

Lecture 3: Evaluating Computer Architectures

- Announcements
 - 10 automatic drops – missing preq's or repeating course
 - HW #1 is due on Feb 3 (one week from today)
- Last Time
 - Computer elements
 - Circuits and timing
- Today
 - Performance analysis
 - Amdahl's Law
 - Performance equation
 - Computer benchmarks

Evaluation Tools

- **Benchmarks, traces, & mixes**

- macrobenchmarks & suites
 - application execution time
- microbenchmarks
 - measure one aspect of performance
- traces
 - replay recorded accesses
 - cache, branch, register

MOVE	39%
BR	20%
LOAD	20%
STORE	10%
ALU	11%

LD 5EA3
ST 31FF
....
LD 1EA2

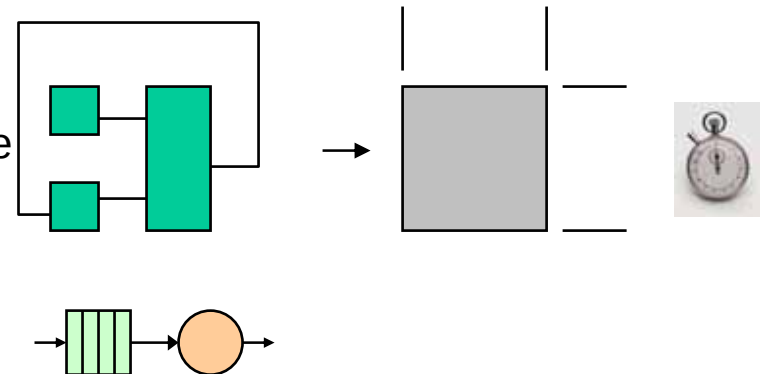
- **Simulation at many levels**

- ISA, cycle accurate, RTL, gate, circuit
 - trade fidelity for simulation rate

- **Area and delay estimation**

- **Analysis**

- e.g., queuing theory



Evaluation metrics

- Metric = something we measure
- Goal: Evaluate how good/bad a design is
- Examples
 - Execution time for a program
 - Cycles per instruction
 - Clock rate

Different metrics for different purposes

- Chose a metric that's appropriate for design level
- Examples
 - Applications perspective
 - Time to run task ([Response Time](#))
 - Tasks run per second ([Throughput](#))
 - Systems perspective
 - Millions of instructions per second (MIPS)
 - Millions of FP operations per second (MFLOPS)
 - Bus/network bandwidth: megabytes per second
 - Function Units: cycles per instruction (CPI)
 - Fundamental elements (transistors, wires, pins): clock rate

Each metric has strengths and weaknesses

Pros

- representative
- portable
- widely used
- improvements useful in reality
- easy to run, early in design cycle
- identify peak capability and potential bottlenecks

Actual Target Workload

Full Application Benchmarks

Small “Kernel” Benchmarks

Microbenchmarks

Cons

- very specific
- non-portable
- difficult to run, or measure
- hard to identify cause
- less representative
- easy to “fool”
- “peak” may be a long way from application performance

Some Warnings about Benchmarks

- Benchmarks measure the whole system
 - application
 - compiler
 - operating system
 - architecture
 - implementation
- Popular benchmarks typically 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

Know what you are measuring!

- Compare apples to apples
- Example
 - Wall clock execution time:
 - User CPU time
 - System CPU time
 - Idle time (multitasking, I/O)

```
csh> time latex lecture2.tex
csh> 0.68u 0.05s 0:01.60 45.6%
```

↓ ↓ ↓ ↓

user system elapsed % CPU time

Two notions of “performance”

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

Which has higher performance?

