Introduction to
Measurement Methodology
Outline

* Introduction
  - Misuse of the data
  - The Basic Equation (how long did it take)
  - The Mean

* How do we Measure
  - Real Hardware, Simulator, Analytical Model
  - Hardware Instrument, μcode, Software Monitor

* What do we Measure (Benchmarks)
  - Synthetic code
  - Kernels
  - Toy Benchmarks
  - SPEC
  - The Perfect Club
  - Your Relevant Workload

* Serious Abuses
From a Welcoming Address At A Well-Known Conference

Attendance

Why Measure

* Before the fact
  - So we know what to build

* After the fact
  - So we know what to do next time
The Standard Performance Equation

\[ T = \ell \times \text{CPI} \times t_c \]

Path Length
- Algorithm
- Language
- Compiler
- ISA

Clock
- Technology
- Organization

Cycles Per Instruction

ISA
Organization
- Pipelining
- Issue Rate
- Branch Handling
Means

* Arithmetic Mean

\[ A = \frac{1}{n} \sum_{i=1}^{n} P_i \]

* Geometric Mean

\[ G = \sqrt[n]{\prod_{i=1}^{n} P_i} \]

* Harmonic Mean

\[ H = \frac{n}{\frac{1}{n} \sum_{i=1}^{n} \frac{1}{P_i}} \]
How Do We Measure

Degree of Santizing

Real Hardware  Simulation  Analytic Model

Real Hardware
- "Gotchas" Have a chance to get in the way
- Least Flexible
- Fast for doing thorough job

Simulation
- Some effects are missing
- Most Flexible
- Slowest

Analytic Model
- Good for gross effects
- Must be validated
### How Do We Measure (Continued)

**Invasiveness**

<table>
<thead>
<tr>
<th>Hardware Instrumentation</th>
<th>Microcode Instrumentation</th>
<th>Software Monitoring</th>
</tr>
</thead>
</table>

**Hardware Instrumentation**
- Most Expensive
- Non-Invasive
- Least Flexible

**Microcoded Instrumentation**
- Best of Both Worlds
- SPAM

**Software Monitoring**
- Cheap
- Very Invasive
- Most Flexible
SPEC 2006

CINT 2006 (12)

9 in C
3 in C++

perlbench
bzip2
gcc
mcf
sjeng
libquantum
h 264ref
omnet++
a star

CFP 2006 (17)

6 in Fortran
3 in C
4 in Fortran
4 in C++

bwaves (F)
games (F)
milc (C)
zenmp (F)
gromacs (C, F)
cactusADM (C, F)
leslie3d (F)
	namel (C++)
soplex (C++)

GEMS.FDTD (F)
tonto (F)
1bn (C)
wrf (C, F)
sphinx3 (C)
Benchmarks

Rationale: Find a set of programs or program fragments representative of the workload you will be expecting of the machine

Types:

1. The ADD instruction - very old
2. Instruction MIX - Old (Gibson MIX, 1959)
3. Kernels
   - e.g., Livermore Loops
4. Synthetic Benchmarks
   - Parameterized
   - Careful: RRW is not RWR
5. Toy Benchmarks
   - Easy to hand-compile
   - Pretty much in disrepute today
     e.g., Towers of Hanoi
6. SPEC Suite (Systems Performance Evaluation Co-operative)
   - At least common agreement, I Guess!!
7. Real Workload
Bad Ways to Measure Performance
(... and each has been used and reported in the Open Literature)

* Apples & Oranges
  - A Lightly Loaded VAX vs. Counting Simulated Cycles

* Who Gets the Credit
  - The Architecture or the Compiler
  - Example: Berkeley Pascal vs VMS Pascal
  - Algorithm Optimizations
  - Instruction set or register windows (Colwell)

* Choice on Benchmarks
  - Selective
    * Overstates significance of one feature
      e.g. Regularity (Fl. Pt.)
      e.g. Procedure Call Intensive
      e.g. No Floating Point
  - Small
    * 100% Cache, TB Hits
    * No I/O, Context Switch
Play with Statistics

<table>
<thead>
<tr>
<th></th>
<th>Program A</th>
<th>Program B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine 1:</td>
<td>1 unit</td>
<td>2 units</td>
</tr>
<tr>
<td>Machine 2:</td>
<td>2 units</td>
<td>1 unit</td>
</tr>
</tbody>
</table>

Machine 1 is $\frac{2}{1}$ on A, $\frac{1}{2}$ on B

Speed Up is $\frac{1}{2} \left( 2 + \frac{1}{2} \right) = 1.25$

Too Focused on Frequency

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calls</td>
<td>2.5%</td>
<td>21.6%</td>
</tr>
<tr>
<td>MOVL</td>
<td>12.4%</td>
<td>6.8%</td>
</tr>
</tbody>
</table>