EE380 Measuring Performance (Chapter 2)

Hank Dietz
Professor and James F. Hardymon Chair in Networking
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506-0046
http://aggregate.org/hankd/

Why Measure Performance?

- Performance is important!
- Identify HW/SW performance problems
- Comparisons
  - Which machine is faster?
  - Which ISA is better?
  - Which implementation (of an ISA) is faster?
- Expose significant performance issues
  (enable us to ignore unimportant issues)

More Than One Way To Measure Performance

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?

Computer Performance Measures

- Response Time and Throughput
  - Response Time: time to respond (complete an operation)
  - Throughput: jobs completed per unit time
  - Often can trade one for the other
- Performance(X) = 1/ExecutionTime(X)
- If Performance(X)>Performance(Y),
  X is Performance(X)/Performance(Y) times faster than Y
  X is ExecutionTime(y)/ExecutionTime(X) ""

For Whom The Clock Ticks?

- Real time: "Wall Clock" time, always ticking
- CPU time: ticks only when CPU is working for you
  - User: #CPUs times time spent executing your code
  - System: #CPUs times time spent executing OS code on your behalf
  - I/O time: not directly reported under UNIX
- Performance counter registers....
  - Processor clock tick counter
  - Counts of various types of events

Running what program?

- Each program can yield different performance numbers
- Applications
- "Toy" programs
- Benchmarks
- Synthetic Benchmarks

Common Performance Metrics

- Application/benchmark run time for specific data;
  e.g.: Quake display updates per second
- LIPS: Logical Inferences Per Second
- FLOPS: Floating-Point arithmetic Operations Per Second
- MB/s, Mb/s: MegaBytes/Megabits per Second
- MIPS: Millions of Instructions Per Second
- CPI: clock Cycles Per Instruction
  IPC: Instructions Per Clock cycle
- MOPS: Millions of (function unit) Operations Per Second
- Hz: (processor clock frequency) cycles Per Second

Thinking In Terms Of Clock Cycles

- Clock ticks at a constant rate, measure time in clock cycles:

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

- Prefer clock frequency? Divide by Hz...

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Cycles}}{\text{Program}} / \text{Frequency}$$

Thinking In Terms Of CPI

- Programs are made of instructions:

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

- Using CPI:

$$\frac{\text{Cycles}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \text{CPI}$$

- Or, using Instructions Per Clock:

$$\frac{\text{Cycles}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} / \text{IPC}$$

Summary

- Putting everything together:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \text{CPI} / \text{Frequency}$$

- Of course,
  different types of instructions have different CPI...
  - Use average CPI
  - Sum over all instruction types

What Affects What?

	Instruction Count	CPI	Clock Rate
Program (Algorithm)			
Compiler			
ISA			
Impl. Arch.			
VLSI			

What Affects What?

	Instruction Count	CPI	Clock Rate
Program (Algorithm)	Yes!	Indirectly...	No!
Compiler	Yes!	Indirectly...	Power?
ISA	Yes!	Yes!	Indirectly...
Impl. Arch.	uOps?	Yes!	Yes!
VLSI	No!	Indirectly...	Yes!

Example: How Does This Affect Design?

- For a particular application program,
- For a particular compiler,
- For a particular ISA,
- For a particular Implementation Architecture,
- For a particular VLSI technology,
- Etc.

Suppose:

Instruction Type	Execution Count	CPI	Clock Period
A	20	10	10ns
B	10	30	10ns

- Can use % rather than Execution Count
- Can use relative CPI rather than CPI

Amdahl's Law

- If 1/N time is not affected by an enhancement, best possible speedup is only N
- Originally for sequential overhead in parallel code, works just as well describing other changes

A Little Lesson From Top500.Org