- ° Time to do the task (Execution Time)
 - execution time, response time, latency
- ° Tasks per day, hour, week, sec, ns. .. (Performance)
 - throughput, bandwidth

Tradeoff: latency vs. throughput

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

In this course:

We will focus primarily on latency
(execution time for a single task)

Definitions

- Performance is in units of things-per-second
 - bigger is better
- If we are primarily concerned with response time
 - $\text{performance}(x) = \frac{1}{\text{execution_time}(x)}$

"X is n times faster than Y" means

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

Brief History of Benchmarking

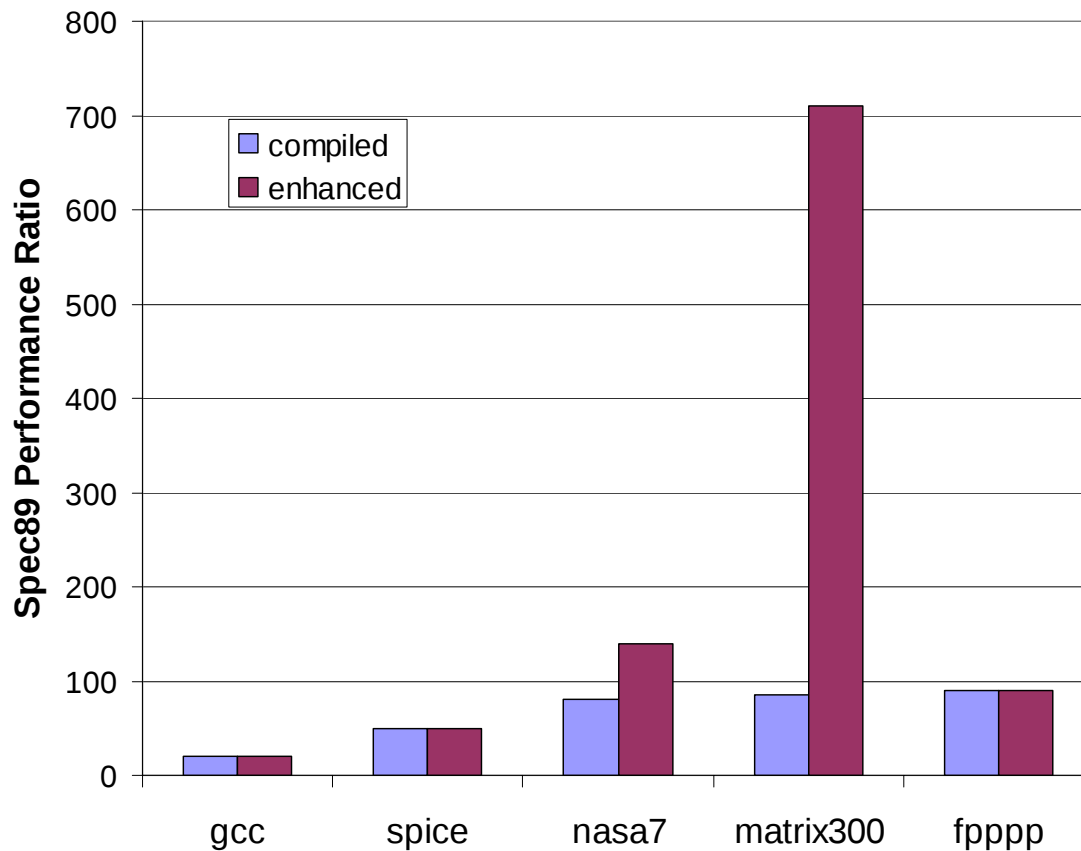
- Early days (1960s)
 - Single instruction execution time
 - Average instruction time [Gibson 1970]
 - Pure MIPS (1/AIT)
- Simple programs (early 70s)
 - Synthetic benchmarks (Whetstone, etc.)
 - Kernels (Livermore Loops)
- Relative Performance (late 70s)
 - VAX 11/780 $\equiv$ 1-MIPS
 - but was it?
 - MFLOPs
- “Real” Applications (late 80s-now)
 - SPEC
 - Scientific
 - Irregular
 - TPC
 - Transaction Processing
 - Winbench
 - Desktop
 - Graphics
 - Quake III, Doom 3
 - MediaBench

SPEC: Standard Performance Evaluation Corporation

(www.spec.org)

