Assessing and understanding performance

Performance

Why is some hardware better than others for different programs?

What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)

How does the machine's instruction set affect performance?

11.3 Mathematical Preliminaries

- Measures of system performance depend upon one’s point of view.
  - A computer user is most often concerned with *response time*: How long does it take the system to carry out a task?
  - System administrators are usually more concerned with *throughput*: How many concurrent tasks can the system handle before response time is adversely affected?
- These two ideas are related: If a system carries out a task in k seconds, then its throughput is 1/k of these tasks per second.

Computer Performance: TIME, TIME, TIME

Response Time (latency)

— How long does it take for my job to run?

— How long must I wait for the database query?

Throughput

— How many jobs can the machine run at once?

— What is the average execution rate?

Exercise

- *If we add a new machine to the lab what do we increase?*
- *If we upgrade a machine with a new processor what do we increase?*

Execution Time

- Elapsed Time
  - counts everything *(disk and memory accesses, I/O , etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program

Book's Definition of Performance

- For some program running on machine X,

  $$\text{Performance}_X = 1 / \text{Execution time}_X$$

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

  $$\text{Performance}_X \;/\; \text{Performance}_Y = n$$

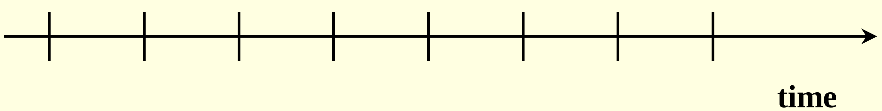
- What if:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles (or number of cycles)

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seco}}{\text{cyc}}$$

- Clock “ticks” indicate when to start activities:



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)

A 4 Ghz. clock has a $\frac{1}{4 \times 10^9} \times 10^{12} = 250 \text{ picoseconds (ps)}$ cycle time

How to Improve Performance

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either (increase or decrease?)

________ the # of required cycles for a program, or

________ the clock cycle time

________ the clock rate.

$$\text{CPU execution time for a program} = \frac{\text{no. of CPU clock cycles for a program}}{\text{Clock rate}}$$

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

  The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  The second has 6 instructions: 4 of A, 1 of B, and 1 of C.

1. Which sequence will be faster? How much?
2. What is the CPI for each sequence?

Amdahl's Law

- The overall performance of a system is a result of the interaction of all of its components.
- System performance is most effectively improved when the performance of the most heavily used components is improved.
- This idea is quantified by Amdahl's Law:

$$S = \frac{1}{(1-f) + \frac{f}{k}}$$

where S is the overall speedup; f is the fraction of work performed by a faster component; and k is the speedup of the faster component.

Amdahl's Law

- Amdahl's Law gives us a handy way to estimate the performance improvement we can expect when we upgrade a system component.
- On a large system, suppose we can upgrade a CPU to make it 50% faster for $10,000 or upgrade its disk drives for $7,000 with a promise that two and a half times the throughput of existing disk will achieve.
- Processes spend 70% of their time running in the CPU and 30% of their time waiting for disk service.
- An upgrade of which component would offer the greater benefit for the same cost?

Amdahl's Law

- The processor option offers a 130% speedup:

$$f = 0.70, \quad k = 1.5 \qquad S = \frac{1}{(1 - 0.7) + 0.7/1.5}$$

- And the disk drive option gives a 122% speedup:

$$f = 0.30, \quad k = 2.5 \qquad S = \frac{1}{(1 - 0.3) + 0.3/2.5}$$

- Each 1% of improvement for the processor costs 10,000/30 = $333,
- and for the disk a 1% improvement costs 7,000/22 = $318.

- So, which improvement is the most cost effective?

System performance

- When we are evaluating system performance, we are most interested in its expected performance under a given workload.
- We use statistical tools that are measures of central tendency.
- The one with which everyone is most familiar is the arithmetic mean (or average), given by:

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

System performance

- The arithmetic mean can be misleading if the data are skewed or scattered.
  - Consider the execution times given in the table below. The performance differences are hidden by the simple average.

Program	System A Execution Time	System B Execution Time	System C Execution Time
v	50	100	500
w	200	400	600
x	250	500	500
y	400	800	800
z	5000	4100	3500
Average	1180	1180	1180

System performance

- If execution frequencies (expected workloads) are known, a weighted average can be revealing.
  - The weighted average for System A is:
  - $50 \times 0.5 + 200 \times 0.3 + 250 \times 0.1 + 400 \times 0.05 + 5000 \times 0.05 = 380.$

Program	Execution Frequency	System A Execution Time	System C Execution Time
v	50%	50	500
w	30%	200	600
x	10%	250	500
y	5%	400	800
z	5%	5000	3500
Weighted Average		380 seconds	695 seconds

System performance

- However, workloads can change over time.
  - A system optimized for one workload may perform poorly when the workload changes, as illustrated below.

