Performance evaluation

- It’s an everyday process…

- When you buy food
  - Same quantity, then you look at cost
  - Same cost, then you look at quantity
  - (Assuming same quality!)

- When you buy a notebook
  - You give weights to different features
    - LCD size?
    - CPU speed?
    - Main memory size?
    - Battery life?
  - How do you quantify these and come up with a single score that will guide your decision?

Performance evaluation

- Often we face an optimization problem…

- When cost is fixed
  - Maximize performance (and features)!

- When performance target is achieved
  - Minimize cost!

- Give me the highest-performance computer! Don’t care about cost…
  - Supercomputers are of this class

Computer performance

- Have you cared about performance seriously when buying a computer?
  - What were the most important factors when you bought a new PC?

- Defining computer performance may not be that simple…
  - What do you mean when you say a computer has better performance than another?

- We need a common metric
  - Remember, however, than a single metric will not say everything about a computer
  - We may need multiple metrics!

- Response time vs. throughput
Response time vs. throughput

<table>
<thead>
<tr>
<th>Plane</th>
<th>DC to Paris</th>
<th>Top Speed (mph)</th>
<th>Passengers</th>
<th>Throughput (p·mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 hours</td>
<td>610</td>
<td>470</td>
<td>286,700</td>
</tr>
<tr>
<td>BAD/Sud Concorde</td>
<td>3 hours</td>
<td>1350</td>
<td>132</td>
<td>176,200</td>
</tr>
</tbody>
</table>

Which has higher performance?
- Time to deliver 1 passenger
- Time to deliver 400 passengers

Time for 1 job called
- This is called “response time” or “execution time” or “latency”
- Time/job

Jobs per day called
- This is called “throughput” or “bandwidth”
- Jobs/time

Conventions

- Throughput
  - E.g., # of transactions processed per second
  - Larger is better

- If we are primarily concerned with response time, performance can be defined as 1/T where T is the response time
  - Smaller T is better
  - That is, larger (1/T) is better

- Machine A is N times faster than B
  - \( N = \frac{\text{performance}(A)}{\text{performance}(B)} = \frac{\text{time}(B)}{\text{time}(A)} \)
  - When we present this statement, we want to have \( N > 1 \) to eliminate confusion

Response time vs. throughput

Time of Concorde vs. Boeing 747
- Concord is faster
  - (6.5 hours / 3 hours) or 2.2 times faster

Throughput of Boeing 747 vs. Concorde
- 286,700 p·mph / 176,200 p·mph
- 1.6 times higher (faster)

“Boeing 747 is 1.6 times (or 60%) faster than Concorde in terms of throughput”

“Concorde is 2.2 times (or 120%) faster than Boeing 747 in terms of flying time (response time)”

We will focus primarily on execution time of a single job for the remaining discussions

Time in a computer

When you time a program, what do you measure?
- Total time to complete a task
- CPU time
  - Overheads include disk accesses, memory accesses, I/O activities, OS overheads, overheads from other programs, ...
  - “Real time,” “response time,” “elapsed time,” ...

We are interested in time spent by CPU on your program only – there are multiple programs running at the same time
- “CPU execution time” or “CPU time”
  - Often divided into system CPU time (in OS) and user CPU time (in user program)
Time in a computer

- In terms of seconds?
- CPU time: computers are constructed using digital circuitry synchronized at common “clock” ticks
  - E.g., Rowers and a captain
  - Clock ticks determine when events take place
- Clock is an endless, repetitive up & down pattern
- Clock cycle time = period of a clock cycle (up and down)
  - CCT = 1 / clock rate (or clock frequency)
  - 1ns if 1GHz clock is used
  - 0.5ns if 2GHz clock is used
  - 0.25ns if 4GHz clock is used
  - ___ if 5GHz clock is used

Measuring time in terms of clocks

- CPU execution time can be represented with # of clocks or clock ticks
  - E.g., it takes 1,000,000 clock ticks to finish program A
- If we know the clock cycle time (equivalently clock rate), we can easily calculate the actual time
  - Time = (# of clock ticks) × (clock cycle time)
  - Time = (# of clock ticks) / (clock rate)
- # of clock ticks
  - Has to do with how many instructions are executed in the program
  - Has to do with how many clock ticks are needed to finish a single instruction on average
- # clock ticks = # instructions (also called “instruction count”) × (clocks per instruction or CPI)
- Time = (# of clock ticks) × (clock cycle time)
  - Time = IC × CPI × CCT
- IC may not change if two machines use the same instruction set architecture
What impacts time?