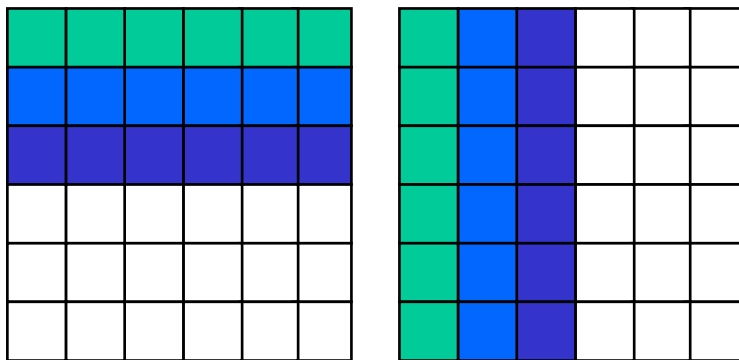
- System Performance and Evaluation Cooperative
 - HP, DEC, Mips, Sun
 - Portable O/S and high level languages
- Spec89 $\Rightarrow$ Spec92 $\Rightarrow$ Spec95 $\Rightarrow$ Spec2000 $\Rightarrow$
- Categories
 - CPU (most popular)
 - JVM
 - SpecWeb - web server performance
 - SFS - file server performance
- Benchmarks change with the times and technology
 - Elimination of Matrix 300
 - Compiler restrictions

How to Compromise a Benchmark



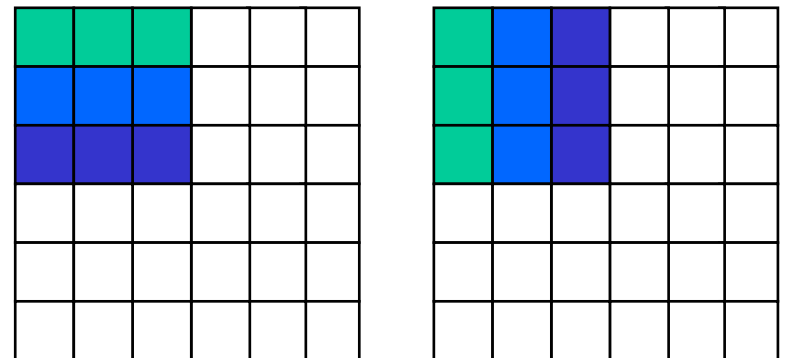
The compiler reorganized the code!

- Change the memory system performance
 - Matrix multiply cache blocking



Before

After



Spec2000 Suite

- 12 Integer benchmarks (C/C++)
 - compression
 - C compiler
 - Perl interpreter
 - Database
 - Chess
- 14 FP applications (Fortran/C)
 - Shallow water model
 - 3D graphics
 - Quantum chromodynamics
 - Computer vision
- Characteristics
 - Computationally intensive
 - Little I/O
 - Small code size
 - Variable data set sizes

SPEC Leaders (4/00)

