CS 700: Quantitative Methods & Experimental Design in Computer Science

Sanjeev Setia
Dept of Computer Science
George Mason University

Logistics

- ☐ Grade: 25% project, 30% Homework assignments, 25% midterm, 20% take home final
- ☐ Slides, assignments, reading material on class web page http://www.cs.gmu.edu/~setia/cs700/
- ☐ Several small assignments related to material discussed in class
  - Not all will be graded, but we will go over solutions in class
- ☐ Term project
  - should involve experimentation (measurement, simulation)
  - select a topic in your research area if possible
  - apply techniques discussed in this class

Readings

- Textbook
  - David Lilja, "Measuring Computer Performance: A Practitioner's Guide"
    - Alternative Text: Raj Jain, "Art of Computer Systems Performance Analysis" (somewhat dated)
- Class notes, slides
- Relevant research articles (links on class web site)
- Books on probability and statistics for engineers (see syllabus)
- More specialized topics
  - Cohen "Empirical techniques in AI"
  - Crovella et al "Internet Measurement"

Course Topics

- Basic techniques in "experimental" computer science
  - Basic measurement tools and techniques
  - Simulation
  - Design of experiments
- Quantitative Methods
  - Use of statistical techniques in design of experiments
  - Use of statistical techniques in comparing alternatives
  - Characterizing and interpreting measured data
- Simple analytical modeling

Most examples will be from performance measurement of computer systems and networks, but techniques are applicable in all fields of CS

Experimental Science

Scientific Method

1. Identify a problem and form hypothesis
   - Hypothesis must be testable and refutable
2. Design an experiment
3. Conduct the experiment
4. Perform hypothesis testing
   - Use statistical techniques

What about Computer Science?

Experimental Computer Science

Three definitions (Feitelson, 2006)

- ☐ Building of systems, hardware or software
  - Counterpart to "theoretical CS"
- ☐ Mathematical modeling of the behavior of computer systems [Denning]
  - Analogous to experimental science being the modeling of nature by mathematical laws
  - In this definition, experimentation is used as a feedback step in engineering loop
- ☐ Evaluation of computer systems using the methodologies of the natural sciences, i.e. rigorous methodologies
  - Focus of this class

Feitelson makes the case that there is a place for observation of the real world, as in the natural sciences, e.g., analyzing measured network traffic

The Role of Experimentation in CS

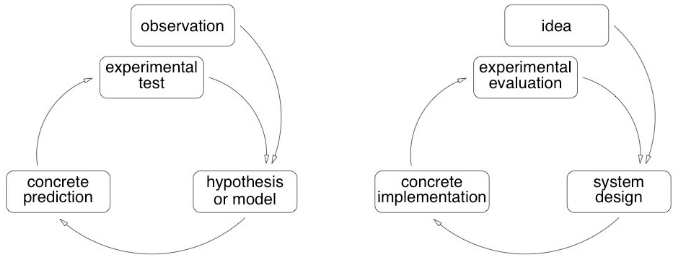


Figure 1: *A comparison of the scientific method (on the left) with the role of experimentation in system design (right).*

Schedule

- Introduction
- Metrics
- Summarizing Measured Data
- Measurement Techniques
- Simulation
- Comparing Alternatives
- Linear Regression Models
- Design of experiments
- Interpreting & characterizing measured data
- Analytical Modeling

Course Goals

- Understand the inherent trade-offs involved in using simulation, measurement, and analytical modeling.
- Rigorously compare computer systems/ networks/software/artifacts/… often in the presence of measurement noise
  - Usually compare performance but in many fields of CS, “quality” of the output is more important than raw performance, e.g. face recognition software
- Determine whether a change made to a system has a statistically significant impact

Course Goals

- Use statistical tools to reduce the number of simulations that need to be performed of a computer system.
- Design a set of experiments to obtain the most information for a given level of effort.

Course Goals

- Provide intuitive conceptual background for some standard statistical tools.
  - Draw meaningful conclusions in presence of noisy measurements.
  - Allow you to correctly and intelligently apply techniques in new situations.

Course Goals

- Present techniques for aggregating and interpreting large quantities of data.
  - Obtain a big-picture view of your results.
  - Obtain new insights from complex measurement and simulation results.

→ E.g. How does a new feature impact the overall system?

Agenda

- Today
  - Overview of course
  - Performance metrics
    - Characteristics of good metrics
    - Standard processor and system metrics
    - Speedup and relative change

Agenda (cont.)

- Measurement tools and techniques
  - Fundamental strategies
  - Interval timers
  - Cycle counters
  - Program profiling
  - Tracing
  - Indirect measurement
  - Measuring network delays and bandwidth

Agenda (cont.)

