“Eighty percent of success is showing up.”

Woody Allen

Quiz: Bad Chinese Food Challenge

You are given 100 marbles, 50 red and 50 yellow.

There are two empty identical baskets.

Distribute all the marbles in the two baskets such that if I randomly, without looking, reach into one of the baskets and pull out one marble, I have the highest probability of pulling out a yellow marble. Give the approximate probabilities.

Basket #1: ___ red ___ yellow
Basket #2: ___ red ___ yellow
Name: ___________________

Probability of picking yellow: ______

“What does showbusiness teach you?
It teaches you that design is war;
it is a power struggle between the produces, directors, authors, everyone who wants to be involved.”

Ted Nelson (Internet Pioneer)

“Everyone wants to direct.”

Hollywood proverb

Three aspects of design

ISA
- Programmer visible instruction set
organization
- High-level design (chip + chipset)
+ hardware
- Detailed design (VLSI, package)

architecture

From market to silicon

Design process

Marketplace
- Needs of users
Existing application
- Legacy code, ISA
Functional requirements of architecture
- Backend compatibility
Implementation of architecture
- Design optimization

Lessons from the market

- Separate architecture design and technology
  - Increases ability to adapt arch family
- Backward compatibility necessary to compete
  - Must support existing apps
- Marketplace demands performance – cheap
  - Users want speed, but will only pay so much

price ➔ Arch design ➔ performance
Quantifying Cost and Performance

“All problems in Computer Science can be solved by another level of indirection.”
Butler Lampson

“All performance problems can be solved by removing a level of indirection.”
M. Haertel

Quantifying Cost

Need to understand cost of computers ~ IC cost.

Number of square dies that can be placed on a round wafer:
\[ \text{dies per wafer} = \left( \frac{\pi \times \text{diameter of wafer}}{\text{diameter of die}} \right)^2 \times \text{die area} \]

Percentage of dies per wafer free from manufacturing defects:
\[ \text{die yield} = \left( \frac{1 - \text{defects per area} \times \text{die area}}{\alpha} \right) \]

Number of ‘good’ dies per wafer:
\[ \text{good dies per wafer} = \text{dies per wafer} \times \text{die yield} \]

Example

- MIPS 4600
  - 77 mm² die area
  - Wafer cost = $3200

(a) Find # good chips for 20-cm wafer diameter
1 defect per cm², wafer yield 95%, α=3.0

\[ \text{77 mm}^2 = 77 \text{cm}^2 \times \text{die area} \]
\[ \text{dies per wafer} = \left( \frac{\pi \times 20}{\text{diameter of die}} \right)^2 \times \text{die area} = 357 \]
\[ \text{die yield} = 95\% \times \left( \frac{12.72}{3.0} \right)^2 = 478 \]
\[ \text{good dies per wafer} = 357 \times 478 = 170 \]

(b) Find cost per projected good die (untested, unplugged)

\[ \text{die cost} = \frac{\text{wafer cost}}{\text{good dies per wafer}} = \frac{\$3200}{170} = \$18.82 \]

(c) Find IC cost

\[ \text{testing cost per die} = \frac{\text{cost per unit time} \times \text{testing time}}{\text{die yield}} = \frac{5.083 \text{sec} \times \text{10 sec}}{478} = \$1.77 \]

MIPS 4600: PQFP package
- Pin count < 220 (MIPS 4600 has 208 pins)
- Package cost = $12
- Test time = 10 sec
- Test cost per hour = $300
- Assume final test yield = 1

\[ \text{IC cost} = \text{die cost} \times \text{package cost} = \$18.82 \times \$12 = \$225.84 \]
System cost

• Cabinet (6%)
  - Sheet metal, plastic, power supply, fan
  - Cables, nuts, bolts, shipping box
• Processor board (37%)
  - Processor (22%)
  - DRAM (5%), Video card (5%), mainboard (5%)
• I/O devices (37%)
  - Keyboard/mouse (3%)
  - Monitor (19%)
  - HD/DVD/etc (15%)
• Software (20%)

Performance Example

• My car (X) travels a distance of 1 in one hour
  - Time between start and completion of event is 1 hour
  - Execution time
• Your car (Y) travels a distance of 1 in two hours
• Intuition: my car is twice as fast as your car
• Intuition assumes performance = speed

