Chapter 1

Fundamentals of Computer Design
What we learned last class?

- Cost trends
- Dependability
- Performance metrics
- What program did we choose to measure the performance?
Running Benchmarks

- Key factor: **Reproducibility** by other experimenters.
- **Details, details, and more details !!!** List all assumptions and conditions of your experiments.
  - i.e. program input, version of the program, version of the compiler, optimization level, OS version, main memory size, disk types, etc.
- A system’s software configuration can significantly affect the performance results for a benchmark.
## Comparing Two Machines

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPI</th>
<th>Clock Period</th>
<th>Avg Instruction Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine A</td>
<td>1.2</td>
<td>2 ns</td>
<td></td>
</tr>
<tr>
<td>Machine B</td>
<td>2.5</td>
<td>1 ns</td>
<td></td>
</tr>
</tbody>
</table>

- **CPU Time** = # of instructions executed * avg instruction time
- Assume 1,000,000,000 instructions

- Machine A: 1,000,000,000 * 2.4ns = 2.4 seconds
- Machine B: 1,000,000,000 * 2.5ns = 2.5 seconds

- Which machine is faster? **Machine A**
- How much faster? 2.5 / 2.4 = 1.04 times faster
Comparing Performance

• Often, we want to compare the performance of different machines or different programs. Why?
  • To help engineers understand which is “better”
  • To give marketing a “silver bullet” for the press release
  • To help customers understand why they should buy <my machine>

• Performance and Execution time are reciprocals

Maximizing performance means minimizing response (execution) time

\[
\text{Performance} = \frac{1}{\text{Execution Time}}
\]
Common used phrases

- “Performance of $P_1$ is better than $P_2$” is, for a given work load $L$, $P_1$ takes less time to execute $L$ than $P_2$ does.

$$\text{performance}(P1) > \text{Performance}(P2) \Rightarrow \text{Execution Time}(P1, L) < \text{Execution Time}(P1, L)$$

- “Processor X is n times fast than Y” is

$$n = \frac{\text{Execution time}_Y}{\text{Execution time}_X}$$
Comparing Performance Across Multiple Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Computer A</th>
<th>Computer B</th>
<th>Computer C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Program 2</td>
<td>1000</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Program 3</td>
<td>1001</td>
<td>110</td>
<td>40</td>
</tr>
</tbody>
</table>

- A is 10 times faster than B for program 1
- B is 10 times faster than A for program 2
- A is 20 times faster than C for program 1
- C is 50 times faster than A for program 2
- B is 2 times faster than C for program 1
- C is 5 times faster than B for program 2

Each statement above is correct..., ...but we want to know which machine is the best?
Let's Try a Simpler Example

- Two machines timed on two benchmarks
  - How much faster is Machine A than Machine B?

<table>
<thead>
<tr>
<th></th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>2 seconds</td>
<td>4 seconds</td>
</tr>
<tr>
<td>Program 2</td>
<td>12 seconds</td>
<td>8 seconds</td>
</tr>
</tbody>
</table>

- Attempt 1: ratio of run times, normalized to Machine A times
  - program1: 4/2 program2 : 8/12
  - Machine A ran 2 times faster on program 1, 2/3 times faster on program 2
  - On average, Machine A is \((2 + 2/3) /2 = 4/3\) times faster than Machine B

- It turns this “averaging” stuff can fool us
Example: Second answer

- Two machines timed on two benchmarks
  - How much faster is Machine A than Machine B?

<table>
<thead>
<tr>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>2 seconds</td>
</tr>
<tr>
<td>Program 2</td>
<td>12 seconds</td>
</tr>
<tr>
<td>Program 1</td>
<td>4 seconds</td>
</tr>
<tr>
<td>Program 2</td>
<td>8 seconds</td>
</tr>
</tbody>
</table>

- Attempt 2: ratio of run times, normalized to Machine B times
  - Program 1: $2/4$ program 2: $12/8$
  - Machine A ran program 1 in $1/2$ the time and program 2 in $3/2$ the time
  - On average, $(1/2 + 3/2) / 2 = 1$
  - Put another way, Machine A is 1.0 times faster than Machine B.
Example: Third answer

- Two machines timed on two benchmarks
  - How much faster is Machine A than Machine B?