- Time = IC × CPI × CCT

  - IC – # of instructions
    - Instruction set architecture (ISA)
    - Compiler used
  
  - CPI (average clock per instruction)
    - Microarchitecture (e.g., pipelining, cache size, …)
    - (Compiler)
    - (ISA)
  
  - CCT (equivalently clock rate)
    - Technology & circuit optimization
    - Microarchitecture, esp. pipelining
    - (ISA)

Workload

- A set of programs run on a computer would form a workload
  - Actual collection of applications
  - Synthetic programs
  - Kernel loops
  - …

- To compare two computer systems, a user would typically compare the execution time of the workload on the two machines

Benchmark suites

- A benchmark suite is a set of applications relevant for performance evaluation

  - SPEC (standard performance evaluation corporation)
    - www.spec.org
    - CPU benchmarks
    - Server benchmarks
    - Graphics benchmarks
    - …

  - EEMBC (embedded microprocessor benchmark consortium)
    - Automotive
    - Consumer
    - Network
    - Telecom
    - Office
    - …

Synthetic benchmarks

- A synthesized program intended to mimic real programs
  - \( A = B \times C + D/E + F; \)

- One may write a synthetic benchmark to exercise a specific sequence of actions
  - A program that maximally exercises the multiplication unit in CPU
Kernel programs (or loops)

- An important portion of a real program
  - Quicksort
  - Matrix multiplication

- Real programs usually consist of many loops and functions and it may be quite difficult to understand and interpret any performance evaluation results if we are only given summary information

- In the case kernel programs, we immediately understand what’s going on in the programs

Summarizing performance

<table>
<thead>
<tr>
<th>Program 1 (sec)</th>
<th>Computer A</th>
<th>Computer B</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 2 (sec)</td>
<td>1000</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Total time (sec)</td>
<td>1001</td>
<td>110</td>
<td></td>
</tr>
</tbody>
</table>

- Computer A is 10 times faster than B for program 1
- Computer B is 10 times faster than A for program 2

Although the above statements are correct individually, they present a confusing picture...

Arithmetic mean

- Assume that we have N programs

- Arithmetic mean (AM) = (∑ Timeᵢ)/N

- Weighted arithmetic mean = ∑ Timeᵢ×Wᵢ, ∑ Wᵢ = 1

- AM is a special case of weighted AM where Wᵢ = 1/N

Normalized time

<table>
<thead>
<tr>
<th>Program</th>
<th>Computer A</th>
<th>Computer B</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>1 (0.1)</td>
<td>10 (1)</td>
<td>0.1</td>
</tr>
<tr>
<td>Program 2</td>
<td>1 (10)</td>
<td>0.1 (1)</td>
<td>10</td>
</tr>
<tr>
<td>Sum</td>
<td>2 (10.1)</td>
<td>10.1 (2)</td>
<td></td>
</tr>
</tbody>
</table>

- Normalized time maintains the ratio of the original times
- However, depending on the reference, the result can be different!
- Summing (e.g., arithmetic mean) the normalized times may not produce the same comparison result
Dealing w/ ratios

<table>
<thead>
<tr>
<th>Computer A</th>
<th>Computer B</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Program 2</td>
<td>1 (1000)</td>
<td>0.1 (100)</td>
</tr>
<tr>
<td>GM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- When performance ratios are given, geometric mean (GM) can summarize the result more consistently
- \( GM = \left( \prod \text{Ratio} \right)^{1/N} \)
  - \( GM(X)/GM(Y) = GM(X/Y) \)
  - Taking either the ratio of GM or GM of the ratios yields the same result

Case study: SPEC benchmark

- SPEC CPU2000 benchmark
  - 12 integer programs
  - 14 floating-point (computation-intensive) programs
- To obtain a “SPECmark”
  - Run each program on the target machine
  - Get each program’s performance ratio by dividing the pre-provided execution time (of an old SUN workstation) with the execution time obtained
  - Get GM of the ratios from all the programs

Summarizing performance

Amdahl’s law

- Your optimization technique is usually applicable to only a limited portion of program execution
  - E.g., larger cache, improved CPU frequency, improved FSB frequency, …
- \( \text{Time}_{\text{improved}} = \text{Time}_{\text{unaffected}} + \frac{\text{Time}_{\text{affected}}}{(\text{improvement factor})} \)
- Make the common case fast!
Amdahl’s law

- If we extend this basic form, we have
  - Time_after = T_1/N_1 + T_2/N_2 + \ldots + T_M/N_M where T_i is a portion of Time_before and N_i is the improvement ratio (or speedup) applied to the program execution portion.

Fallacies and pitfalls

- Pitfall 1: Expecting the improvement of one aspect of a computer to increase performance by an amount proportional to the size of the improvement.
- Pitfall 2: Using a subset of the performance equation as a performance metric.

Summary

- Performance evaluation is an important stage of any design and engineering process.
- We are interested in measuring computer performance where the goal is
  - Software improvement
  - Hardware improvement
  - Or both
- We need relevant metrics and a method to summarize measurements.
- Latency vs. throughput.

Response time (latency) = time to finish a given job.

Throughput = # of jobs done in a second.

Time = # of clock cycles \times clock cycle time.

# of clock cycles = # of instructions \times CPI.
Summary

- Workloads used in performance evaluation are typically drawn from real applications
- Synthetic or kernel benchmarks are often useful
- A benchmark suite is a set of applications designed to aid performance evaluation tasks
  - Easier experiments, fair comparison, consistent reporting, …

- Summarizing results
  - Use arithmetic mean (AM) for measured times
  - Use geometric mean (GM) for ratios
- Amdahl’s law
  - Dictates the actual performance improvement due to a speedup factor applied to a limited portion of a program