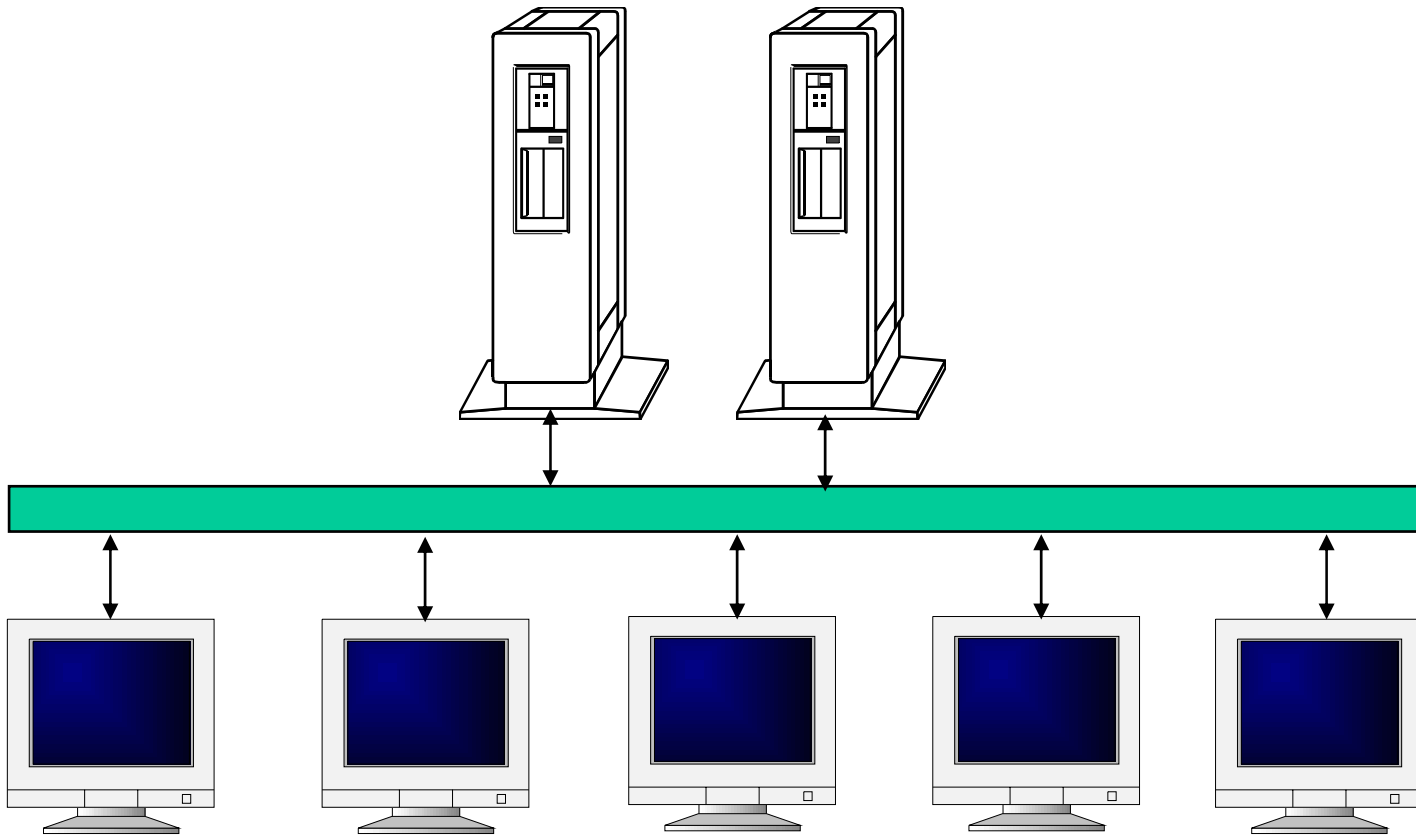
	Intel Pentium III	AMD Athlon	Compaq Alpha 21264	Sun Ultra-2	IBM Power3	HP PA-8600
Clock rate	1000MHz	1000 MHz	700MHz	450MHz	400MHz	552MHz
Issue rate	3 x86	3 x86	4	4	4	4
Cache (I/D)	16/16/256K	64K/64K	64K/64K	16K/16K	32K/64K	512K/1M
# transistors	24 million	22 million	15.2 million	3.8 million	23 million	130 million
Technology	0.18 μ m	0.18 μ m	0.25 μ m	0.29 μ m	0.22 μ m	0.25 μ m
Die Size	106mm ²	102mm ²	205mm ²	126mm ²	163mm ²	477mm ²
Estimated mfg. Cost	\$40	\$70	\$160	\$70	\$110	\$330
SPECint95	46.6	42.0	34.7	16.2	23.5	38.4
SPECfp95	31.9	29.4	54.5	24.6	46.0	61.0

12/2003: AMD Opteron 148, 2.0 GHz:

