Lecturer SOE
Dan Garcia

inst.eecs.berkeley.edu/~cs61c

UCB CS61C : Machine Structures

Lecture 38 – Performance 2008-04-30

How fast is your computer?

IBM BLUE GENE FASTEST COMPUTER EVER!

Every 6 months (Nov/June), the fastest supercomputers in the world face off. IBM's Blue Gene won again, with 65,536 processors. They use the LINPACK benchmark (A x = B).

www.research.ibm.com/bluegene/

Why Performance? Faster is better!

- Purchasing Perspective: given a collection of machines (or upgrade options), which has the
  - best performance ?
  - least cost ?
  - best performance / cost ?
- Computer Designer Perspective: faced with design options, which has the
  - best performance improvement ?
  - least cost ?
  - best performance / cost ?
- All require basis for comparison and metric for evaluation!
  - Solid metrics lead to solid progress!

Two Notions of "Performance"

Plane	DC to Paris	Top Speed	Passen-gers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concorde	3 hours	1350 mph	132	178,200

- Which has higher performance?
  - Interested in time to deliver 100 passengers?
  - Interested in delivering as many passengers per day as possible?
- In a computer, time for one task called Response Time or Execution Time
- In a computer, tasks per unit time called Throughput or Bandwidth

Definitions

- Performance is in units of things per sec
  - bigger is better
- If mostly concerned with response time
  - $\text{performance}(x) = \frac{1}{\text{execution\_time}(x)}$
- " F(ast) is *n* times faster than S(low) " means:

$$n = \frac{\text{performance(F)}}{\text{performance(S)}} = \frac{\text{execution\_time(S)}}{\text{execution\_time(F)}}$$

Example of Response Time v. Throughput

- Time of Concorde vs. Boeing 747?
  - Concord is 6.5 hours / 3 hours = 2.2 times faster
  - Concord is 2.2 times ("120%") faster in terms of flying time (response time)
- Throughput of Boeing vs. Concorde?
  - Boeing 747: 286,700 pmph / 178,200 pmph = 1.6 times faster
  - Boeing is 1.6 times ("60%") faster in terms of throughput
- We will focus primarily on response time.

Words, Words, Words…

- Will (try to) stick to "n times faster"; its less confusing than "m % faster"
- As faster means both decreased execution time and increased performance, to reduce confusion we will (and you should) use "improve execution time" or "improve performance"

What is Time?

- **Straightforward definition of time:**
  - Total time to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, ...
  - "real time", "response time" or "elapsed time"
- **Alternative: just time processor (CPU) is working only on your program (since multiple processes running at same time)**
  - "CPU execution time" or "CPU time"
  - Often divided into system CPU time (in OS) and user CPU time (in user program)

How to Measure Time?

- **Real Time ⇒ Actual time elapsed**
- **CPU Time: Computers constructed using a clock that runs at a constant rate and determines when events take place in the hardware**
  - These discrete time intervals called clock cycles (or informally clocks or cycles)
  - Length of clock period: clock cycle time (e.g., ½ nanoseconds or ½ ns) and clock rate (e.g., 2 gigahertz, or 2 GHz), which is the inverse of the clock period; use these!

Measuring Time using Clock Cycles (1/2)

- **CPU execution time for a program**
  - Units of [seconds / program] or [s/p]

= Clock Cycles for a program x Clock Period

- Units of [s/p] = [cycles / p] x [s / cycle] = [c/p] x [s/c]

- **Or**

$$= \frac{\text{Clock Cycles for a program [c / p]}}{\text{Clock Rate [c / s]}}$$

Measuring Time using Clock Cycles (2/2)

- **One way to define clock cycles:**

Clock Cycles for program [c/p]

= Instructions for a program [i/p] (called "Instruction Count")

x Average Clock cycles Per Instruction [c/i] (abbreviated "CPI")

- **CPI one way to compare two machines with same instruction set, since Instruction Count would be the same**

Performance Calculation (1/2)

- **CPU execution time for program [s/p]**
  = Clock Cycles for program [c/p]
  x Clock Cycle Time [s/c]

- **Substituting for clock cycles:**

CPU execution time for program [s/p]
= (Instruction Count [i/p] x CPI [c/i])
x Clock Cycle Time [s/c]

= Instruction Count x CPI x Clock Cycle Time

Performance Calculation (2/2)

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

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

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

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

Product of all 3 terms: if missing a term, can't predict time, the real measure of performance

How Calculate the 3 Components?

- **Clock Cycle Time: in specification of computer (Clock Rate in advertisements)**
- **Instruction Count:**
  - Count instructions in loop of small program
  - Use simulator to count instructions
  - Hardware counter in spec. register
    - (Pentium II,III,4)
- **CPI:**
  - Calculate: $\frac{\text{Execution Time / Clock cycle time}}{\text{Instruction Count}}$
  - Hardware counter in special register (PII,III,4)

Calculating CPI Another Way

- **First calculate CPI for each individual instruction (`add`, `sub`, `and`, etc.)**
- **Next calculate frequency of each individual instruction**
- **Finally multiply these two for each instruction and add them up to get final CPI (the weighted sum)**

Example (RISC processor)

Op	$Freq_i$	CPI_i	Prod	(% Time)
ALU	50%	1	.5	(23%)
Load	20%	5	1.0	(45%)
Store	10%	3	.3	(14%)
Branch	20%	2	.4	(18%)
	Instruction Mix		**2.2**	**(Where time spent)**

- **What if Branch instructions twice as fast?**

What Programs Measure for Comparison?

