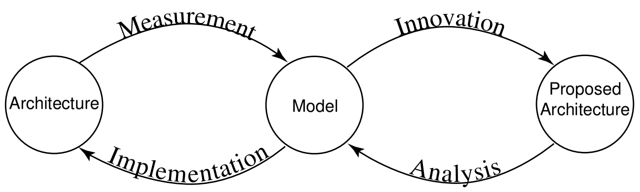
Quantitative Computer Architecture

How to measure, analyze, and specify computer system performance

or

"My computer is faster than your computer!"

Performance Measurement and Analysis in Computer Architecture



What is Performance?

- Execution Time?
- Throughput?
- Of What?
- What is relative performance? How is it specified?

How to measure Execution Time?

```
% time program
... program results ...
160.7u 19.9s 4:15 71%
%
```

- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
- Answer:

Relative Performance

- can be confusing
  - A runs in 12 seconds
  - B runs in 20 seconds
  - A/B = .6 , so A is 40% faster, or 1.4X faster, or B is 40% slower
  - B/A = 1.67, so A is 67% faster, or 1.67X faster, or B is 67% slower
  -
- needs a precise definition

Relative Performance, the Definition

$$\text{Relative Performance} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

- We can remove all ambiguity by always constraining n to be > 1 => machine x is *n* times faster than y.

- your program runs in 5 minutes on a 486, but 2 minutes on a Pentium processor. How much faster is the Pentium processor?
- another program runs in 10 minutes with the standard compiler, but when recompiled with a new compiler, the program runs in 9 minutes. How much faster is the new compiled program?

How to Specify Performance, in summary

- Performance only has meaning in the context of a program or workload (MIPS, MFLOPS???).
- When talking about the performance of a single machine, we talk about “response time” or “throughput.”
- When talking about relative performance, we will say “machine x is n times faster than machine y” based on the ratio of their execution times for a workload.

But What Workload?

- Synthetic workloads
  - whetstone, dhrystone, ...
- toy benchmarks
  - puzzle, quicksort, sieve, ...
- kernels
  - livermore loops, linpack
- real programs

To maximize their efforts, architects will attempt to mirror the decision process of the market. When the market uses poor measurement methodology, we can get poor architectures!

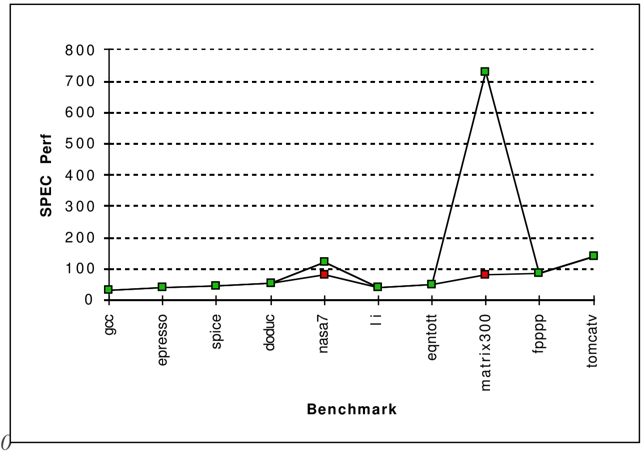
SPEC: System Performance Evaluation Cooperative

- First Round 1989
  - 10 programs yielding a single number
- Second Round 1992
  SpecInt92 (6 integer programs) and SpecFP92 (14 floating point programs)
  - Compiler Flags unlimited.
- Third Round 1995
  - Single flag setting for all programs; new set of programs

SPEC combines real programs with enforced measurement standards.

SPEC First Round

- One program: 99% of time in single line of code
- New front-end compiler could improve dramatically



How to Summarize Performance

- Real workloads typically involve multiple programs, and thus, multiple results.
- Popular benchmarks (e.g., SPEC, livermore loops, ...) involve multiple programs.
- Everyone wants to summarize results with a single number.
- But the summarized result can be dramatically skewed by the method used to combine them.

How to Summarize Performance

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Total time	1001	110	40

Which machine is fastest?

How to Summarize Performance

- Arithmetic Mean

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

- Weighted Arithmetic Mean

$$\sum_{i=1}^{n} Time_i * Weight_i$$ where the sum of the weights is 1.