SPECint2000base 16.3

SPECfp2000base 17.5

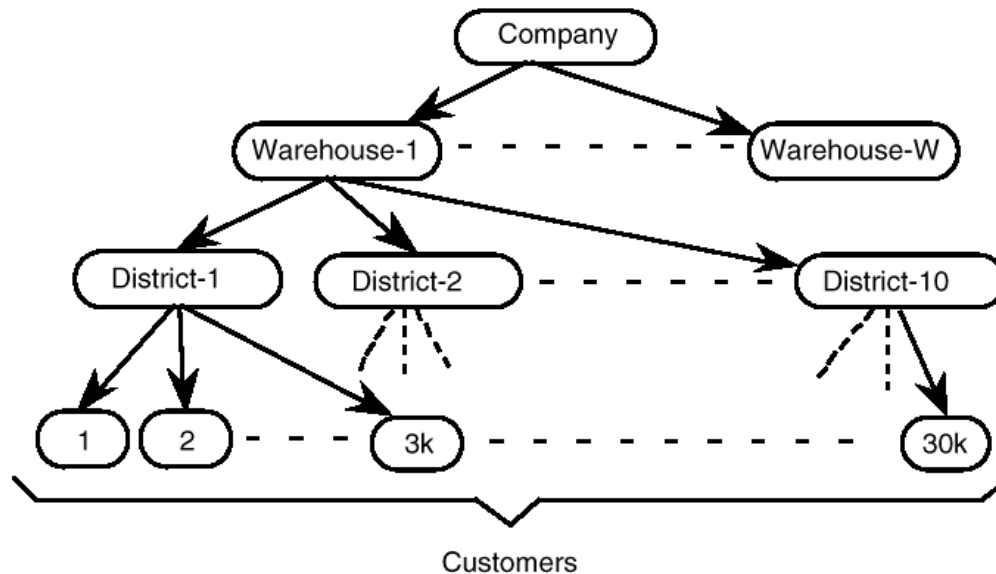
Transaction-based Systems



TPC - Transaction Processing Council

- Established in 1988
- Measure whole system performance and cost
 - Maximum TPS (transactions per second)
 - \$/TPS
- Test specifies high level functional requirements
 - Independent of platform
- Workload scales
- Transaction rate constrained by response time requirement
 - 90% of transactions complete in < 2 seconds

TPC-C: OLTP



- W warehouses
- 10 districts/warehouse
- 3,000 customers/district
- 100,000 items

- 10 items per order
- 1% not in stock at regional warehouse
- Frequent reads and writes

TPC-C Results (8/00)

- IBM Netfinity 8500R
 - Platform
 - 32 servers, 4 CPUs each
 - 700MHz PentiumIII - Xeon
 - 128 GB memory
 - 4TB disk
 - Windows 2000 server
 - IBM DB2 database server
 - 368,640 users
 - Results
 - Cost: \$14.2M
 - Throughput: 440K tpm
 - Price/perf: \$32/tpm
- Compaq ProLiant ML570
 - Platform
 - 2 servers, 3 CPUs
 - 700MHz PentiumIII - Xeon
 - 2.5 GB memory
 - 1.5TB disk
 - Windows 2000 server
 - Microsoft SQL
 - 16,200 users
 - Results
 - Cost: \$200K
 - Throughput: 20K tpm
 - Price/perf: \$10/tpm

Desktop/Graphics Benchmarks

- WinStone
 - Corel WordPerfect suite, Lotus Smartsuite, Microsoft Office
 - High end: Photoshop
- CPU Mark99
 - Synthetic benchmark - tests CPU, caches, external memory
- 3DMark2003
 - Synthetic benchmark for 3-D rendering

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?