<table>
<thead>
<tr>
<th>Program</th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 seconds</td>
<td>4 seconds</td>
</tr>
<tr>
<td>2</td>
<td>12 seconds</td>
<td>8 seconds</td>
</tr>
</tbody>
</table>

- Attempt 3: ratio of run times, aggregate (total sum) times,
  - Machine A took 14 seconds for both programs
  - Machine B took 12 seconds for both programs
  - Therefore, Machine A takes 14/12 of the time of Machine B
  - Put another way, Machine A is 6/7 faster than Machine B
Which is Right?

- **Question:**
  - How can we get three different answers?

- **Solution**
  - Because, while they are all reasonable calculations...
  - ...each answers a different question

- We need to be more precise in understanding and posing these performance & metric questions
Arithmetic and Harmonic Mean

- **Total Execution Time:** A Consistent Summary Measure
  - Arithmetic mean is the average of the execution time that tracks total execution time.
    
    $$\frac{1}{n} \sum_{i=1}^{n} Time_i$$
  
  - If performance is expressed as a rate, then the average that tracks total execution time is the harmonic mean
    
    $$\frac{n}{\sum_{i=1}^{n} \frac{1}{Rate_i}}$$
Problems with Arithmetic Mean

- Applications do not have the same probability of being run
- Longer programs weigh more heavily in the average
- For example, two machines timed on two benchmarks

<table>
<thead>
<tr>
<th></th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>2 seconds (20%)</td>
<td>4 seconds (20%)</td>
</tr>
<tr>
<td>Program 2</td>
<td>12 seconds (80%)</td>
<td>8 seconds (80%)</td>
</tr>
</tbody>
</table>

- If we do arithmetic mean, Program 2 “counts more” than Program 1
  - an improvement in Program 2 changes the average more than a proportional improvement in Program 1
- But perhaps Program 2 is 4 times more likely to run than Program 1
Weighted Execution Time

- Often, one runs some programs more often than others. Therefore, we should weight the more frequently used programs' execution time.

\[ \sum_{i=1}^{n} Weight_i \times Time_i \]

- Weighted Harmonic Mean

\[ \frac{1}{\sum_{i=1}^{n} Weight_i} \cdot Rate_i \]
Using a Weighted Sum (or weighted average)

<table>
<thead>
<tr>
<th></th>
<th>Machine A</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>2 seconds (20%)</td>
<td>4 seconds (20%)</td>
</tr>
<tr>
<td>Program 2</td>
<td>12 seconds (80%)</td>
<td>8 seconds (80%)</td>
</tr>
<tr>
<td>Total</td>
<td>10 seconds</td>
<td>7.2 seconds</td>
</tr>
</tbody>
</table>

Allows us to determine relative performance

\[
\frac{10}{7.2} = 1.38
\]

--> Machine B is **1.38 times faster than Machine A**
Another Solution

- Normalize run time of each program to a reference

<table>
<thead>
<tr>
<th></th>
<th>Machine A (ref)</th>
<th>Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>2 seconds</td>
<td>4 seconds</td>
</tr>
<tr>
<td>Program 2</td>
<td>12 seconds</td>
<td>8 seconds</td>
</tr>
<tr>
<td>Total</td>
<td>10 seconds</td>
<td>7.2 seconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Machine A (norm to B)</th>
<th>Machine B (norm to A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>0.5</td>
<td>2.0</td>
</tr>
<tr>
<td>Program 2</td>
<td>1.5</td>
<td>0.666</td>
</tr>
<tr>
<td>Average?</td>
<td>1.0</td>
<td>1.333</td>
</tr>
</tbody>
</table>

- So when we normalize A to B, and average, it looks like A & B are the same.
- But when we normalize B to A, it looks like B is 33% better!
Example on P37

\[
W(B)_1 = \frac{1}{10 \times (1/10 + 1/100)} = 0.909
\]

\[
W(B)_2 = \frac{1}{100 \times (1/10 + 1/100)} = 0.091
\]
Geometric Mean

- Used for relative rate or performance numbers

\[ \text{Relative Rate} = \frac{\text{Rate}}{\text{Rate}_{\text{ref}}} = \frac{\text{Time}_{\text{ref}}}{\text{Time}} \]