- Simulation
  - Types of simulations
  - Random number generation
  - Verification and validation

Agenda (cont.)

- Statistical interpretation of measured data
  - Arithmetic, harmonic, geometric means
  - Sources of measurement error
  - Confidence intervals
  - Statistically comparing alternatives

Agenda (cont.)

- Design of experiments
  - Terminology
  - One-factor analysis of variance (ANOVA)
  - Two-factor ANOVA
  - Generalized *m*-factor experiments
  - Fractional factorial designs
  - Multi-factorial designs (Plackett and Berman)

Agenda (cont'd)

- Characterizing Measured Data
  - Workload Characterization
  - Fitting Measured Data to Known Distributions
    - Q-Q plots
    - Chi-square, K-S tests

Agenda (cont'd)

- Simple analytical modeling
  - Simple queuing models
    - Single queue models
    - Networks of queues
  - Operational analysis
    - Little's Law

Readings

- Dror Feitelson, "Experimental Computer Science: The need for a cultural change"

Metrics

Performance metrics

- ☐ What is a performance metric?
- ☐ Characteristics of good metrics
- ☐ Standard processor and system metrics
- ☐ Speedup and relative change

Discussion focuses on *performance* metrics, but concepts apply more generally

What is a performance metric?

- Count
  - Of how many times an event occurs
- Duration
  - Of a time interval
- Size
  - Of some parameter
- A value derived from these fundamental measurements

Time-normalized metrics

- “Rate” metrics
  - Normalize metric to common time basis
    - Transactions per second
    - Bytes per second
  - (Number of events) ÷ (time interval over which events occurred)
- “Throughput”
- Useful for comparing measurements over different time intervals

What makes a “good” metric?

- Allows accurate and detailed comparisons
- Leads to correct conclusions
- Is well understood by everyone
- Has a quantitative basis
- A good metric helps avoid erroneous conclusions

Good metrics are …

- Linear
  - Fits with our intuition
  - If metric increases 2x, performance should increase 2x
  - Not an absolute requirement, but very appealing
    - dB scale to measure sound is nonlinear

Good metrics are …

- Reliable
  - If metric A > metric B
    - Then, Performance A > Performance B
  - Seems obvious!
  - However,
    - MIPS(A) > MIPS(B), but
    - Execution time (A) > Execution time (B)

Good metrics are …

- Repeatable
  - Same value is measured each time an experiment is performed
  - Must be *deterministic*
  - Seems obvious, but not always true…
    - E.g. Time-sharing changes measured execution time

Good metrics are …

- Easy to use
  - No one will use it if it is hard to measure
  - Hard to measure/derive
    - → less likely to be measured correctly
  - A *wrong value* is worse than a bad metric!

Good metrics are …

- Consistent
  - Units and definition are constant across systems
  - Seems obvious, but often not true…
    - E.g. MIPS, MFLOPS
  - Inconsistent → impossible to make comparisons

Good metrics are …

- ❑ Independent
  - A lot of $$$ riding on performance results
  - Pressure on manufacturers to *optimize* for a particular metric
  - Pressure to influence *definition* of a metric
  - But a good metric is independent of this pressure

Good metrics are …

- ❑ Linear -- *nice, but not necessary*
- ❑ Reliable -- *required*
- ❑ Repeatable -- *required*
- ❑ Easy to use -- *nice, but not necessary*
- ❑ Consistent -- *required*
- ❑ Independent -- *required*

Clock rate

- Faster clock == higher performance
  - 1 GHz processor always better than 2 GHz
- But is it a proportional increase?
- What about architectural differences?
  - Actual operations performed per cycle
  - Clocks per instruction (CPI)
  - Penalty on branches due to pipeline depth
- What if the processor is not the bottleneck?
  - Memory and I/O delays

Clock rate

- (Faster clock) ≠ (better performance)
- A good first-order metric

Linear	
Reliable	
Repeatable	☺
Easy to measure	☺
Consistent	☺
Independent	☺

MIPS

- Measure of computation "speed"
- *Millions of instructions executed per second*
- MIPS = n / (T_e * 1000000)
  - n = number of instructions
  - T_e = execution time
- Physical analog
  - Distance traveled per unit time

MIPS

- But how much actual computation per instruction?
  - E.g. *CISC* vs. *RISC*
  - Clocks per instruction (CPI)
- MIPS = *Meaningless Indicator of Performance*

Linear	
Reliable	
Repeatable	☺
Easy to measure	☺
Consistent	
Independent	☺

MFLOPS

- Better definition of “distance traveled”
- 1 unit of computation (~distance) ≡ 1 floating-point operation
- *Millions of floating-point ops per second*
- MFLOPS = $f / (T_e * 1000000)$
  - f = number of floating-point instructions
  - T_e = execution time
- GFLOPS, TFLOPS,…

