

## Performance II

---

- ◆ **Last Time**

- Computer Architecture: definition and drivers
- Basic notions of Performance and Relative Performance

- ◆ **Today**

- Time bases and Performance Metrics
- Amdahl's Law
- Comparing Performance

- ◆ **Reminders/Announcements**

- Homework #1 posted to the web on 1/14/00, **Due 1/24/00 at the beginning of discussion section (no late homeworks)**
- TA office hours: W9-11am, F1-3pm, both in 3337D APM
- Quiz on Thursday, as usual

## Performance Metrics

---

- ◆ **Execution Time**

- ◆ **Many are units of work / units of time**

- ◆ **Work: instructions, floating point operations, polygons, \*answers\*, etc.**

- ◆ **Time: seconds, other things...**

## What is Time?

---

$$\begin{aligned}\text{CPU Execution Time} &= \text{CPU clock cycles} * \text{Clock cycle time} \\ &= \text{CPU clock cycles} / \text{Clock rate}\end{aligned}$$

- ◆ Every conventional processor has a clock with an associated clock cycle time or clock rate.
- ◆ Every program runs in an integral number of clock cycles.

x MHz = x millions of cycles/second (clock rate)

1/ (x MHz) = cycle time, 1/(500 MHz) = 2 ns

CS 141 Chien

3

Jan 18, 2000

## How many clock cycles?

---

$$\begin{aligned}\text{Number of CPU cycles} &= \text{Instructions executed} * \\ &\quad \text{Average Clock Cycles per Instruction (CPI)}\end{aligned}$$

*or*

$$\text{CPI} = \text{CPU clock cycles} / \text{Instruction count}$$

=> Nice technology scaled “architecture metric”

CS 141 Chien

4

Jan 18, 2000

## All Together Now

---

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

*seconds* (pointing to CPU Execution Time)  
*instructions* (pointing to Instruction Count)  
*cycles/instruction* (pointing to CPI)  
*seconds/cycle* (pointing to Clock Cycle Time)

CS 141 Chien

5

Jan 18, 2000

## Who Affects Performance?

---

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- ◆ programmer
- ◆ compiler
- ◆ instruction-set architect
- ◆ machine architect
- ◆ hardware designer
- ◆ materials scientist/physicist/silicon engineer

CS 141 Chien

6

Jan 18, 2000

## Performance Variation

---

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	different	similar	same
same programs, different machines, same ISA	same	different	different
Same programs, different machines	somewhat different	different	different

CS 141 Chien

7

Jan 18, 2000

## Speedup

---

- ◆ **Speedup is just relative performance on the same machine with something changed.**

$$\text{speedup} = \text{relative performance} = \frac{\text{ET for entire task without change}}{\text{ET for entire task with change}}$$

CS 141 Chien

8

Jan 18, 2000

## Amdahl's Law

---

- ◆ The impact of a performance improvement is limited by the percent of execution time affected by the improvement

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

- ◆ Make the common case fast!!

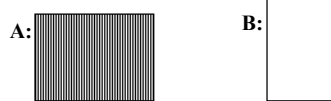
CS 141 Chien

9

Jan 18, 2000

## Comparing Computers using Metrics

---



- ◆ Run programs, record execution times

	<u>Runtime (secs)</u>
System A	50
System B	75

- ◆ How can we describe the relative performance of machines with such a metric?

CS 141 Chien

10

Jan 18, 2000

## Relative Performance

---

◆ Can be confusing

- A runs in 12 seconds
- B runs in 20 seconds
- $A/B = .6$ , so A is 40% faster, or 1.4X faster, or B is .40% slower
- $B/A = 1.67$ , so A is 67% faster, or 1.67X faster, or B is 67% slower

◆ Needs a precise definition

CS 141 Chien

11

Jan 18, 2000

## Relative Performance Statements

---

	<u>Runtime (secs)</u>
System A	50
System B	75

◆ Performance Ratio (A/B) =  $\frac{\text{ExecTime}_B}{\text{ExecTime}_A} = \frac{75}{50} = 1.5$

- "A is 1.5 times faster than B"

◆ Performance Ratio (A/B) =  $\frac{\text{Perf}_A}{\text{Perf}_B} = \frac{1/\text{ExecTime}_A}{1/\text{ExecTime}_B} = \frac{75}{50} = 1.5$

◆ Performance Ratio (B/A) =  $\frac{\text{ExecTime}_A}{\text{ExecTime}_B} = \frac{50}{75} = 0.67$

- "B is 0.67 times the performance of A"

CS 141 Chien

12

Jan 18, 2000

## Performance

---

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X}, \text{ for program } X$$

- ◆ only has meaning in the context of a program or workload
- ◆ Not very intuitive as an absolute measure

## Defining Relative Performance

---

$$\text{Relative Performance} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

- ◆ We can remove all ambiguity by always constraining  $n$  to be  $> 1 \Rightarrow$  machine  $x$  is  $n$  times faster than  $y$ .

