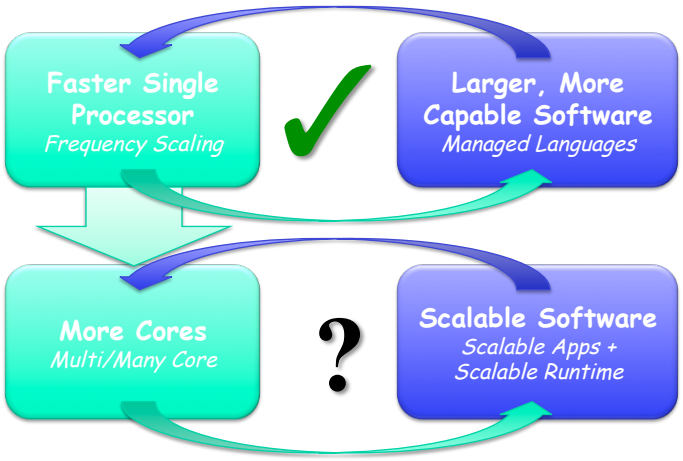
Lecture 3: Evaluating Computer Architectures

- Announcements
  - Reminder: Homework 1 due Thursday 2/2
- Last Time – technology back ground
  - Computer elements
  - Circuits and timing
  - Virtuous cycle of the past and future?
- Today
  - What is computer performance?
  - What programs do I care about?
  - Performance equations
  - Amdahl's Law

Software & Hardware: The Virtuous Cycle?



Performance Hype

"AMD Performance Preview: Taking Phenom II to 4.2 GHz"

"Intel Core i7…8 processing threads… They are the best desktop processor family on the planet."

"With 8 cores, each supporting 4 threads, the UltraSPARC T1 processor executes 32 simultaneous threads within a design consuming only 72 watts of power."

"…improves throughput by up to 41x"

"speed up by 10-25% in many cases…"

"…about 2x in two cases…"

"…more than 10x in two small benchmarks"

"speedups of 1.2x to 6.4x on a variety of benchmarks"

"can reduce garbage collection time by 50% to 75%"

"…demonstrating high efficiency and scalability"

"our prototype has usable performance"

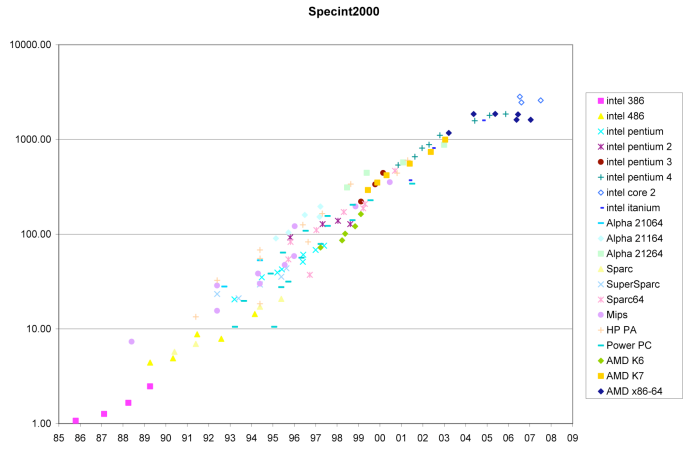
"speedups…. are very significant (up to 54-fold)"

"sometimes more than twice as fast"

"our …. is better or almost as good as …. across the board"

What Does this Graph Mean?

Performance Trends on SPEC Int 2000



Computer Performance Evaluation

- Metric = something we measure
- Goal: Evaluate how good/bad a design is
- Examples
  - Clock rate of computer
  - Power consumed by a program
  - Execution time for a program
  - Number of programs executed per second
  - Cycles per program instruction
- How should we compare two computer systems?

Tradeoff: latency vs. throughput

- Pizza delivery
  - Do you want your pizza hot?
  - Or do you want your pizza to be inexpensive?
  - Two different delivery strategies for pizza company!

This course focuses primarily on latency (hot pizza)

Latency = execution time for a single task
Throughput = number of tasks per unit time

Two notions of "performance"

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

Which has plane higher performance?

