Performance Evaluation

Johan Garcia

Common Goals of Performance Measurement

- Compare alternatives
- Find “optimal” parameter values
- Identify good/bad performance relative to
  - Peak
  - Expectations
  - History (e.g. previous generations)
  - Competition
- Performance debugging
  - Fix performance problems
  - Find bottlenecks
  - System errors
- Set expectations
  - Tell a customer what is reasonable
- Measure execution time for your program
  - Will it meet requirements?

Solution Techniques

	Technique			
Characteristic	**Analytical**	**Simulation**	**Emulation**	**Measurement**
Flexibility	High	High	Medium	Low
Cost	Low	Medium	Medium+ +	High
Believability	Low	Medium	Medium+ +	High
Accuracy	Low	Medium	Medium+ +	High

Reasonable?

- Never trust results until validated with second solution technique
  - Analytical model
    - Good model?
    - Correct solution technique?
  - Simulation
    - Programming errors?
    - Untested corner cases?
    - Are the random numbers really random?
  - Measurement
    - Measuring perturbs system
    - → Not really the system we want to measure!

Reasonable?

- Sometimes just intuition
  - Measure memory BW
    - 68 Mbyte/sec
  - Clock = 4 ns, 32-bit bus
  - Clock → 1000 Mbytes/sec
  - Is the measurement reasonable?

Performance Measurement “Rules of Thumb”

- Know what you are saying/reading
  - Paper design?
  - Simulation results only?
  - Elapsed time versus CPU time? Time-sharing?
    - “My Mac is faster than your Cray.”
  - Compiler options, OS release, hardware configuration?
  - Slowest component eliminated?
    - No I/O?

Performance Measurement "Rules of Thumb"

- Use the correct type of mean value

	System 1	System 2
Program 1	10 s	36 s
Program 2	250 s	100 s
Program 3	201 s	150 s
Arithmetic mean	154 s	95 s
Geometric mean	79 s	81 s

Performance Measurement "Rules of Thumb"

- Know the program being tested
  - Real program?
    - Includes all I/O?
    - Real inputs?
  - Kernel program?
    - Exclude I/O, loops, subroutines, …
  - Benchmark program?
    - Scaled-down application?
    - Synthetic behavior?

Performance Measurement "Rules of Thumb"

- Define all terms
  - E.g. "% parallelized" means
    - % of all loops parallelized?
    - % of execution time spent in parallel code?
    - % of instructions executed in parallel?
    - Parallel speedup / number of processors?
  - Specify all hardware and software features
    - Options, release numbers, etc.

Performance Measurement "Rules of Thumb"

- Know your tools
  - Timer resolution, accuracy, precision
  - Background noise
  - Compiler does fewer/more optimizations on instrumented code?
  - Perturbations due to measurement
    - *Computer performance measurement uncertainty principle* – "Accuracy is inversely proportional to resolution."

Performance Measurement "Rules of Thumb"

- Bottom line

Are the results reproducible by someone else using only the information you provided?

Performance metrics

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

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

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

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,…

SPEC

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

QUIPS

- Developed as part of HINT benchmark
- Instead of effort expended, use *quality* of solution
- *Quality* has rigorous mathematical definition
- QUIPS = *Quality Improvements Per Second*

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

Performance metrics summary

	Clock	MIPS	MFLOPS	SPEC	QUIPS	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

Networking metrics

- Bandwidth
- Throughput
- Goodput
- Jitter
- Delay
- Response time

Units

- Bandwidth bits/second
- Throughput bits per second or byte per second
- Note base 10 or base 2.
- 2 MB file over 2MBps connection
- 2*8*1024*1024 / 2e6

VERY simple statistics refresher

What Do All of These Means Mean?

- Indices of central tendency
  - Sample mean
  - Median
  - Mode
- Other means
  - Arithmetic
  - Harmonic
  - Geometric

Why mean values?

- Desire to reduce performance to a single number
  - Makes comparisons easy
    - Mine Apple is faster than your Cray!
  - People like a measure of “typical” performance
- Leads to all sorts of crazy ways for summarizing data
  - X = *f*(10 parts A, 25 parts B, 13 parts C, …)
  - X then represents “typical” performance?!

The Problem

- Systems are often specialized
  - Performs great on application type X
  - Performs lousy on anything else
- Potentially a wide range of execution times on one system using different benchmark programs
- Basically, people want *simple* performance metrics.
- *How to (correctly) summarize a wide range of measurements with a single value?*

Index of Central Tendency

- Tries to capture "center" of a distribution of values
- Use this "center" to summarize overall behavior
- Not recommended for real information, but …
  - You will be pressured to provide mean values
    - Understand how to choose the best type for the circumstance
    - Be able to detect bad results from others

Indices of Central Tendency

- Sample mean
  - Common "average"
- Sample median
  - ½ of the values are above, ½ below
- Mode
  - Most common

Indices of Central Tendency

- "Sample" implies that
  - Values are measured from a random process on discrete random variable X
- Value computed is only an approximation of true mean value of underlying process
- True mean value cannot actually be known
  - Would require infinite number of measurements