- Geometric mean

\[ \sqrt[n]{\prod_{i=1}^{n} \text{Relative Rate}_i} = \frac{\sqrt[n]{\prod_{i=1}^{n} \text{Rate}_i}}{\text{Rate}_{\text{ref}}} \]
# Using Geometric Mean

<table>
<thead>
<tr>
<th>Machine A (norm to B)</th>
<th>Machine B (norm to A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>0.5</td>
</tr>
<tr>
<td>Program 2</td>
<td>1.5</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>0.866</td>
</tr>
</tbody>
</table>

$1.155 = 1/0.8666$!

- **Drawback:**
  - Geometric mean does **NOT** predict run time

- **Normalizes.**
  - Each application now counts equally.
  - **Advantage:** Irrelevance of the reference computer in relative performance
Summary of comparing performance

- **Total execution time or arithmetic mean**
  - consistent result
  - programs in the workload are **NOT** always run an equal number of times

- **Weighted arithmetic mean**
  - take into account the frequency of use in the workload
  - solution depends on which machine is the reference.

- **Normalized Geometric Mean**
  - consistent result, no matter which machine is the reference.
  - Geometric mean does **NOT** predict run time

- **Ideal solution**: *Measure a real workload and weight the programs according to their frequency of execution.*

- **What really matters is how YOUR application performs**
New SPEC Performance Numbers

- **Geometric Mean of 12 (SpecInt) and 14 (SpecFP) Benchmarks**
  - Performance measured against SPARC 10/40

<table>
<thead>
<tr>
<th></th>
<th>Alpha 21264B 833MHz</th>
<th>Intel PentiumIII 1GHz</th>
<th>MIPS R12000 400MHz</th>
<th>HP PA-8600 552MHz</th>
<th>IBM Power 3-II Ultra III 450MHz</th>
<th>Sun 900MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>518</td>
<td>454</td>
<td>320</td>
<td>417</td>
<td>286</td>
<td>438</td>
</tr>
<tr>
<td>FP</td>
<td>590</td>
<td>329</td>
<td>319</td>
<td>400</td>
<td>356</td>
<td>369</td>
</tr>
</tbody>
</table>
New SPEC Performance Numbers

- Geometric Mean of 12 (SpecInt) and 14 (SpecFP) Benchmarks
  - Performance measured against SPARC 10/40
- 2001 Performance Numbers (Microprocessor Report, Aug. 2001)

<table>
<thead>
<tr>
<th></th>
<th>Alpha 21264C</th>
<th>Intel P4</th>
<th>MIPS R14000</th>
<th>HP PA-8600</th>
<th>IBM Power 3-II</th>
<th>Ultra III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>561</td>
<td>599</td>
<td>397</td>
<td>417</td>
<td>286</td>
<td>439</td>
</tr>
<tr>
<td>FP</td>
<td>585</td>
<td>615</td>
<td>362</td>
<td>400</td>
<td>356</td>
<td>369</td>
</tr>
</tbody>
</table>
Topics in Chapter

1.1 Why take this course?
1.2 Classes of computers in current computer market
1.3 Defining computer architecture and What’s the task of computer design?
1.4 Trends in Technology
1.5 Trends in power in Integrated circuits
1.6 Trends in Cost
1.7 Dependability
1.8 Measuring, Reporting and summerizing Perf.
1.9 Quantitative Principles of computer Design
1.10 Putting it altogether
1.9 Quantitative Principles

- Take advantage of parallelism
- Principle of Locality
- Focus on the common case
- Amdahl's Law
- CPU Performance Equation
Take advantage of parallelism

- Most important methods of improving performance
- Parallelism levels
  - System level: use multiple processors
  - Instruction level: Pipelining
  - Operation level:
    - set-associate cache
    - Pipelined function unit

Any other examples?
Principle of Locality

- **Program Property**: Programs tend to reuse data and instructions they have used recently.