## Performance Measurements

---

◆ => Of metrics, not computer architectures!

◆ Other Metrics:

Millions of Insts/Sec = (MIPS)	$\frac{\text{\# of Instructions}}{\text{ExecTime} \times 10^6}$	(for a particular program)  (particular program)
Cycles per Instruction = (CPI)	$\frac{\text{ExecTime} \times \text{Clock Rate}}{\text{\# of Instructions}}$	“Complexity of Instructions”
Clock Rate (Megahertz) =	Hardware Implementation Characteristic	“Complexity of a cycle”

CS 141 Chien

15

Jan 18, 2000

## Performance Summary

---

◆ Many metrics, basis for comparison

- Relative comparison
- Quantitative comparison

◆ Execution time is the preferred metric.

- Cannot hide anything, include things by default
- Easiest to avoid errors in comparison
- Corresponds to user waiting time, resource usage

◆ What metrics?

CS 141 Chien

16

Jan 18, 2000

## Performance Metrics

---

- ◆ **Comparing performance of computer systems based on**

- execution time
- relative performance
- MIPS
- MFLOPS
- etc.

- ◆ **Problem: everyone's use of computers differs**

CS 141 Chien

17

Jan 18, 2000

## More Meaningful Performance Characterization

---

- ◆ **Most computers are used to run a variety of programs.**

- ◆ **=> set of application programs = “workload mix”**

- ◆ **Each user should run their exact mix on the desired systems of interest.**

- ◆ **Is this easy?**

- Applications are dynamic, large, and complex
- Configurations may be large, dynamic, and complex
- Applications and the systems may not be available

CS 141 Chien

18

Jan 18, 2000

## Combining Performance

---

- ◆ If workload consists of 10 programs, how to combine the performance metric results?
- ◆ Incomparable results in a 10-dimensional space?
- ◆ Want a one-dimensional metric space of better-worse

CS 141 Chien

19

Jan 18, 2000

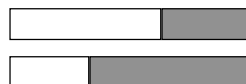
## Summarizing Performance

---

- ◆ Arithmetic mean is a consistent summary:

$$\text{Arithmetic Mean} = \text{AM} = (1/N) \sum \text{Time}_i$$

- average execution time
- uses equal measures of each application
- contribution is based on the data sets actually run!!



=> same performance

=> Different mix, very different performance

CS 141 Chien

20

Jan 18, 2000

## Practical Performance Evaluation

---

- ◆ Is this how people really evaluate machines?
  - Some, but not most.
- ◆ Why not?
  - Time, effort, expense, approximations of the workload and system
- ◆ Practically, several well established “benchmark sets”.
  - Whetstone and Dhrystones
  - Livermore Loops (Kernels)
  - Systems Performance Evaluation Cooperative (SPEC)
  - PERFECT Club, SPLASH
  - Transaction Processing Council (TPC)
  - Norton Index, WinMarks, Xstones
- ◆ Read and buy, areas, specialization, complications

CS 141 Chien

21

Jan 18, 2000

## Summarizing Performance

---

<u>Benchmark</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
Machine A	3	8	12	8	5
Machine B	4	5	14	4	6

- ◆ Arithmetic Mean
  - $AM(A) = 36/5 = 7.2$  secs
  - $AM(B) = 33/5 = 6.6$  secs
- ◆ Machine A is factor on 3 of 5 Applications, however... arithmetic mean says that is inferior...
- ◆ What does this mean?

CS 141 Chien

22

Jan 18, 2000

## Arithmetic Mean

---

- ◆ Includes not only how often it was faster, but also how much faster.
- ◆ Results may be fragile:
  - Eliminating any of the 2 programs for which  $B > A$ , then the summary outcome would change
- ◆ What if  $A = 100x B$  on one benchmark, but  $0.8x$  on the rest?

## Weighted Arithmetic Mean

---

- ◆ Benchmarks and Data sets not always available in convenient/appropriate sizes
- ◆ Arithmetic summaries are sensitive to the workload mix
- ◆  $\Rightarrow$  weight the contribution by expected usage
- ◆  $WAM = (1/N) \sum w_i * T_i$
- ◆ where  $\sum w_i = 1$

## WAM Example

---

<u>Benchmark</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
Machine A	3	8	12	8	5
Machine B	4	5	14	4	6
Weights	1/5	1/5	1/10	3/10	1/5

- ◆ Calculations can be done with smaller runs times in many cases.
  - $WAM(A) = 3/5 + 8/5 + 12/10 + 24/10 + 5/5 = 34/5 = 6.8$  secs
  - $WAM(B) = 4/5 + 5/5 + 14/10 + 12/10 + 6/5 = 28/5 = 5.6$  secs
- ◆ Weights can affect the outcome. Easily rejiggered to match different workloads, a single set of measurements can be used in many contexts.

