Lecture 2: Performance

• Today’s topics:
  - Performance trends and equations

• Reminders: YouTube videos, canvas, and class webpage: http://www.cs.utah.edu/~rajeev/cs3810/
Important Trends

- Historical contributions to performance:
  1. Better processes (faster devices) ~20%
  2. Better circuits/pipelines ~15%
  3. Better organization/architecture ~15%

In the future, bullet-2 will help little and bullet-1 will eventually disappear!

<table>
<thead>
<tr>
<th>Year</th>
<th>Pentium</th>
<th>P-Pro</th>
<th>P-II</th>
<th>P-III</th>
<th>P-4</th>
<th>Itanium</th>
<th>Montecito</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>3.1M</td>
<td>5.5M</td>
<td>7.5M</td>
<td>9.5M</td>
<td>42M</td>
<td>300M</td>
<td>1720M</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>60M</td>
<td>200M</td>
<td>300M</td>
<td>500M</td>
<td>1500M</td>
<td>800M</td>
<td>1800M</td>
</tr>
</tbody>
</table>

Moore’s Law in action

At this point, adding transistors to a core yields little benefit
Processor Technology Trends


• Transistor density increases by 35% per year and die size increases by 10-20% per year… functionality improvements!

• Transistor speed improves linearly with size (complex equation involving voltages, resistances, capacitances)

• Wire delays do not scale down at the same rate as transistor delays
Memory and I/O 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.
Performance Metrics

• Possible measures:
  ▪ response time – time elapsed between start and end of a program
  ▪ throughput – amount of work done in a fixed time

• The two measures are usually linked
  ▪ A faster processor will improve both
  ▪ More processors will likely only improve throughput
  ▪ Some policies will improve throughput and worsen response time

• What influences performance?
Execution Time

Consider a system X executing a fixed workload W

Performance_X = 1 / Execution time_X

Execution time = response time = wall clock time
- Note that this includes time to execute the workload as well as time spent by the operating system co-ordinating various events

The UNIX “time” command breaks up the wall clock time as user and system time
Speedup and Improvement

• System X executes a program in 10 seconds, system Y executes the same program in 15 seconds

• System X is 1.5 times faster than system Y

• The speedup of system X over system Y is 1.5 (the ratio)

• The performance improvement of X over Y is $1.5 - 1 = 0.5 = 50\%$

• The execution time reduction for the program, compared to Y is $(15 - 10) / 15 = 33\%$
  The execution time increase, compared to X is $(15 - 10) / 10 = 50\%$
A Primer on Clocks and Cycles
Performance Equation - I

CPU execution time = CPU clock cycles × Clock cycle time
Clock cycle time = 1 / Clock speed

If a processor has a frequency of 3 GHz, the clock ticks 3 billion times in a second – as we’ll soon see, with each clock tick, one or more/less instructions may complete

If a program runs for 10 seconds on a 3 GHz processor, how many clock cycles did it run for?

If a program runs for 2 billion clock cycles on a 1.5 GHz processor, what is the execution time in seconds?
Performance Equation - II

CPU clock cycles = number of instrs x avg clock cycles per instruction (CPI)

Substituting in previous equation,

Execution time = clock cycle time x number of instrs x avg CPI

If a 2 GHz processor graduates an instruction every third cycle, how many instructions are there in a program that runs for 10 seconds?
Factors Influencing Performance

Execution time = clock cycle time x number of instrs x avg CPI

- Clock cycle time: manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)

- Number of instrs: the quality of the compiler and the instruction set architecture

- CPI: the nature of each instruction and the quality of the architecture implementation
Example

Execution time = clock cycle time x number of instrs x avg CPI

Which of the following two systems is better?

- A program is converted into 4 billion MIPS instructions by a compiler; the MIPS processor is implemented such that each instruction completes in an average of 1.5 cycles and the clock speed is 1 GHz

- The same program is converted into 2 billion x86 instructions; the x86 processor is implemented such that each instruction completes in an average of 6 cycles and the clock speed is 1.5 GHz
Benchmark Suites

• Each vendor announces a SPEC rating for their system
  ▪ a measure of execution time for a fixed collection of programs
  ▪ is a function of a specific CPU, memory system, IO system, operating system, compiler
  ▪ enables easy comparison of different systems

The key is coming up with a collection of relevant programs
SPEC CPU

- SPEC: System Performance Evaluation Corporation, an industry consortium that creates a collection of relevant programs

- The 2006 version includes 12 integer and 17 floating-point applications

- The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400

- Note that this rating incorporates the behavior of all 29 programs – this may not necessarily predict performance for your favorite program!
Deriving a Single Performance Number

How is the performance of 29 different apps compressed into a single performance number?

• SPEC uses geometric mean (GM) – the execution time of each program is multiplied and the N\text{th} root is derived

• Another popular metric is arithmetic mean (AM) – the average of each program’s execution time

• Weighted arithmetic mean – the execution times of some programs are weighted to balance priorities
Amdahl’s Law

• Architecture design is very bottleneck-driven – make the common case fast, do not waste resources on a component that has little impact on overall performance/power

• Amdahl’s Law: performance improvements through an enhancement is limited by the fraction of time the enhancement comes into play

• Example: a web server spends 40% of time in the CPU and 60% of time doing I/O – a new processor that is ten times faster results in a 36% reduction in execution time (speedup of 1.56) – Amdahl’s Law states that maximum execution time reduction is 40% (max speedup of 1.66)
Common Principles

• Amdahl’s Law

• Energy: systems leak energy even when idle

• Energy: performance improvements typically also result in energy improvements

• 90-10 rule: 10% of the program accounts for 90% of execution time

• Principle of locality: the same data/code will be used again (temporal locality), nearby data/code will be touched next (spatial locality)
Example Problem

• A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?
Example Problem

• A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?

Normal mode energy = 100 W x 100 s = 10,000 J
Turbo mode energy = (70 x 1.2 + 30) x 100/1.2 = 9,500 J

Note:
Frequency only impacts dynamic power, not leakage power. We assume that the program’s CPI is unchanged when frequency is changed, i.e., exec time varies linearly with cycle time.
Recap

• Knowledge of hardware improves software quality: compilers, OS, threaded programs, memory management

• Important trends: growing transistors, move to multi-core and accelerators, slowing rate of performance improvement, power/thermal constraints, long memory/disk latencies

• Reasoning about performance: clock speeds, CPI, benchmark suites, performance equations

• Next: assembly instructions
Title

• Bullet