Performance Metrics

1-31-2003

Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news about the hardware market?
- Do you have any questions about the graded quizzes?

Definition of Performance

- I didn't specifically state this last class so I want to now. We define performance as the inverse of execution time for a particular program.
- Given this definition, we can look at relative performance between computers and given performance measurements we can make predictions of how long the program would take to run on other machines.

Amdahl's Law

- One of the most significant things to know about improving performance of computers is that focussing on one aspect typically doesn't help that much.

$$T_f = \frac{T_{affected}}{ImprovementFactor} + T_{unaffected}$$

- The last term really kills you.

Hardware Independent Metrics

- Trying to draw any conclusions about performance without taking the actual hardware into account will utterly fail.
- The book points to an example from the 70s where code size was used as a metric. This is even worse on modern machines where the instruction sets can vary tremendously between architectures.

MIPS as a Metric

- A very popular metric to use is "Millions of Instructions Per Second", or MIPS.
- One big problem with this is that it depends on the CPI or the program being used so the MIPS measurement will vary for different work loads.
- Also, different instructions get different amounts of work done on different machines. In particular, RISC machines often have a higher CPI, but each instruction does less than one CISC instruction would.

Different Types of MIPS

- Another problem with MIPS is that it can have different meanings.
- To escape the problem of varying CPIs, some companies started touting peak MPIs. This is MPI assuming all instructions execute at the speed of the fastest instruction.
- To fix this, the relative MIPs standard came into use, but it only applies to one program and needs a reference computer.

Normalized Arithmetic Mean

- As was mentioned last time, we have a problem of compiling results when we do benchmarking with multiple test programs.
- Instead of general weighting, we could normalize to one particular system. The problem with this is that if you do well in benchmarks the base machine does poorly in, you stand out. Also, results vary by choice of normalizing machine.

Geometric Mean

- To get around the problem of different results with different normalization machines, we can use the geometric mean.

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

- Unfortunately, this mean doesn't work to predict running time. Then again, that probably depends on the program anyway.

My Take

- The problem that I see with this chapter is that they are requesting something that CAN'T exist, but don't really say so.
- They want a metric that is based on execution time and has predictive abilities. However, that can't happen without knowledge of the exact workload and getting the exact workload typically isn't an option.

Minute Essay

- What do you think is the ideal metric for measuring performance? Is there one? If not, what should we do looking at? Are things like the PR rating using by AMD worth anything?
- You should not leave here without turning in assignment #2.