Amdahl's Law

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

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

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

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

Amdahl's Law: Example

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

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

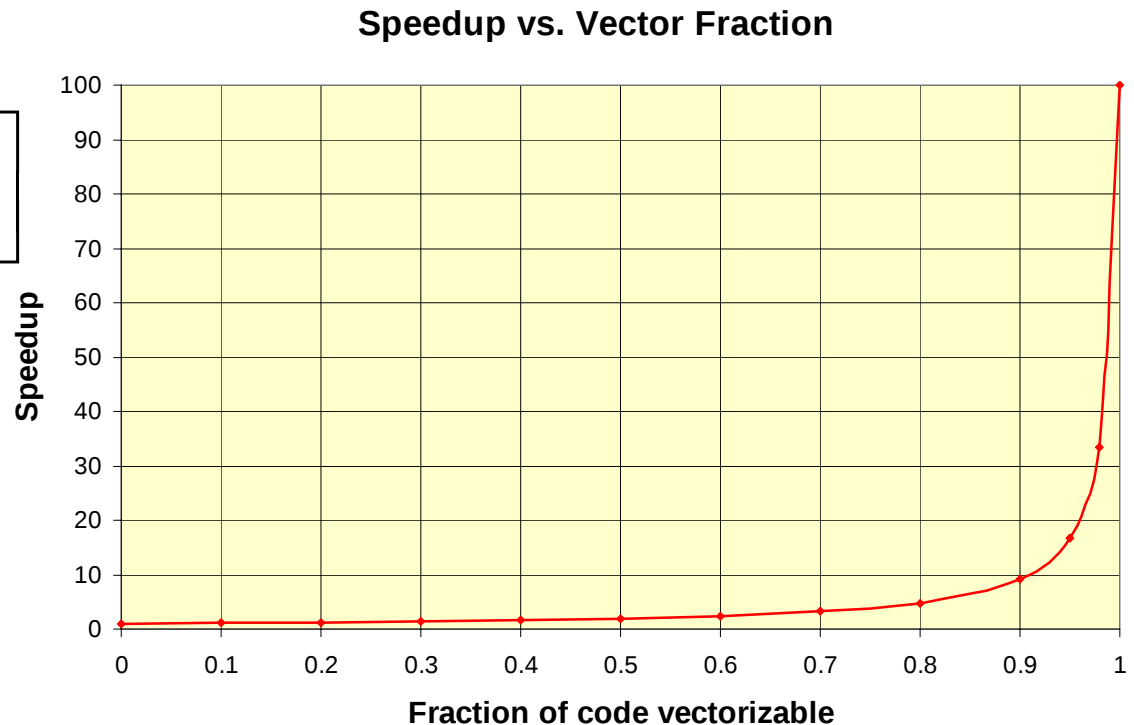
$$Speedup_{total} = \frac{1}{0.95} = 1.053$$

- Speedup bounded by $\frac{1}{\text{fraction of time not enhanced}}$

Amdahl's Law: Example 2

- Speed up vectorizable code by 100x

$$T_1 = T_0 \left[(1 - p) + \frac{p}{S} \right]$$



Amdahl's Law: Summary message

- Make the common case fast
- Examples:
 - All instructions require instruction fetch, only fraction require data
⇒ optimize instruction access first
 - Data locality (spatial, temporal), small memories faster
⇒ storage hierarchy: most frequent accesses to small, local memory

CPU Performance Equation

- 3 components to execution time:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Cycles}}{\text{Instruction}} * \frac{\text{Seconds}}{\text{Cycle}}$$

- Factors affecting CPU execution time:

	Inst. Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set	X	X	(X)
Organization		X	X
MicroArch		X	X
Technology			X

- Consider all three elements when optimizing
- Workloads change!

Cycles Per Instruction (CPI)

- Depends on the instruction

$$CPI_i = \text{Execution time of instruction } i * \text{Clock Rate}$$

- Average cycles per instruction

$$CPI = \sum_{i=1}^n CPI_i * F_i \quad \text{where } F_i = \frac{IC_i}{IC_{tot}}$$

- Example:

Op	Freq	Cycles	CPI(i)	%time
ALU	50%	1	0.5	33%
Load	20%	2	0.4	27%
Store	10%	2	0.2	13%
Branch	20%	2	0.4	27%
		CPI(total)	1.5	

Comparing and Summarizing Performance

- Fair way to summarize performance?
- Capture in a single number?
- Example: Which of the following machines is best?