° **Time to do the task (Execution Time)**
- execution time, response time, latency

° **Tasks per day, hour, week, sec, ns. .. (Performance)**
- throughput, bandwidth

Definitions

- Performance is in units of things-per-second
  - bigger is better
- Response time of a system Y running program Z

$$\text{performance (Y)} = \frac{1}{\text{execution time (Z on Y)}}$$

- Throughput of system Y running many programs

$$\text{performance (Y)} = \frac{\text{number of programs}}{\text{unit time}}$$

- " System X is n times faster than Y" means

$$n = \frac{\text{performance(X)}}{\text{performance(Y)}}$$

Definitions

- Performance is in units of things-per-second
  - bigger is better
- Response time of a system Y running program Z

$$\text{performance (Y)} = \frac{1}{\text{execution time (Z on Y)}}$$

- Throughput of system Y running many programs

$$\text{performance (Y)} = \frac{\text{number of programs}}{\text{unit time}}$$

- " System X is n times faster than Y" means

$$n = \frac{\text{performance(X)}}{\text{performance(Y)}}$$

Which Programs Should I Measure?

Which Programs Should I Measure?

Pros

Cons

Actual Target Workload

Full Application Benchmarks

Small "Kernel" Benchmarks

Microbenchmarks

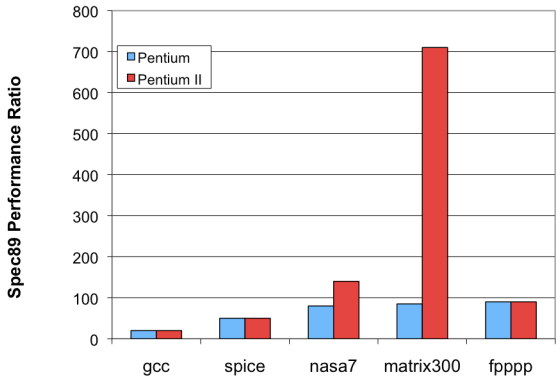
Which Programs Should I Measure?

Pros		Cons
• representative	Actual Target Workload	• very specific • non-portable • difficult to run, or measure • hard to identify cause
• portable • widely used • improvements useful in reality	Full Application Benchmarks	•less representative
• easy to run, early in design cycle	Small "Kernel" Benchmarks	• easy to "fool"
• identify peak capability and potential bottlenecks	Microbenchmarks	• "peak" may be a long way from application performance

Brief History of Benchmarking

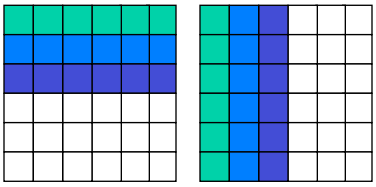
- Early days (1960s)
  - Single instruction execution time
  - Average instruction time [Gibson 1970]
  - Pure MIPS (1/AIT)
- Simple programs(early 70s)
  - Synthetic benchmarks (Whetstone, etc.)
  - Kernels (Livermore Loops)
- Relative Performance (late 70s)
  - VAX 11/780 ≡ 1-MIPS
    - but was it?
  - MFLOPs
- "Real" Applications (1989-now)
  - SPEC CPU C/Fortran
    - Scientific, Irregular
    - 89, 92, 95, 00, 07, ??
  - TPC C: Transaction Processing
  - SPECWeb
  - WinBench: Desktop
  - Graphics C/C++
    - Quake III, Doom 3
    - MediaBench
  - Java: SPECJVM98
- **Problem**: Programming Language
  - Parallel?, Java, C#, JavaScript??
  - DaCapo Java Benchmarks 06, 09
  - Parsec: Parallel C/C++, 2008

How to Compromise a Comparison: C programs running on two architectures



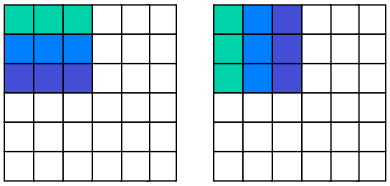
The compiler reorganized the code!

- Change the memory system performance
  - Matrix multiply cache blocking



Before

After



There are lies, damn lies, and statistics

Desraeli

benchmarks

There are lies, damn lies, and statistics

Desraeli

Benchmarking Java Programs

- Let's consider the performance of the DaCapo Java Benchmarks
- What do we need to think about when comparing two computers running Java programs?