Is intuition correct?

\[
\frac{\text{execution time of your car (Y)}}{\text{execution time of my car (X)}} = n
\]

Implies X performs n times better than Y

\[
\text{performance} = \frac{1}{\text{execution time}}
\]

\[
n = \frac{\text{execution time of Y}}{\text{execution time of X}} \cdot \frac{\text{speed of Y}}{\text{speed of X}} \cdot \frac{\text{performance of X}}{\text{performance of Y}}
\]

Speed is one measure of performance, throughput is another

Speed or throughput

• Response time: time it takes to complete one task
• Throughput: time it takes to complete many tasks
• Depends on user’s preference, but
  - Must be consistent (same workload)
  - Must be in units of time (observed period)
• Types of time
  - Wall clock time, response time, elapsed time, execution time
  - User CPU time
  - System CPU time

Benchmarking

• Evaluation of machine performance for a given workload
• Methods
  - Real programs
  - Modified apps (cin2000, cfp2000 of SPEC)
  - Kernels
    - Livermore Kernels, LinPACK
  - Toy Benchmarks
    - Sieve of Eratosthenes (highest common factor), quicksort
  - Synthetic Benchmarks
    - Whetstone, Dhrystone
• Problems
  - Benchmarks drive vendors to improve benchmarks
  - How do we compare a suite of applications

Time waits for no one...

• Golden Rule of Performance Evaluation
  - Ensure measured results are deterministic (reproducible)
  - Dedicated runs, unloaded system
• Corollary to the golden rule
  - When evaluating CPU performance, ignore system performance (I/O) in this class
    - CPU performance = user CPU time
    - System performance = elapsed time including I/O
Lies, darn lies, and statistics

• To minimize
  - Cheating on results
  - Apples to oranges
• Define methods for comparisons (benchmarks)
• How to fairly compare multiple machines and codes

<table>
<thead>
<tr>
<th>Computer</th>
<th>Program A</th>
<th>Program B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Computer 1</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Computer 2</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Computer 3</td>
<td>130</td>
<td>200</td>
</tr>
</tbody>
</table>

C3 is 1.5x faster than C2 for Prog A
C3 is 10x faster than C2 for Prog B
C2 is 10x faster than C1 for Prog A
C1 is 10x faster than C2 for Prog B
C1 is 1.5x faster than C2
C1 is 2.4x faster than C2

All quantifiable methods should be proportional to total execution time – the great comparator – ensuring reflection of real performance.

Let’s get mean

Arithmetic mean
\[ \frac{1}{n} \sum_{i=1}^{n} \text{time} \]
Arithmetic mean (weighted)
\[ \frac{1}{n} \sum_{i=1}^{n} \text{weight} \times \text{time} \]

<table>
<thead>
<tr>
<th>Computer</th>
<th>Program A</th>
<th>Program B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Computer 1</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Computer 2</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Computer 3</td>
<td>130</td>
<td>200</td>
</tr>
</tbody>
</table>

Normalizing execution time: normalize to a particular machine by dividing all execution times by chosen machine’s time.

Example: Program P1 has the following execution times:
- On machine A: 10 secs
- On machine B: 100 secs
- On machine C: 150 secs
Normalized to A: A=1, B=10, C=15
Normalized to B: A=1, B=1, C=1.5

Getting Meaner

Taking the average of the normalized times

Normalized arithmetic mean
\[ \frac{1}{n} \sum_{i=1}^{n} \text{execution time ratio} \]
Normalized geometric mean
\[ \prod_{i=1}^{n} \text{execution time ratio} \]

Be normal

Normalized example
Geometric vs. Arithmetic

- Arithmetic mean
  - Provides weighted average
  - Pros: proportional to overall execution time
  - Cons:
    - Can be rigged easily (disproportionate problem size)
    - Cannot use with normalizing

- Normalized Geometric mean
  - Provides relative performance of machines to ref
  - Pros: same results regardless of ref machine
  - Cons:
    - Not proportional to overall execution time
    - Large % change in small overall time contributor can skew

Suggestions
- Weight programs according to their actual frequency
- Use problem size to pre-normalize program execution time
- Combine approaches: summary of simple means and relative performance to base machine