- **Rule of thumb**:
  - a program spends 90% of its execution time in only 10% of the code.

- **Temporal locality**:
  - Recently accessed items are likely to be accessed in the near future.

- **Spatial locality**:
  - Items whose addresses are near one another tend to be referenced close together in time.

Any example?
Focus on the common case

- The most important and pervasive principle of computer design.
  - Power, resource allocation, performance, dependability.
  - Rule of thumb: simple is fast.
  - Frequent case is often simpler and can be done faster.

- A fundamental law, called Amdahl’s Law, can be used to quantify this principle.
Amdahl’s Law

The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Example
Amdahl’s law

Execution time after improvement =

\[
\frac{\text{Execution time affected by the improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}
\]

- Increasing the clock rate would not affect memory access time.
- Using a floating point processing unit does not speed integer ALU operations.
Amdahl’s law

- Amdahl's law defines the speedup

\[
\text{Speedup} = \frac{\text{Performance with enhancement}}{\text{Performance without enhancement}} = \frac{\text{Execution time w/o enhancement}}{\text{Execution time with enhancement}}
\]

- If we know two factors:
  - Fraction enhanced: Fraction of computation time in original machine that can be converted to take advantage of the enhancement.
  - Speedup enhanced in enhanced mode: Improvement gained by enhanced execution mode:

\[
\text{Exec time}_{\text{new}} = \text{Exec time}_{\text{old}} \times \left(1 - \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}
\]
**Speedup Equation**

\[
\text{Speedup}_{\text{overall}} = \frac{\text{ExecTime}_{\text{old}}}{\text{ExecTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}
\]

- **Example:**
  A server system with an enhanced CPU (10 times faster than the original one) used for Web serving. Assuming the original CPU is busy with computation 40% of the time and is waiting for I/O 60% of the time.

- **Answer:**
  - \( \text{Fraction}_{\text{enhanced}} = 0.4, \text{Speedup}_{\text{enhanced}} = 10 \)
  - \( \text{Speedup} = \frac{1}{0.6 + 0.4} = \frac{1}{10} = 0.64 \)
  - \( \frac{1}{0.64} = 1.56 \)
Another Example

- Implementations of floating-point (FP) square root vary significantly in performance.

- Two enhancement proposal
  - One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10.
  - The alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6.

- Assuming
  - FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark.
  - FP instructions are responsible for a total of 50% of the execution time for the application.
  - The design team believes that they do both enhancement with the same effort.

- Compare these two design alternatives.
Solution of the example

\[
\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22
\]

\[
\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23
\]
example

Assume:

- An enhancement to a computer that improves some mode of execution by a factor of 10.
- Enhanced mode is used 50% of the time, measured as a percentage of the execution time when the enhanced mode is in use.

Question:

- What is the speedup we have obtained from fast mode?
- What percentage of the original execution time has been converted from fast mode?
Example for dependability-1

Given:

- Failure rate_{system} = 10*1/1000,000 + 1/500,000 + 1/200,000 + 1/100,000
  = 10 + 2 + 5 + 5 + 1
  = 23
  = 23000/1000,000,000

So, Power failure% = 5/23 = 0.22
Example for dependability-2

- Reliability of the power supply can be improved via redundancy:
  - $200000 \rightarrow 830,000,000 \sim 4150X$
  - Reliability improvement
    - $\frac{1}{(1-0.22) + \frac{0.22}{4150}} = 1.28$
What the Amdahl’s Law imply?

- If an enhancement is only usable for a fraction of task, then the total speedup will be no more than $\frac{1}{1-F}$.
- Serve the guide
  - to how much an enhancement will improve performance
  - to how to distribute resource to improve cost-performance
- Useful for comparing
  - the overall system performance of two alternatives,
  - two CPU design alternatives
- We can improve the performance by
  - increasing the Fraction$\text{enhanced}$
  - or, increasing the Speedup$\text{enhanced}$
The CPU Performance Equation

- The "Iron Law" of processor performance:
  - Often it is difficult to measure the improvement in time using a new enhancement directly.