- Geometric Mean

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

- Harmonic Mean

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

Summarizing Performance

	A	B	C	W(1)	W(2)	W(3)
Program 1	1	10	20	.5	.909	.999
Program 2	1000	100	20	.5	.091	.001
AM/W(1)	500.5	55	20			
AM/W(2)	91.82	18.18	20			
AM/W(3)	2	10.09	20			
GM	31.6	31.6	20			

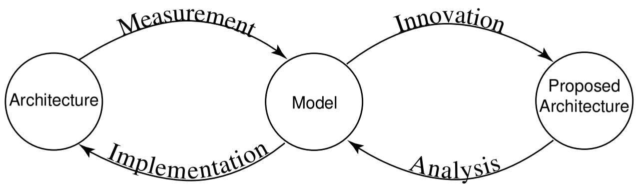
which machine is fastest now?

Summarizing Performance

- Even the unweighted arithmetic mean implies a weighting
- Geometric mean does not necessarily predict execution time for any mix of the programs
- ratios of geometric means never change, and always give equal weight to all benchmarks
- To give unequal weight requires weighted arithmetic mean

Analyzing Performance

- That was all about measuring performance. What tools do we use to analyze (predict) performance in the absence of something to measure?
  - models, equations, queueing theory, mean value analysis, instruction-level simulation, gate-level simulation, ...



Speedup

- Speedup is just relative performance on the same machine with something changed.

$$\text{speedup} = \text{relative performance} = \frac{\text{ET for entire task without change}}{\text{ET for entire task with change}}$$

Amdahl's Law

The impact of a performance improvement is limited by the percent of execution time affected by the improvement

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Make the common case fast!!

- program A runs for 30 seconds, but 5 seconds of that time is just waiting for memory. If we double the speed of the memory subsystem, what is the speedup?
- fp instructions account for 10% of execution time of program B. Should we double the speed of the fp instructions, or speed up integer by 20%?
- How much do we need to speed up the memory to get a 20% improvement in program A?

What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time
= CPU clock cycles / Clock rate

Every conventional processor has a clock with an associated clock cycle time or clock rate.

Every program runs in an integral number of clock cycles.

MHz = millions of cycles/second

X MHz = 1000/X nanoseconds cycle time

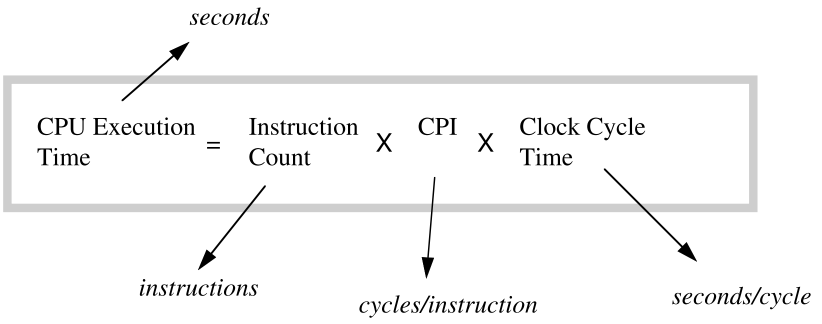
How many clock cycles?

Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)

or

CPI = CPU clock cycles / Instruction count

All Together Now



seconds

CPU Execution Time = Instruction Count X CPI X Clock Cycle Time

instructions

cycles/instruction

seconds/cycle

- 100 MHz processor, program runs in 30 seconds, executing a billion instructions: CPI = ??
- If we reduce CPI to 2.4, ET = ??
- new compiler reduces IC to .8 billion, but increases CPI to 2.6: good or bad?
- 100 MHz pentium has CPI of .9, 200 MHz pentium has a CPI of 1.1 (why?): What’s the speedup for that workload?

Who Affects Performance?

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- programmer
- compiler
- instruction-set architect
- machine architect
- hardware designer
- materials scientist/physicist/silicon engineer

What Affects Performance?

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- pipelining
- superpipelining
- cache
- from CISC to RISC
- superscalar

Key Points

- We need to be precise about how to specify performance.
- Performance is only meaningful in the context of a workload.
- Be careful how you summarize performance.
- Amdahl's law
- ET = IC * CPI * CT