- http://dacapo.anu.edu.au/regression/perf/2006-10-MR2.html

Pay Attention to Benchmarks & System

- Benchmarks measure the whole system
  - application
  - compiler, VM, memory management
  - operating system
  - architecture
  - implementation
- Popular benchmarks often reflect yesterday's programs
  - what about the programs people are running today?
  - need to design for tomorrow's problems
- Benchmark timings are sensitive
  - alignment in cache
  - location of data on disk
  - values of data
- Danger of *inbreeding* or positive feedback
  - if you make an operation fast (slow) it will be used more (less) often
    - therefore you make it faster (slower)
      - and so on, and so on…
  - the optimized NOP

Performance Summary so Far

- Key concepts
  - Throughput and Latency
- Best benchmarks are real programs
  - DaCapo, Spec, TPC, Doom3
- Pitfalls
  - Whole system measurement
  - Workload may not match user's
  - Compiler, VM, memory management
- Next
  - Amdahl's Law

Improving Performance: Fundamentals

- Suppose we have a machine with two instructions
  - Instruction A executes in 100 cycles
  - Instruction B executes in 2 cycles
- We want better performance….
  - Which instruction do we improve?

Speedup

- Make a change to an architecture
- Measure how much faster/slower it is

$$\text{Speedup} = \frac{\text{ExTime before change}}{\text{ExTime after change}}$$

Speedup when we know details about the change

- Performance improvements depend on:
  - how good is enhancement (factor **S**)
  - how often is it used (fraction **p**)
- Speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{ExTime w/out E}}{\text{ExTime w/ E}} = \frac{\text{Perf w/ E}}{\text{Perf w/out E}}$$

$$ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$$

$$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - p) + \frac{p}{S}}$$

Amdahl's Law: Example

- FP instructions improved by 2x
- But….only 10% of instructions are FP

$$ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$$

$$ExTime_{new} = ExTime_{old} * \left(0.9 + \frac{0.1}{2} \right) = 0.95 * ExTime_{old}$$

$$Speedup_{total} = \frac{1}{0.95} = 1.053$$

- Amdahl's Law: Speedup bounded by

$$\frac{1}{\text{fraction of time not enhanced}}$$

How Does Amdahl's Law Apply to Multicore?

- Given N cores what is our ideal speedup?

How Does Amdahl's Law Apply to Multicore?

- Given N cores what is our ideal speedup?

$$ExTime_{new} = ExTime_{old} / N$$

- Say 90% of the code is parallel and N = 16?

How Does Amdahl's Law Apply to Multicore?

- Given N cores what is our ideal speedup?

$$ExTime_{new} = ExTime_{old} / N$$

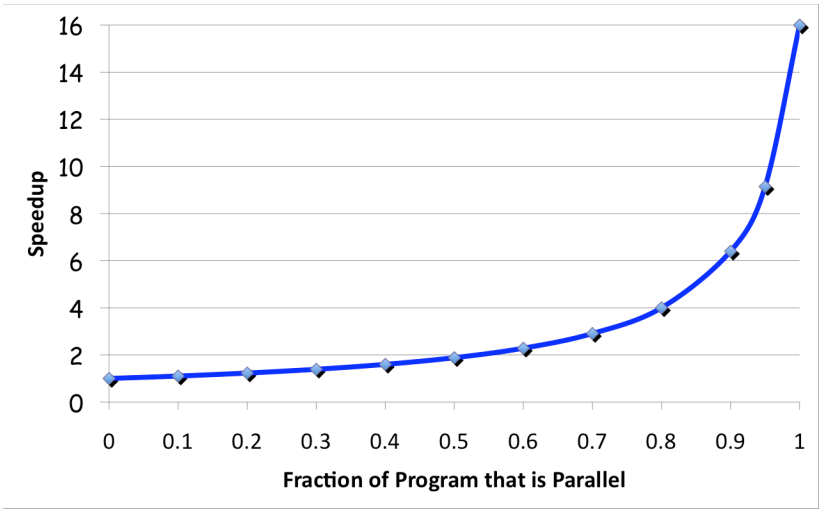
- Say 90% of the code is parallel and N = 16?

$$ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{N} \right]$$

$$ExTime_{new} = ExTime_{old} * \left(0.1 + \frac{0.9}{16} \right) = 0.15625 * ExTime_{old}$$

$$Speedup_{total} = \frac{1}{0.15625} = 6.2$$

How Does Amdahl's Law Apply to Multicore?



Performance Summary so Far

- Amdahl's law: Pay attention to what are you speeding up.
- Next Time
  - More on Performance
    - Cycles per Instruction
    - Means
  - Start: Instruction Set Architectures (ISA)
  - Read: P&H 2.1 – 2.5
  - Turn in your homework at the beginning of class