- **Ideally run typical programs with typical input before purchase, or before even build machine**
  - Called a "workload"; For example:
  - Engineer uses compiler, spreadsheet
  - Author uses word processor, drawing program, compression software
- **In some situations its hard to do**
  - Don't have access to machine to "benchmark" before purchase
  - Don't know workload in future

Benchmarks

- **Obviously, apparent speed of processor depends on code used to test it**
- **Need industry standards so that different processors can be fairly compared**
- **Companies exist that create these benchmarks: "typical" code used to evaluate systems**
- **Need to be changed every ~5 years since designers could (and do!) target for these standard benchmarks**

Example Standardized Benchmarks (1/2)

- **Standard Performance Evaluation Corporation (SPEC) SPEC CPU2006**
  - CINT2006 12 integer (perl, bzip, gcc, go, ...)
  - CFP2006 17 floating-point (povray, bwaves, ...)
  - All relative to base machine (which gets 100) Sun Ultra Enterprise 2 w/296 MHz UltraSPARC II
  - They measure
    - System speed (SPECint2006)
    - System throughput (SPECint_rate2006)
  - `www.spec.org/osg/cpu2006/`

Example Standardized Benchmarks (2/2)

- **SPEC**
  - Benchmarks distributed in source code
  - Members of consortium select workload
    - 30+ companies, 40+ universities, research labs
  - Compiler, machine designers target benchmarks, so try to change every 5 years
  - SPEC CPU2006:

CINT2006		
perlbench	C	Perl Programming language
bzip2	C	Compression
gcc	C	C Programming Language Compiler
mcf	C	Combinatorial Optimization
gobmk	C	Artificial Intelligence : Go
hmmer	C	Search Gene Sequence
sjeng	C	Artificial Intelligence : Chess
libquantum	C	Simulates quantum computer
h264ref	C	H.264 Video compression
omnetpp	C++	Discrete Event Simulation
astar	C++	Path-finding Algorithms
xalancbmk	C++	XML Processing

CFP2006		
bwaves	Fortran	Fluid Dynamics
gamess	Fortran	Quantum Chemistry
milc	C	Physics / Quantum Chromodynamics
zeusmp	Fortran	Physics / CFD
gromacs	C,Fortran	Biochemistry / Molecular Dynamics
cactusADM	C,Fortran	Physics / General Relativity
leslie3d	Fortran	Fluid Dynamics
namd	C++	Biology / Molecular Dynamics
dealII	C++	Finite Element Analysis
soplex	C++	Linear Programming, Optimization
povray	C++	Image Ray-tracing
calculix	C,Fortran	Structural Mechanics
GemsFDTD	Fortran	Computational Electromegnetics
tonto	Fortran	Quantum Chemistry
lbm	C	Fluid Dynamics
wrf	C,Fortran	Weather
sphinx3	C	Speech recognition

Another Benchmark

- **PCs: Ziff-Davis Benchmark Suite**
  - "Business Winstone is a system-level, application-based benchmark that measures a PC's overall performance when running today's top-selling Windows-based 32-bit applications… it doesn't mimic what these packages do; it runs real applications through a series of scripted activities and uses the time a PC takes to complete those activities to produce its performance scores.
  - Also tests for CDs, Content-creation, Audio, 3D graphics, battery life
- **`www.etestinglabs.com/benchmarks/`**

Performance Evaluation: An Aside Demo

If we're talking about performance, let's discuss the ways shady salespeople have fooled consumers (so you don't get taken!)

5. **Never let the user touch it**
4. **Only run the demo through a script**
3. **Run it on a stock machine in which "no expense was spared"**
2. **Preprocess all available data**
1. **Play a movie**

Megahertz Myth Marketing Movie

Peer Instruction

A. **Rarely does a company selling a product give unbiased performance data.**

B. **The Sieve of Eratosthenes and Quicksort were early effective benchmarks.**

C. **A program runs in 100 sec. on a machine, `mult` accounts for 80 sec. of that. If we want to make the program run 6 times faster, we need to up the speed of `mult`s by AT LEAST 6.**

	ABC
0:	FFF
1:	FFT
2:	FTF
3:	FTT
4:	TFF
5:	TFT
6:	TTF
7:	TTT

"And in conclusion…"

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

- **Latency v. Throughput**
- **Performance doesn't depend on any single factor: need Instruction Count, Clocks Per Instruction (CPI) and Clock Rate to get valid estimations**
- **User Time: time user waits for program to execute: depends heavily on how OS switches between tasks**
- **CPU Time: time spent executing a single program: depends solely on design of processor (datapath, pipelining effectiveness, caches, etc.)**
- **Benchmarks**
  - Attempt to predict perf, Updated every few years
  - Measure everything from simulation of desktop graphics programs to battery life
- **Megahertz Myth**
  - MHz ≠ performance, it's just one factor