Introduction

• Background: CS 3810 or equivalent, based on Hennessy and Patterson’s Computer Organization and Design


• Topics
  ➢ Measuring performance/cost/power
  ➢ Instruction level parallelism, dynamic and static
  ➢ Memory hierarchy
  ➢ Multiprocessors
  ➢ Storage systems and networks
Organizational Issues

• Office hours, MEB 3414, by appointment

• TA: Kenneth Williams; TA office hrs: TBA

• Special accommodations, add/drop policies (see class webpage)

• Class web-page and class mailing list at http://www.eng.utah.edu/~cs6810

• Grades:
  ➢ Two midterms, 25% each
  ➢ Homework assignments, 50%, you may skip one
  ➢ No tolerance for cheating
Lecture 1: Measuring Performance

• How do we conclude that System-A is “better” than System-B?

• Topics: (Sections 1.1, 1.4, 1.5, 1.8)
  - Technology trends
  - Performance summaries
  - Performance equations
15x performance growth can be attributed to architectural innovations.
Where Are We Headed?

• Modern trends:

  ➢ Clock speed improvements are slowing
    ▪ power constraints
    ▪ already doing less work per stage

  ➢ Difficult to further optimize a single core for performance

  ➢ Multi-cores: each new processor generation will accommodate more cores
Processor Technology Trends


- Transistor density increases by 35% per year and die size increases by 10-20% per year… more cores!

- Transistor speed improves linearly with size (complex equation involving voltages, resistances, capacitances)… clock speed improvements!

- Wire delays do not scale down at the same rate as logic delays… the Pentium 4 has pipeline stages for wire delays
Technology Trends

• DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases

• Disk density improves by 100% every year, latency improvement similar to DRAM

• Networks: primary focus on bandwidth; 10Mb → 100Mb in 10 years; 100Mb → 1Gb in 5 years
Power Consumption Trends

• Dyn power $\propto$ activity x capacitance x voltage$^2$ x frequency

• Capacitance per transistor and voltage are decreasing, but number of transistors and frequency are increasing at a faster rate

• Leakage power is also rising and will soon match dynamic power

• Power consumption is already between 100-150W in high-performance processors today
Measuring Performance

• Two primary metrics: wall clock time (response time for a program) and throughput (jobs performed in unit time)

• To optimize throughput, must ensure that there is minimal waste of resources

• Performance is measured with benchmark suites: a collection of programs that are likely relevant to the user
  - SPEC CPU 2006: cpu-oriented programs (for desktops)
  - SPECweb, TPC: throughput-oriented (for servers)
  - EEMBC: for embedded processors/workloads
Summarizing Performance

• Consider 25 programs from a benchmark set – how do we capture the behavior of all 25 programs with a single number?

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys-A</td>
<td>10</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>Sys-B</td>
<td>12</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>Sys-C</td>
<td>8</td>
<td>8</td>
<td>30</td>
</tr>
</tbody>
</table>

- Total (average) execution time
- Total (average) weighted execution time
- Average of normalized execution times
- Geometric mean of normalized execution times
AM Example

• We fixed a reference machine X and ran 4 programs A, B, C, D on it such that each program ran for 1 second

• The exact same workload (the four programs execute the same number of instructions that they did on machine X) is run on a new machine Y and the execution times for each program are 0.8, 1.1, 0.5, 2

• With AM of normalized execution times, we can conclude that Y is 1.1 times slower than X – perhaps, not for all workloads, but definitely for one specific workload (where all programs run on the ref-machine for an equal #cycles)

• With GM, you may find inconsistencies
GM Example

<table>
<thead>
<tr>
<th></th>
<th>Computer-A</th>
<th>Computer-B</th>
<th>Computer-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1 sec</td>
<td>10 secs</td>
<td>20 secs</td>
</tr>
<tr>
<td>P2</td>
<td>1000 secs</td>
<td>100 secs</td>
<td>20 secs</td>
</tr>
</tbody>
</table>

Conclusion with GMs: (i) A=B
(ii) C is ~1.6 times faster

• For (i) to be true, P1 must occur 100 times for every occurrence of P2

• With the above assumption, (ii) is no longer true

Hence, GM can lead to inconsistencies
Summarizing Performance

• GM: does not require a reference machine, but does not predict performance very well
  ➢ So we multiplied execution times and determined that sys-A is 1.2x faster…but on what workload?

• AM: does predict performance for a specific workload, but that workload was determined by executing programs on a reference machine
  ➢ Every year or so, the reference machine will have to be updated
Normalized Execution Times

• Advantage of GM: no reference machine required

• Disadvantage of GM: does not represent any “real entity” and may not accurately predict performance

• Disadvantage of AM of normalized: need weights (which may change over time)

• Advantage: can represent a real workload
CPU Performance Equation

• CPU time = clock cycle time x cycles per instruction x number of instructions

• Influencing factors for each:
  ➢ clock cycle time: technology and organization
  ➢ CPI: organization and instruction set design
  ➢ instruction count: instruction set design and compiler

• CPI (cycles per instruction) or IPC (instructions per cycle) can not be accurately estimated analytically
Measuring System CPI

• Assume that an architectural innovation only affects CPI

• For 3 programs, base CPIs: 1.2, 1.8, 2.5
  CPIs for proposed model: 1.4, 1.9, 2.3

• What is the best way to summarize performance with a single number? AM, HM, or GM of CPIs?
Example

• AM of CPI for base case = \(1.2\, \text{cyc} + 1.8\, \text{cyc} + 2.5\, \text{cyc}\) 
  \[\text{instr} \quad \text{instr} \quad \text{instr}\]

  5.5 cycles is execution time if each program ran for one instruction – therefore, AM of CPI defines a workload where every program runs for an equal #insts

• HM of CPI = \(1 / \text{AM of IPC}\); defines a workload where every program runs for an equal number of cycles

• GM of CPI: warm fuzzy number, not necessarily representing any workload
“Speedup” is a ratio

“Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline

A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
- What is the speedup?
- What is the percentage increase in performance?
- What is the reduction in execution time?
Title

• Bullet