CMSC 611: Advanced Computer Architecture

Performance

Some material adapted from Mohamed Younis, UMBC CMSC 611 Spr 2003 course slides
Some material adapted from Hennessy & Patterson / © 2003 Elsevier Science

Response-time Metric

- Maximizing performance means minimizing response (execution) time

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

Designer's Performance Metrics

- Users and designers measure performance using different metrics
  - Users: quotable metrics (GHz)
  - Designers: program execution

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

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

- Designer 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" with a 400 MHz clock. We desire 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"?

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

$$10 \text{ seconds} = \frac{\text{CPU clock cycles of program}}{400 \times 10^6 \text{ cycles/second}}$$

$$\text{CPU clock cycles of program} = 10 \text{ seconds} \times 400 \times 10^6 \text{ cycles/second}$$

$$= 4000 \times 10^6 \text{ cycles}$$

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

$$6 \text{ seconds} = \frac{1.2 \times \text{CPU clock cycles of program}}{\text{clock rate (B)}} = \frac{1.2 \times 4000 \times 10^6 \text{ cycles}}{\text{clock rate (B)}}$$

$$\text{clock rate (B)} = \frac{1.2 \times 4000 \times 10^6 \text{ cycles}}{6 \text{ second}} = 800 \times 10^6 \text{ cycles/second}$$

Calculation of CPU Time

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

Or

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

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

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

CPU Time (Cont.)

- CPU execution time can be measured by running the program
- The clock cycle is usually published by the manufacture
- Measuring the CPI and instruction count is not trivial
  - Instruction counts can be measured by: 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

CPU Time (Cont.)

- Designers sometimes uses the following formula:

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

Where: C_i is the count of number of instructions of class *i* executed
CPI_i is the average number of cycles per instruction for that instruction class
n is the number of different instruction classes

Example

Suppose we have two implementation of the same instruction set architecture. Machine "A" has a clock cycle time of 1 ns and a CPI of 2.0 for some program, and machine "B" has a clock cycle time of 2 ns 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 × 2.0 CPU clock cycles (B) = I × 1.2

The CPU time required for each machine is as follows:

CPU time (A) = CPU clock cycles (A) × Clock cycle time (A)
= I × 2.0 × 1 ns = 2 × I ns

CPU time (B) = CPU clock cycles (B) × Clock cycle time (B)
= I × 1.2 × 2 ns = 2.4 × I ns

Therefore machine A will be faster by the following ratio:

$$\frac{\text{CPU Performance (A)}}{\text{CPU Performance (B)}} = \frac{\text{CPU time (B)}}{\text{CPU time (A)}} = \frac{2.4 \times I \text{ ns}}{2 \times I \text{ ns}} = 1.2$$

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

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

Sequence 1: CPU clock cycles = (2 ×1) + (1 ×2) + (2 ×3) = 10 cycles

Sequence 2: CPU clock cycles = (4 ×1) + (1 ×2) + (1 ×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

The Role of Performance

- Hardware performance is a key to the effectiveness of the entire system
- Performance has to be measured and compared to evaluate designs
- To optimize the performance, major affecting factors have to be known
- For different types of applications
  - different performance metrics may be appropriate
  - 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

Calculation of CPU Time

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

	Instr. Count	CPI	Clock Rate
Program	X		
Compiler	X	X	
Instruction Set	X	X	
Organization		X	X
Technology			X

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

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

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

n is the number of different instruction classes

Important Equations (so far)

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

$$\text{Speedup} = \frac{\text{Performance (B)}}{\text{Performance (A)}} = \frac{\text{Time (A)}}{\text{Time (B)}}$$

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

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{CPI}_i \times \text{Instructions}_i$$

Amdahl's Law

The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by the improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

- A common theme in Hardware design is to *make the common case fast*
- Increasing the clock rate would not affect memory access time
- Using a floating point processing unit does not speed integer ALU operations

Example: Floating point instructions improved to run 2X; but only 10% of actual instructions are floating point

$$\textit{Exec-Time}_{new} = \textit{Exec-Time}_{old} \times (0.9 + .1/2) = 0.95 \times \textit{Exec-Time}_{old}$$

$$\textit{Speedup}_{overall} = \textit{Exec-Time}_{old} / \textit{Exec-Time}_{new} = 1/0.95 = 1.053$$

Ahmdal's Law for Speedup

$$\text{Time}_{\text{old}} = \text{Time}_{\text{old}} * \left(\text{Fraction}_{\text{unchanged}} + \text{Fraction}_{\text{enhanced}}\right)$$

