ECSE 425 Lecture 5: Quantifying Computer Performance

H&P Chapter 1

Last Time

- Trends in Dependability
- Quantitative Principles of Computer Design

Today

- Quantifying Computer Performance
- Looking ahead …
  - On Friday, Pipelining
    - Read Appendix A!
  - Homework 1 due Monday
    - OH Tomorrow: 11 AM-12 PM

Processor Performance Equation

$$\text{CPU\_Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{ClockCycles}}{\text{Instructions}} \times \frac{\text{Seconds}}{\text{ClockCycle}}$$

IC CPI CT

- Instruction count (IC): how many instructions are required to execute a program
  - a function of compiler technology and the ISA
- Clock cycles per instruction (CPI): how many cycles are required to execute an instruction
  - a function of the ISA and organization
- Clock cycle time (CT): clock cycle length in seconds
  - a function of organization and technology

IC and CPI by Instruction Class

$$\text{ClockCycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

$$\text{CPI} = \sum_{i=1}^{n} \frac{\text{IC}_i}{\text{Instruction Count}} \times \text{CPI}_i$$

Operation	Freq (%)	CPI_i	Term
ALU	50	5	2.5
Load	20	4	0.8
Store	10	5	0.5
Branch	20	2	0.4

- Example: *gcc*
- CPI = 4.2

PPE Example

- Assume
  - Frequency of FP operations = 25%
  - Average CPI of FP operations = 4
  - Average CPI of other instructions = 1.33
- Two possible enhancements:
  - Decrease the average CPI of all FP to 2.5, or
  - Decrease the CPI of FPSQR from 20 to 2, given that the frequency of FPSQR is 2%
- Which enhancement is better?

Measuring System Performance

- How should we measure *system performance*?
  - Response time (latency)
    - time needed to get a result
  - Throughput (bandwidth)
    - amount of computation per unit time
- Example: Montreal to Paris

Aircraft	Time (hours)	Speed (mph)	Passengers	PMPH
747	6.5	610	470	286,700
Concorde	3	1350	132	178,200

- Different applications call for different metrics

Relative Performance

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

$$n = \frac{\text{ExecTime}_Y}{\text{ExecTime}_X} = \frac{1/\text{Performance}_Y}{1/\text{Performance}_X} = \frac{\text{Performance}_X}{\text{Performance}_Y}$$

- How do you measure "time"?

Different Measures of "Time"

- Wall-clock time
  - Elapsed time required to complete a given task
    - Includes I/O, memory, OS overhead, everything.
    - Other processes may interfere
- CPU time
  - Time spent by the CPU on behalf of one task
    - User CPU time (spent running user code)
    - System CPU time (time spent running OS code)
- Unix `time` function reports wall-clock, cpu, and system time

“Time” required for what? Benchmarks.

- Real applications
  - Whatever software your customer cares about most
- Kernels
  - Small, key pieces of real program
  - Ideally, your customer cares about this piece
- Toy applications
  - Small and interesting programs, Sieve of Eratosthenes, Towers of Hanoi, Puzzles, Quicksort
- Synthetic benchmarks
  - Fake programs that execute a representative mix of operations

Summarizing Performance

- What if many programs are important?

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

- Summarizing the results in single number
  - A is 10x faster than B for Program 1
  - B is 10x faster than A for Program 2
  - Perf. B / Perf. A = Total A / Total B = 1001/110 = 9.1
- B is 9.1x faster than A if Program 1 and Program 2 are run an equal number of times

Arithmetic Mean

- The **arithmetic mean** tracks with the total execution time

$$\text{ArithmeticMean} = \frac{1}{n}\sum_{i=1}^{n} \text{Time}_i$$

- What if some programs have a much longer execution time than others?
  - Use weights to remove bias with normalization

$$WeightedArithmeticMean = \sum_{i=1}^{n} Weight_i \times Time_i$$

- Any problem with this?

Normalized Ratios

- SPEC publishes benchmarks results relative to a reference computer

$$\text{SPECRatio} = \frac{\text{ExecTimeRef}}{\text{ExecTime}}$$

- If the SPECRatio of computer A on a benchmark is 1.25 times higher than computer B then

$$1.25 = \frac{\text{SPECRatioA}}{\text{SPECRatioB}} = \frac{\text{tref/tA}}{\text{tref/tB}} = \frac{\text{tB}}{\text{tA}} = \frac{\text{perfA}}{\text{perfB}}$$

- The choice of reference is not important

Geometric Mean

- Ratios must be averaged geometrically:

$$GeometricMean = \sqrt[n]{\prod_{i=1}^{n} sample_i}$$

- In the case of SPEC, $sample_i$ is $SPECRatio_i$
  - The mean of the ratios = the ratio of the means
  - The choice of reference machine is irrelevant

GM is the only correct mean when averaging normalized results!

Standard Deviation

- Is one number enough?
- *Stdev* characterizes the variability around the mean

$$stdev = \sqrt{\sum_{i=1}^{n} (sample_i - Mean)^2}$$

- Recall the geometric mean; we can re-write it as

$$GeometricMean = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln(sample_i)\right)$$

- The geometric standard deviation is then

$$gstdev = \exp\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(\ln(sample_i) - \ln(GeometricMean)\right)^2}\right)$$

System Performance Example

- From data in Figure 1.14 (H&P)
- Geometric mean
  - Opteron = 20.86
  - Itanium 2 = 27.12
- Standard deviation
  - Opteron = 1.38
  - Itanium 2 = 1.93
- The Itanium 2 results differ more widely from the mean and are therefore likely less predictable
- What are the implications for the comparison?

Summary

- Processor performance equation
  - CPU time = IC * CPI * CT
- System performance
  - Latency vs. throughput
  - Wall-clock vs. CPU time
  - *Different applications call for different metrics*
- Benchmarks
  - Use real applications to get real measurements

Next Time

- Pipelining
  - Read Appendix A!