do
  - Varies by program; no single number for a machine
  - Can vary inversely with performance! (example to follow)

Example: MIPS Performance

- Given the following tables of instruction counts (in billions) and CPI for each instruction class, find the MIPS and execution times on a 4GHz machine.

Class	A	B	C
CPI	1	2	3

Sequence	A	B	C
1	5	1	1
2	10	1	1

$\text{Cycles}_1 = (5*1 + 1*2 + 1*3) * 10^9 = 10*10^9$ cycles

$\text{Cycles}_2 = (10*1 + 1*2 + 1*3) * 10^9 = 15*10^9$ cycles

$\text{Time}_1 = 10*10^9/4*10^9 = 2.5$ sec and $\text{Time}_2 = 15*10^9/4*10^9 = 3.75$ sec

Sequence1 is faster

$\text{MIPS1} = (5+1+1)*10^9 / 2.5*10^6 = 2800$

$\text{MIPS2} = (10+1+1)*10^9 / 3.75*10^6 = 3200$ $\leftarrow$ but Sequence2 has higher MIPS