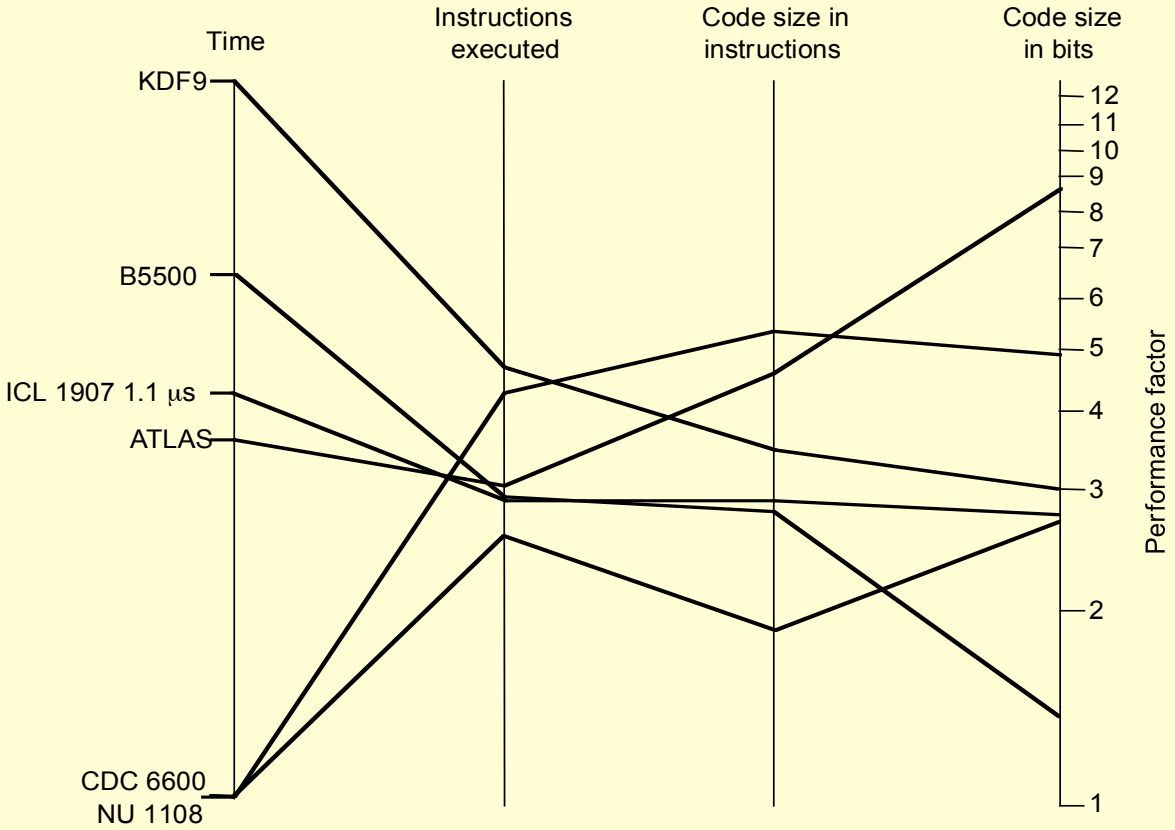
$$\text{Time}_{\text{new}} = \text{Time}_{\text{old}} * \left(\text{Fraction}_{\text{unchanged}} + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Time}_{\text{old}}}{\text{Time}_{\text{new}}} = \frac{\text{Time}_{\text{old}}}{\text{Time}_{\text{old}} * \left(\text{Fraction}_{\text{unchanged}} + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right)}$$

$$= \frac{1}{\text{Fraction}_{\text{unchanged}} + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Can Hardware-Independent Metrics Predict Performance?



- The Burroughs B5500 machine is designed specifically for Algol 60 programs
- Although CDC 6600's programs are over 3 times as big as those of B5500, yet the CDC machine runs them almost 6 times faster
- Code size cannot be used as an indication for performance

Comparing & Summarizing Performance

	Computer A	Computer B
Program 1 (seconds)	1	10
Program 2 (seconds)	1000	100
Total time (seconds)	1001	110

- Wrong summary can present a confusing picture
  - A is 10 times faster than B for program 1
  - B is 10 times faster than A for program 2
- Total execution time is a consistent summary measure
- Relative execution times for the same workload
  - Assuming that programs 1 and 2 are executing for the same number of times on computers A and B

$$\frac{\text{CPU Performance (B)}}{\text{CPU Performance (A)}} = \frac{\text{Total execution time (A)}}{\text{Total execution time (B)}} = \frac{1001}{110} = 9.1$$

Execution time is the only valid and unimpeachable measure of performance

Performance Summary (Cont.)

$$\text{Arithmetic Mean (AM)} = \frac{1}{n}\sum_{i=1}^{n}\text{Execution\_Time}_i$$

$$\text{Weighted Arithmetic Mean (WAM)} = \sum_{i=1}^{n} w_i \times \text{Execution\_Time}_i$$

Where: n is the number of programs executed

w_i is a weighting factor that indicates the frequency of executing program i

$$\text{with } \sum_{i=1}^{n} w_i = 1 \quad \text{and} \quad 0 \le w_i \le 1$$

- Weighted arithmetic means summarize performance while tracking exec. time
- Never use AM for normalizing time relative to a reference machine

	Time on A	Time on B	Norm. to A		Norm. to B	
			A	B	A	B
Program 1	1	10	1	10	0.1	1
Program 2	1000	100	1	0.1	10	1
AM of normalized time			1	5.05	5.05	1
AM of time	500.5	55	1	0.11	9.1	1

Performance Summary (Cont.)

$$\text{Geometric Mean (GM)} = \sqrt[n]{\prod_{i=1}^{n} \text{Execution\_Time\_ratio}_i}$$

Where: *n* is the number of programs executed

With $$\frac{\text{Geometric Mean}(X_i)}{\text{Geometric Mean}(Y_i)} = \text{Geometric Mean}\left(\frac{X_i}{Y_i}\right)$$

➡ Geometric mean is suitable for reporting average normalized execution time

	Time on A	Time on B	Norm. to A		Norm. to B	
			A	B	A	B
Program 1	1	10	1	10	0.1	1
Program 2	1000	100	1	0.1	10	1
GM of time or normalized time	31.62	31.62	1	1	1	1