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Total Time	1001	110	40

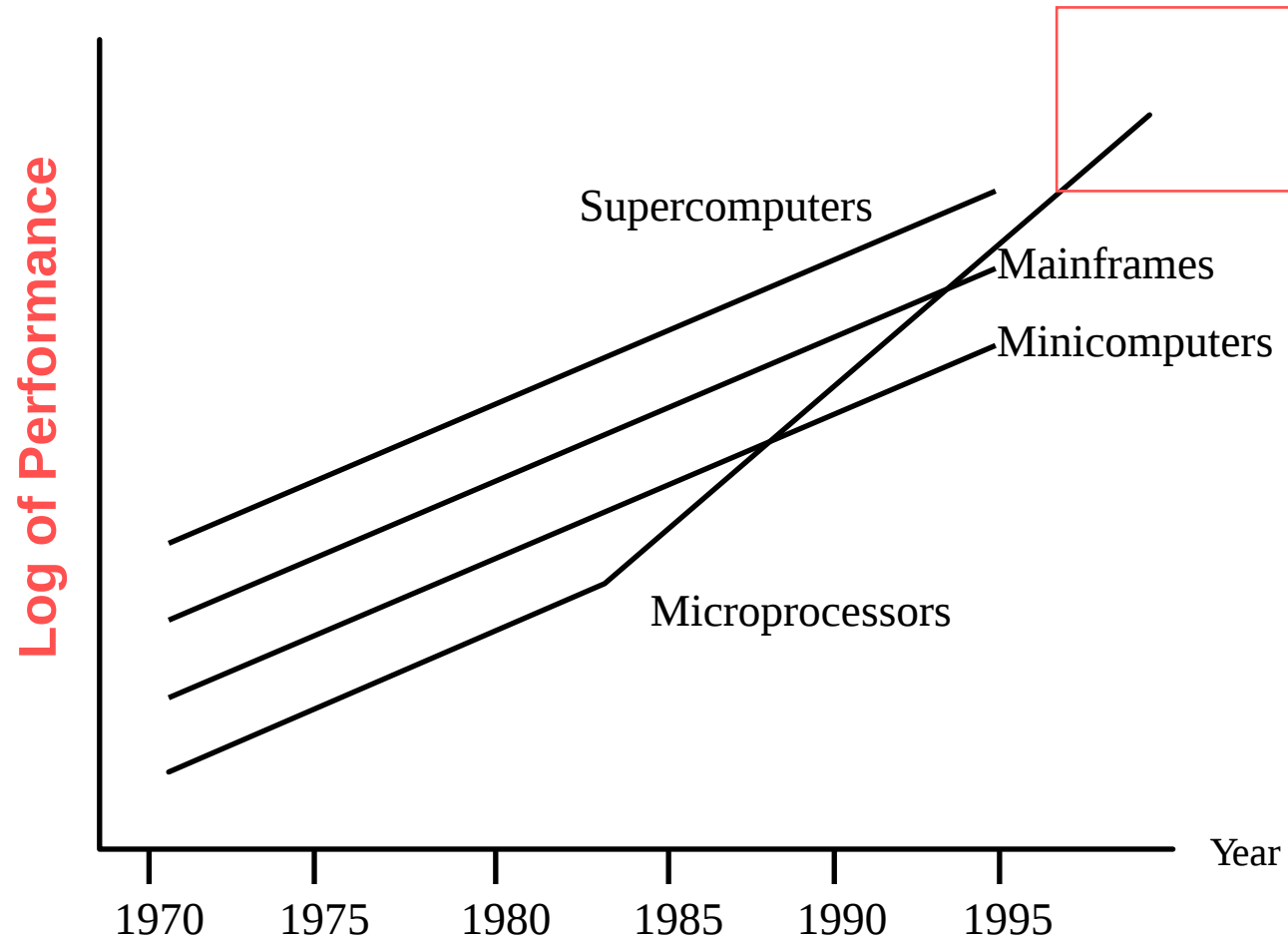
Means

Arithmetic mean $\frac{1}{n} \sum_{i=1}^n T_i$ Can be weighted: $a_i T_i$
Represents total execution time

Harmonic mean $\frac{n}{\sum_{i=1}^n \frac{1}{R_i}}$ $R_i = 1/T_i$

Geometric mean $\left(\prod_{i=1}^n \frac{T_i}{T_{ri}} \right)^{\frac{1}{n}}$ Good for mean of ratios,
where the ratio is with respect to a reference.

Platform Performance Trends



Is Speed the Last Word in Performance?

- Depends on the application!
- Cost
 - Not just processor, but other components (ie. memory)
- Power consumption
 - Trade power for performance in many applications
- Capacity
 - Many database applications are I/O bound and disk bandwidth is the precious commodity

Summary

- Best benchmarks are real programs
 - Spec, TPC, Winstone
- Pitfalls still exist
 - Whole system measurement
 - Workload may not match user's
- Next time
 - Instruction set architectures (ISA)