- **CPU Performance Equation**

\[
\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}
\]

\[
\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}
\]
Calculation of CPU Time

CPU time = Instruction count × CPI × Clock cycle time

Or

CPU time = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}

CPU time = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}

Architecture --> Implementation --> Realization
Compiler Designer  Processor Designer  Chip Designer

<table>
<thead>
<tr>
<th>Component of performance</th>
<th>Units of measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU execution time for a program</td>
<td>Seconds for the program</td>
</tr>
<tr>
<td>Instruction count</td>
<td>Instructions executed for the program</td>
</tr>
<tr>
<td>Clock cycles per instructions (CPI)</td>
<td>Average number of clock cycles/instruction</td>
</tr>
<tr>
<td>Clock cycle time</td>
<td>Seconds per clock cycle</td>
</tr>
</tbody>
</table>
Related technologies

- CPU performance is dependent upon 3 characteristics:
  - clock cycle (or rate) \( (CCT) \)
  - clock cycles per instruction \( (CPI) \)
  - instruction count \( (IC) \)

One difficulty: It is difficult to change one in isolation of the others.
Other format of CPU Performance Equation

\[
\text{CPU clock cycles} = \sum_{i=1}^{n} IC_i \times CPI_i
\]

\[
\text{CPU time} = \left( \sum_{i=1}^{n} IC_i \times CPI_i \right) \times \text{Clock cycle time}
\]

\[
\text{CPI} = \frac{\sum_{i=1}^{n} IC_i \times CPI_i}{\text{Instruction count}} = \sum_{i=1}^{n} \frac{IC_i}{\text{Instruction count}} \times CPI_i
\]
Example of CPU time calculation

- Suppose we have made the following measurements:
  - Frequency of FP operations (other than FPSQR) = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
  - Frequency of FPSQR = 2%
  - CPI of FPSQR = 20

- Two design alternatives
  - decrease the CPI of FPSQR to 2
  - decrease the average CPI of all FP operations to 2.5.

- Compare these two design alternatives using the CPU performance equation.
Answer to the question

\[
\text{CPI}_{\text{original}} = \sum_{i=1}^{n} \text{CPI}_i \times \left( \frac{\text{IC}_i}{\text{Instruction count}} \right)
\]

\[
= (4 \times 25\%) + (1.33 \times 75\%) = 2.0
\]

\[
\text{CPI}_{\text{with new FPSQR}} = \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}})
\]

\[
= 2.0 - 2\% \times (20 - 2) = 1.64
\]

\[
\text{CPI}_{\text{new FP}} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625
\]

- Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better.
Compare the result with that from Amdahl’s law

This is the same speedup we obtained using Amdahl’s Law:

\[
\text{Speedup}_{\text{new FP}} = \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}}
\]

\[
= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23
\]
Performance & price-performance

- Performance & price-performance for desktop systems  Fig1.18
- Factors that responsible for the wide variation in price
  - Different levels of expandability
  - Use of cheaper disks and cheaper memory
  - Cost of CPU varies
  - Software differences
  - Lower-end system use PC commodity parts in fans, power supply, support chip sets
  - Commoditization effect
Five desktop and rack-mountable systems

<table>
<thead>
<tr>
<th>Vendor/model</th>
<th>Processor</th>
<th>Clock rate</th>
<th>L2 cache</th>
<th>Type</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Precision Workstation 380</td>
<td>Intel Pentium 4 Xeon</td>
<td>3.8 GHz</td>
<td>2 MB</td>
<td>Desk</td>
<td>$334</td>
</tr>
<tr>
<td>HP ProLiant BL25p</td>
<td>AMD Opteron 252</td>
<td>2.6 GHz</td>
<td>1 MB</td>
<td>Rack</td>
<td>$309</td>
</tr>
<tr>
<td>HP ProLiant ML350 G4</td>
<td>Intel Pentium 4 Xeon</td>
<td>3.4 GHz</td>
<td>1 MB</td>
<td>Desk</td>
<td>$290</td>
</tr>
<tr>
<td>HP Integrity rx2620-2</td>
<td>Itanium 2</td>
<td>1.6 GHz</td>
<td>3 MB</td>
<td>Rack</td>
<td>$520</td>
</tr>
<tr>
<td>Sun Java Workstation W1100z</td>
<td>AMD Opteron 150</td>
<td>2.4 GHz</td>
<td>1 MB</td>
<td>Desk</td>
<td>$214</td>
</tr>
</tbody>
</table>

