Price and Performance

Review

Many levels of abstraction - hardware & software

Instructions and data naturally represented in binary

Important abstraction - Instruction Set Architecture

Five components of computers -

- Processor: datapath and control
- Memory
- Input and Output devices

Technology Trends
Read

- Chapter 2 of Tanenbaum
- Chapter 1 of Hennessy & Patterson

**Hardware Cost**

\[
DieCost = \frac{Wafer\ Cost}{Dies\ per\ Wafer \times Die\ Yield}
\]

\[
Die\ Yield = Wafer\ Yield \times \left\{1 + \frac{Defects\ per\ unit \times Die\ area}{a}\right\}^{-a}
\]

Typical values: Wafer Yield = 90%, Defects per unit area = 1 \(cm^2\), \(a = 2\ or\ 3\)

The bigger the chip, the less likely it is to come out working correctly

Die cost goes up roughly with the cube of the area

\[
IC\ Cost = \frac{Die\ Cost + Testing\ Cost + Packaging\ Cost}{Final\ Test\ Yield}
\]
What is Execution Time?

- wall clock, response, or elapsed time
- CPU time - system & user time
- system performance - elapsed time on an unloaded machine
- CPU performance - user CPU time

Metrics of Performance

- Application Programming Language Compiler
- Answers per month Operations per second Execution time
- ISA
- Millions of instructions or floating point operations per second (MIPS or MFLOPS)
- Datapath Control Function Units
- Clock rate (cycles per second)
- Megabytes per second
Performance Metrics

Throughput

Execution time (T)

Instruction count of program (IC)

MIPS/MFLOPS (MIPS=IC/T or Clock rate (in MHz)/CPI)

Cycles per instruction (CPI)

Clock cycle time - discrete time interval that determines rate at which events take place in hardware and how fast hardware can perform basic functions (clock runs at a constant rate)

Clock rate - inverse of clock cycle time - clock cycles per second

Relating Performance Metrics

\[ CPUTime = \text{Instructions} \times \text{Average CPI} \times \text{Cycletime} \]

<table>
<thead>
<tr>
<th>Instr. Count</th>
<th>CPI</th>
<th>Clock Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instr.Set Arch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relating Performance Metrics

\[ CPUTime = Instructions \times AverageCPI \times Cycletime \]

<table>
<thead>
<tr>
<th></th>
<th>Instr. Count</th>
<th>CPI</th>
<th>Clock Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td>X</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td>Instr.Set Arch.</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

How to Evaluate Performance

Benchmarks used to

- Evaluate different systems or changes to a single system
- Represent a large class of important programs so as to provide a target to improve performance
- Choice of benchmarks important
Programs to Evaluate Processor Performance

- Toy benchmarks - 10-100 lines, e.g., matrix multiply, quicksort
- Synthetic benchmarks - attempt to match average frequencies of real workloads, e.g. Whetstone, Dhrystone
- Kernels - time critical excerpts of programs, e.g., Livermore loops
- Real programs - e.g., gcc, spice.

How to Summarize Performance

- Arithmetic (or weighted arithmetic) mean tracks execution time: \( \text{SUM}(T_i)/n \) or \( \text{SUM}(W_i/T_i) \)
- Harmonic mean (or weighted harmonic mean) of rates (e.g., MFLOPS) tracks execution time: \( n/\text{SUM}(1/R_i) \) or \( n/\text{SUM}(W_i/R_i) \)
- Normalized execution time is handy for scaling performance (e.g., time on reference machine/time on measured machine)
- BUT, do not take the arithmetic mean of normalized execution time, use the geometric mean \( (\prod R_i)^{1/n} \)
- Geometric mean assigns equal weight to all programs regardless of actual execution time
Evolution of SPEC

Standard Performance Evaluation Corporation (SPEC) - consortium to establish standards of evaluation

Provide and emphasize

- Speedup measures
- Throughput measures
- Reproducibility of results

Constantly updated - SPEC89, SPEC92, SPEC95, SPEC2000 - includes integers, floating point, web server, JVM, ... benchmarks

Some SPEC Performance numbers

Intel Pentium II - 450 MHz, .35 micron process, 19.0 SPECint95, 15.2 SPECfp95, $669/-

Compaq Alpha 21264 - 500 MHz, .35 micron process, 27.7 SPECint95, 58.7 SPECfp95

See http://infopad.eecs.berkeley.edu/CIC

For the SPEC benchmarks, See http://www.specbench.org
Amdahl’s Law

Speedup is limited by the part of the program that is not enhanced

Suppose enhancement $E$ accelerates a fraction $F$ of the task by a factor $S$ and the remainder of the task in unaffected.

$$\text{ExTime(with } E) = ((1 - F) + F/S) \times \text{ExTime(without E)}$$

$$\text{Speedup(with E)} = \frac{\text{ExTime(without E)}}{\text{ExTime(with E)}}$$

Summary - Performance

Time is the measure of computer performance!

Need for good benchmarks and performance summaries

Summarize performance using
- the arithmetic (could be weighted) mean of execution time, or
- the geometric mean of execution time ratios - problem - does not track execution time, rewards all relative improvements equally

**Amdahl’s Law**: Speedup is limited by unimproved part of program