## Pitfalls

---

- ◆ Inaccuracies in Measuring Time
- ◆ Changes in units of work
- ◆ System Perturbations -- multitasking, sharing the I/O system, network operations, etc.
- ◆ Computing Arithmetic Means on rates! (not normalized correctly)

## Example of Incorrectly Summarizing Rates

<u>Benchmark</u>	<u>1</u>	<u>2</u>	<u>3</u>
Machine A	1	8	25
Machine B	2	5	50

- ◆ Measurements in rates, say megaflops
  - AM(A) ?= (1 + 8 + 25)/3 = 11.33
  - AM(A) ?= (2+5+50)/3 = 19.0
- ◆ Is the summary representative of the results?
- ◆ Problem: Benchmarks which achieve high rates have a disproportionate influence.
- ◆ => implicitly normalizing with each benchmark run for same amount of TIME.
  - As benchmark gets faster, preserving this assumption increases the distortion.

CS 141 Chien

27

Jan 18, 2000

## Summarizing Rates using Harmonic Mean

<u>Benchmark</u>	<u>1</u>	<u>2</u>	<u>3</u>
Machine A	1	8	25
Machine B	2	5	50

- ◆ Harmonic mean normalizes rates assuming an equal amount of work.
 
$$\frac{1}{R_a} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$
- ◆ Ra(A) = 1/(1 + 1/8 + 1/25) = 0.86
- ◆ Ra(B) = 1/(1/2 + 1/5 + 1/50) = 1.38
- ◆ Emphasizes improvement on the slowest benchmark!

CS 141 Chien

28

Jan 18, 2000

## SPEC Benchmarks

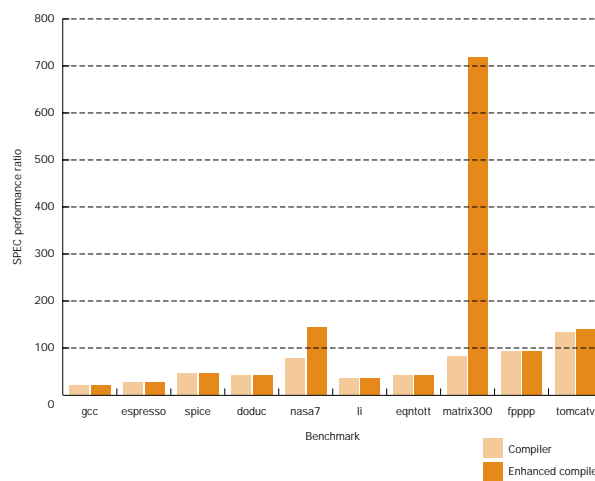
- ◆ Popular performance suite
- ◆ Contains
  - 8 “integer” programs (compilers, compression, lisp interpreter, perl, database)
  - 9 “floating point” benchmarks
  - Results are combined using the geometric mean (see textbook)
- ◆ Caveats:
  - This is a CPU benchmark (not a good test of the entire system...); no I/O and cpu-intensive programs
  - some testing of the memory system
  - compiler is a critical part of the test

CS 141 Chien

29

Jan 18, 2000

## SPEC89 Suite -- “Compiler Breakthrough”



CS 141 Chien

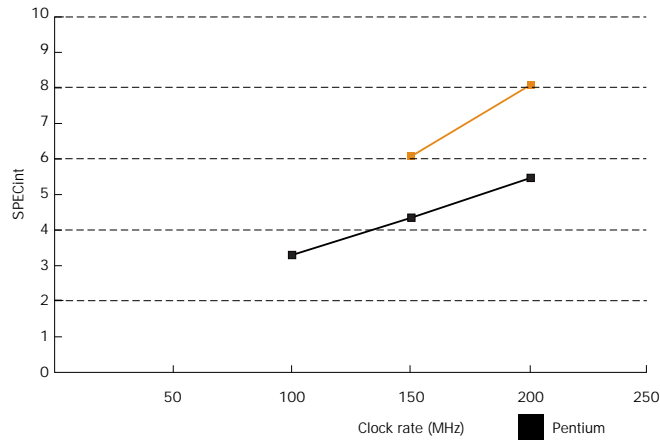
◆ measures compiler as much as architecture!

30

Jan 18, 2000

## SPEC Results -- Pentium and Pentium Pro

---



CS 141 Chien

31

Jan 18, 2000

## Performance II Summary

---

- ◆ CPI Metric
- ◆ Amdahl's Law
- ◆ Pitfalls of single metrics
- ◆ Summarizing Performance: AM, WAM
- ◆ Pitfalls of summarizing performance
- ◆ Spec benchmark suite
- ◆ Next Time:
  - Instruction Sets: the Software/Hardware Interface

CS 141 Chien

32

Jan 18, 2000