Expandability: Sun Java workstation < Dell .... < HP BL25p

Cost of processor: die size and L2 cache , processor

Software difference
Price-performance
Messurements-1

- For Servers  Fig 1.17, 1.18
  - TPC-C : standard industry benchmark for OLTP
    - Reasonable approximation
    - Measure total system performance
    - Rules of measurement are very complete
    - Vendors devote significant effort
    - Report both performance & price-performance
Price-performance
Fallacies & pitfalls

- **Pitfall (易犯的错误):**
  - Falling prey to Amdahl’s Law.
    - Not to try to improve some unit of small $F$
  - A single point of failure
    - An example of dependability
    - Don’t forget the single fan!
Fallacy 1: the cost of the processor dominates the cost of the system.

<table>
<thead>
<tr>
<th></th>
<th>Processor + cabinetry</th>
<th>Memory</th>
<th>Storage</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM eServer p5 595</td>
<td>28%</td>
<td>16%</td>
<td>51%</td>
<td>6%</td>
</tr>
<tr>
<td>IBM eServer p5 595</td>
<td>13%</td>
<td>31%</td>
<td>52%</td>
<td>4%</td>
</tr>
<tr>
<td>HP Integrity rx5670 Cluster</td>
<td>11%</td>
<td>22%</td>
<td>35%</td>
<td>33%</td>
</tr>
<tr>
<td>HP Integrity Superdome</td>
<td>33%</td>
<td>32%</td>
<td>15%</td>
<td>20%</td>
</tr>
<tr>
<td>IBM eServer pSeries 690</td>
<td>21%</td>
<td>24%</td>
<td>48%</td>
<td>7%</td>
</tr>
<tr>
<td><strong>Median of high-performance computers</strong></td>
<td>21%</td>
<td>24%</td>
<td>48%</td>
<td>7%</td>
</tr>
<tr>
<td>Dell PowerEdge 2800</td>
<td>6%</td>
<td>3%</td>
<td>80%</td>
<td>11%</td>
</tr>
<tr>
<td>Dell PowerEdge 2850</td>
<td>7%</td>
<td>3%</td>
<td>76%</td>
<td>14%</td>
</tr>
<tr>
<td>HP ProLiant ML350</td>
<td>5%</td>
<td>4%</td>
<td>70%</td>
<td>21%</td>
</tr>
<tr>
<td>HP ProLiant ML350</td>
<td>9%</td>
<td>8%</td>
<td>65%</td>
<td>19%</td>
</tr>
<tr>
<td>HP ProLiant ML350</td>
<td>8%</td>
<td>6%</td>
<td>65%</td>
<td>21%</td>
</tr>
<tr>
<td><strong>Median of price-performance computers</strong></td>
<td>7%</td>
<td>4%</td>
<td>70%</td>
<td>19%</td>
</tr>
</tbody>
</table>
Fallacy

- Benchmarks remain valid indefinitely
  - “Benchmarksmanship” “benchmark engineering”

- The rated mean time to failure of the disks is 1200000 hours or almost 140 years, so disks practically never fail.
  - Real-world MTTF is about 2-4 times worse than manufacturer’s MTTF for ATA disks, 4-8 times worse for SCSI disks
    ( ATA: AT Attachment )
    ( SCSI: Small Computer System Interface )

- Peak performance tracks observed performance.
Homework for Chapter 1

- Read the section of 1.7 (Putting it all together) and 1.9 (Fallacies and Pitfalls)

- Question:
  - 3rd e-Edition: 1.1
  - 4th e-Edition: P62, 1.13, 1.14

- Due time: March 16, 2009, 9:30am.

- Do the homework in English.

- Recommend to submit it in both paper exercise-book (in class) and e-version named as STID_name_hw1 (to ftp), NOT via email.

- NOTES: do the homework not only for 10% score, but for your understanding of the contents.
Over now! Have a nice meal!