MFLOPS

- Integer program = 0 MFLOPS
  - But still doing useful work
  - Sorting, searching, etc.
- How to count a FLOP?
  - E.g. transcendental ops, roots
- Too much flexibility in definition of a FLOP
- Not consistent across machines

Linear	
Reliable	
Repeatable	☺
Easy to measure	☺
Consistent	
Independent	

SPEC

- System Performance Evaluation Coop
- Computer manufacturers select "representative" programs for benchmark suite
- Standardized methodology
  - Measure execution times
  - Normalize to standard basis machine
  - SPECmark = geometric mean of normalized values

SPEC

- Geometric mean is inappropriate (more later)
- SPEC rating does not correspond to execution times of non-SPEC programs
- Subject to tinkering
  - Committee determines which programs should be part of the suite
  - Targeted compiler optimizations

Linear	
Reliable	
Repeatable	☺
Easy to measure	½ ☺
Consistent	☺
Independent	

Execution time

- Ultimately interested in time required to execute *your* program
- Smallest total execution time == highest performance
- Compare times directly
- Derive appropriate rates
- Time == *fundamental metric* of performance
  - If you can't measure time, you don't know anything

Execution time

- "Stopwatch" measured execution time

  Start_count = read_timer();

  Portion of program to be measured

  Stop_count = read_timer();

  Elapsed_time = (stop_count – start_count) * clock_period;
- Measures "wall clock" time
  - Includes I/O waits, time-sharing, OS overhead, …
- "CPU time" -- include only processor time

Execution time

- ❑ Best to report both wall clock and CPU times
- ❑ Includes system noise effects
  - ➢ Background OS tasks
  - ➢ Virtual to physical page mapping
  - ➢ Random cache mapping and replacement
  - ➢ Variable system load
- ❑ Report both mean and variance (more later)

Linear	☺
Reliable	☺
Repeatable	≈ ☺
Easy to measure	☺
Consistent	☺
Independent	☺

Performance metrics summary

	Clock	MIPS	MFLOPS	SPEC	TIME
Linear					☺
Reliable					≈ ☺
Repeatable	☺	☺	☺	☺	☺
Easy to measure	☺	☺	☺	½ ☺	☺
Consistent	☺			☺	☺
Independent	☺	☺			☺

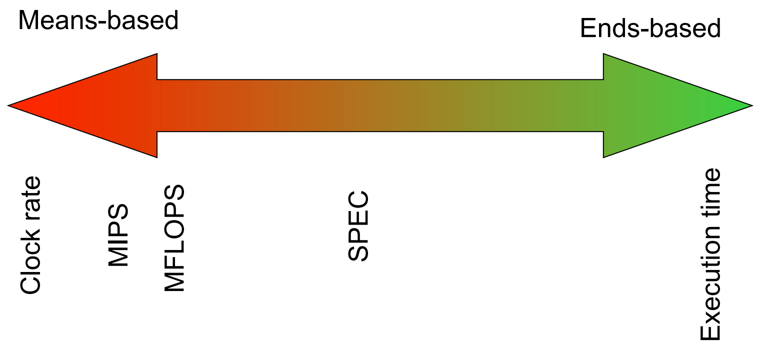
Other metrics

- Response time
  - Elapsed time from request to response
- Throughput
  - Jobs, operations completed per unit time
  - E.g. video frames per second
- Bandwidth
  - Bits per second
- *Ad hoc* metrics
  - Defined for a specific need

Means vs. ends metrics

- Means-based metrics
  - Measure what was done
  - Whether or not it was useful!
    - Nop instructions, multiply by zero, …
  - Produces unreliable metrics
- Ends-based metrics
  - Measures progress towards a goal
  - Only counts what is actually accomplished

Means vs. ends metrics



Means-based

Ends-based

Clock rate

MIPS

MFLOPS

SPEC

Execution time

Important Points

- Metrics can be
  - Counts
  - Durations
  - Sizes
  - Some combination of the above

Important Points

- ❑ Good metrics are
  - Linear
    - More “natural”
  - Reliable
    - Useful for comparison and prediction
  - Repeatable
  - Easy to use
  - Consistent
    - Definition is the same across different systems
  - Independent of outside influences

Exercise 1

- ❑ Feitelson makes the point that in CS experimentation, it is not easy to find “good” metrics
- ❑ Identify one or more metrics commonly used in your area of research
  - If you haven’t yet decided on your research area, pick one of the areas you might be interested in
- ❑ Evaluate the metric(s) based on the criteria discussed in today’s lecture
  - Submit a short report in next week’s class