Sample mean

- Expected value of X = E[X]
  - "First moment" of X
  - x_i = values measured
  - $p_i = Pr(X = x_i)$ = Pr(we measure x_i)

$$E[X] = \sum_{i=1}^{n} x_i p_i$$

Sample mean

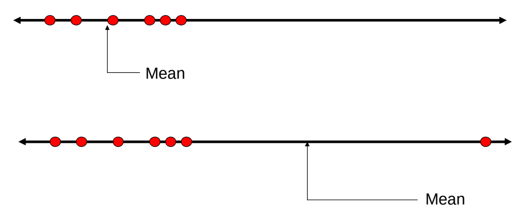
- Without additional information, assume
  - p_i = constant = 1/n
  - n = number of measurements
- ***Arithmetic mean***
  - Common "average"

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Potential Problem with Means

- Sample mean gives equal weight to all measurements
- *Outliers* can have a large influence on the computed mean value
- Distorts our intuition about the *central tendency* of the measured values

Potential Problem with Means



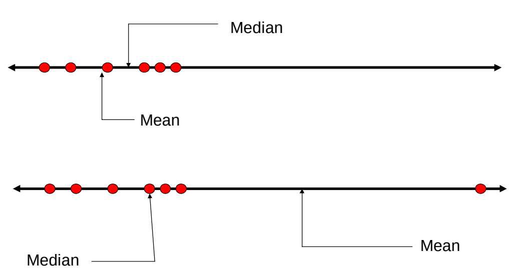
Median

- Index of central tendency with
  - ½ of the values larger, ½ smaller
- Sort n measurements
- If n is odd
  - Median = middle value
  - Else, median = mean of two middle values
- Reduces skewing effect of outliers on the value of the index

Example

- Measured values: 10, 20, 15, 18, 16
  - Mean = 15.8
  - Median = 16
- Obtain one more measurement: 200
  - Mean = 46.5
  - Median = ½ (16 + 18) = 17
- Median give more intuitive sense of central tendency

Potential Problem with Means



Mode

- Value that occurs most often
- May not exist
- May not be unique
  - E.g. “bi-modal” distribution
    - Two values occur with same frequency

Mean, Median, or Mode?

- Mean
  - If the sum of all values is meaningful
  - Incorporates all available information
- Median
  - Intuitive sense of central tendency with outliers
  - What is “typical” of a set of values?
- Mode
  - When data can be grouped into distinct types, categories (*categorical data*)

Mean, Median, or Mode?

- Size of messages sent on a network
- Number of cache hits
- Execution time
- MFLOPS, MIPS
- Bandwidth
- Speedup
- Cost

Different types of means –The Pythagorean means

- Arithmetic mean
  - Often called just ”mean”
  - Suitable when averaging *directly proportional* relationships (e.g. average time (s))
- Harmonic mean
  - Suitable when averaging *inversely proportional* relationships (e.g. rate (something/s))
- Geometric mean
  - Suitable when averaging factors contributing to a product (e.g. interest rates)

Arithmetic Mean

To average a *directly proportional relationship*, the arithmetic mean should be used:

$$\overline{x}_A = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Example: Eight measurements of the execution time of a program have been made, and the corresponding times are {2, 1, 3, 2, 4, 5, 2, 2} seconds.

What was the average execution time?

$$\overline{x}_A = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{8}(2+1+3+2+4+5+2+2) = 2.625 \text{ s}$$

Harmonic mean

To calculate means of *inversely proportional* relationships, we can use the harmonic mean instead:

$$\overline{x}_H = \frac{n}{\sum_{i=1}^{n} 1/x_i}$$

For example, let’s say that we travel 20 km with a car. The first 10 km we have a speed of 70 km/h, and the last 90 km/h, what is the average speed?

$$\overline{x}_H = \frac{n}{\sum_{i=1}^{n} 1/x_i} = \frac{2}{1/70 + 1/90} = 78.75 \text{ km/h}$$

Why does not the arithmetic mean work here?

Geometric mean

The last mean we will consider is the geometric mean:

$$\overline{x}_G = \sqrt[n]{x_1 x_2 \cdots x_i \cdots x_n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n}$$

The geometric mean can be used when calculating the average factor of a product.

When dealing with interest rates, for example, the geometric mean can be used with. Suppose you have an investment that earns 10% the first year, 20% the second, and 5% the third. What is the average rate of return?

$$\overline{x}_G = \sqrt[n]{x_1 x_2 \cdots x_i \cdots x_n} = \sqrt[3]{1.1 * 1.2 * 1.05} \approx 1.115$$

Weighted means

$$\sum_{i=1}^{n} w_i = 1$$

$$\bar{x}_A = \sum_{i=1}^{n} w_i x_i$$

$$\bar{x}_H = \frac{1}{\sum_{i=1}^{n} \frac{w_i}{x_i}}$$

- Standard definition of mean assumes all measurements are equally important
- Instead, choose weights to represent relative importance of measurement *I*
- Easy to miss! Check out correction in the errata of the book!