Program	Execution Time	Execution Frequency #1	Execution Frequency #2
v	50	50%	25%
w	200	30%	5%
x	250	10%	10%
y	400	5%	5%
z	5000	5%	55%
Weighted Average		380 seconds	2817.5 seconds

System performance

- When comparing the relative performance of two or more systems, the geometric mean is the preferred measure of central tendency.
  - It is the n^{th} root of the product of n measurements.

$$G = \left[x_1 \cdot x_2 \cdot x_3 \cdot \; \cdot \; \cdot x_n \right]^{\frac{1}{n}}$$

- Unlike the arithmetic means, the geometric mean does not give us a real expectation of system performance. It serves only as a tool for comparison.

System performance

- The geometric mean is often uses normalized ratios between a system under test and a reference machine.
  - We have performed the calculation in the table below.

Program	System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
v	50	2	100	1	500	0.2
w	200	2	400	1	600	0.6667
x	250	2	500	1	500	1
y	400	2	800	1	800	1
z	5000	0.82	4100	1	3500	1.1714
Geometric Mean		1.6733		1		0.6898

System performance

- When another system is used for a reference machine, we get a different set of numbers.

Program	System A Execution Time	Execution Time Normalized to C	System B Execution Time	Execution Time Normalized to C	System C Execution Time	Execution Time Normalized to C
v	50	10	100	5	500	1
w	200	3	400	1.5	600	1
x	250	2	500	1	500	1
y	400	2	800	1	800	1
z	5000	0.7	4100	0.8537	3500	1
Geometric Mean		2.4258		1.4497		1

System performance

- The real usefulness of the normalized geometric mean is that no matter which system is used as a reference, the ratio of the geometric means is consistent.
- This is to say that the ratio of the geometric means for System A to System B, System B to System C, and System A to System C is the same no matter which machine is the reference machine.

System performance

- The results that we got when using System B and System C as reference machines are given below.
- We find that 1.6733/1 = 2.4258/1.4497.

System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
Geometric Mean	1.6733		1		0.6898

System A Execution Time	Execution Time Normalized to C	System B Execution Time	Execution Time Normalized to C	System C Execution Time	Execution Time Normalized to C
Geometric Mean	2.4258		1.4497		1

System performance

- The inherent problem with using the geometric mean to demonstrate machine performance is that all execution times contribute equally to the result.
  - So shortening the execution time of a small program by 10% has the same effect as shortening the execution time of a large program by 10%.
- Shorter programs are generally easier to optimize, but in the real world, we want to shorten the execution time of longer programs.
- Also, as the geometric mean is not proportionate. A system giving a geometric mean 50% smaller than another is not necessarily twice as fast!
  - Geometric mean of A = (10 x 20 x 30)^(1/3) = 18.17 mph
  - Geometric mean of B = (20 x 40 x 60)^(1/3) = 34.64 mph

System performance

- The harmonic mean provides us with a way to compare execution times that are expressed as a rate.
- The harmonic mean allows us to form a mathematical expectation of throughput, and to compare the relative throughput of systems and system components.
- To find the harmonic mean, we add the reciprocals of the rates and divide them into the number of rates:

$$H = n \div (1/x_1 + 1/x_2 + 1/x_3 + \ldots + 1/x_n)$$

System performance

- The harmonic mean holds two advantages over the geometric mean.
- First, it is a suitable predictor of machine behavior.
  - Not a normarized ratio as usually done in geometric.
- Second, the slowest rates have the greatest influence on the result, so improving the slowest performance-- usually what we want to do-- results in better performance.
  - More conservative than geometric, e.g. for a set of 5 task
    - Task 1: 10s Task 2: 15s Task 3: 20s Task 4: 25s Task 5: 30s
    - Harmonic mean 5 / [(1/10) + (1/15) + (1/20) + (1/25) + (1/30)] = 16.3s
    - Geometric mean = (10 x 15 x 20 x 25 x 30)^(1/5) = 19.7 s

System performance

- This chart summarizes when the use of each of the performance means is appropriate.

Mean	Uniformly Distributed Data	Skewed Data	Data Expressed as a Ratio	Indicator of System Performance Under a Known Workload	Data Expressed as a Rate
Arithmetic	X			X	
Weighted Arithmetic				X	
Geometric		X	X		
Harmonic				X	X

System performance benchmark

- The objective assessment of computer performance is most critical when deciding which one to buy.
  - For enterprise-level systems, this process is complicated, and the consequences of a bad decision are grave.
- Unfortunately, computer sales are as much dependent on good marketing as on good performance.
- The wary buyer will understand how objective performance data can be slanted to the advantage of anyone giving a sales pitch.

Benchmarking

- Many people erroneously equate CPU speed with performance.
- Measures of CPU speed include cycle time (MHz, and GHz) and millions of instructions per second (MIPS).
- Saying that System A is faster than System B because System A runs at 1.4GHz and System B runs at 900MHz is valid only when the ISAs of Systems A and B are identical.
  - With different ISAs, it is possible that both of these systems could obtain identical results within the same amount of wall clock time.

Benchmarking

- Performance benchmarking is the science of making objective assessments concerning the performance of one system over another.
- Price-performance ratios can be derived from standard benchmarks.
- Example benchmarking report:
  - https://www.spec.org/benchmarks.html#cpu
  - https://www.geekbench.com

End of Lecture