

Computer Performance

CIT 595
Spring 2007

CIT 595

8 - 1

Motivation for Examining Performance

- Hardware performance is often key to the effectiveness of an entire system of hardware and software
- Why certain piece of software performs the way it does?
- Why one instruction set can be implemented to perform better than another?

CIT 595

8 - 2

What defines Performance?

- If you are running a program on 2 different workstations, the faster one is the one that gets the job done first
 - As a user you are interested reducing the **response time** i.e. time from start to completion of task (a.k.a. execution time)
- If you are running a computer center that had two large timeshared computer running jobs submitted by many users then the faster computer is the one that completed most jobs during the day
 - As system administrator you are interested increasing **throughput** i.e. the total amount of work done in a given time

CIT 595

8 - 3

Measuring Response Time

- **Response Time** is one of the measure of computer performance
- So how do we measure the time?
- We could just time from started our program till the time it was done executing (i.e. elapsed or response time)
- However modern computers are often timeshared and may simultaneously work on several programs
 - So we want distinguish between the time the processor is working on our behalf vs. the time spent waiting
 - **CPU (processor) execution time** i.e. time spent computing for a particular program (not running other programs or waiting on I/O)
 - Response Time = CPU Execution Time + Wait Time

CIT 595

8 - 4

CPU Execution Time

- CPU execution time of program depends on:
 - Remember that a program is made of some number of instructions, hence it dependent on the total number of instructions i.e. **instruction count**
 - Each instruction performance is dependent on the number of clock cycles it takes to complete the instruction cycle (Fetch thru Store) called as **cycles per instruction (CPI)**
 - One clock cycle is time interval with clock ticks i.e. **clock cycle time**

$$\text{CPU Time} = \frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

↓ Clock Cycle time
 ↓ CPI
 ↓ Instruction Count

CIT 595

8 - 5

CPU Execution Time (contd..)

- Clock cycle time = 1/Clock Rate or 1/Frequency

$$\text{CPU Time} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

CIT 595

8 - 6

Cycles Per Instruction (CPI)

- CPI is an average of all the instructions executed in the program
- Depends on design details (micro-architecture) of the machine including memory system and the processor structure
 - We have seen a multiple cycle implementation (Hardwired/Microprogram)
 - Later we will see pipelined implementation as well as how memory system affects CPI
 - Thus CPI varies among different implementations of the same ISA
- CPI is obtained by performing a detailed simulation of an implementation

CIT 595

8 - 7

Example: Calculating CPI

- Given (assuming each instr. phase takes 1 cycle):

Instruction	# cycles	% Instr. Count
Load	5	25
Store	5	15
ALU	5	40
Branch/Jump	3	15
Others	6	5

The average CPI is

$$= 0.25 \times 5 + 0.15 \times 5 + 0.40 \times 5 + 0.15 \times 3 + 0.05 \times 5$$

$$= 4.75 \text{ cycles}$$

CIT 595

8 - 8

Number of Instructions

- Number of Instructions depends on:
 - Instruction Set Architecture (ISA)
 - Application Developer
 - Compiler
- Note: The instruction count is the number of actual instructions executed at runtime

CIT 595

8 - 9

Clock Cycle Time

- Clock Cycle Time depends on:
 - Technology and circuit optimization (i.e. minimized logic)
 - Micro architecture (esp. Pipelined Implementation)

CIT 595

8 - 10

Maximizing Performance in relation to Execution Time

- One way to maximize performance, we want to minimize execution time
- Thus we can relate performance and execution time for a machine X as:

$$\text{Performance}_x = \frac{1}{\text{Execution Time}_x}$$

CIT 595

8 - 11

Performance Ratio

- To claim that Machine X performs better than Machine Y:

$\text{Performance}_x > \text{Performance}_y$ i.e.

$$\frac{1}{\text{Execution Time}_x} > \frac{1}{\text{Execution Time}_y}$$

$$\text{Performance Ratio} = \frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution Time}_y}{\text{Execution Time}_x}$$

CIT 595

8 - 12

Relative Performance Example

- If machine A runs a program in 10 seconds and machine B runs program in 15 seconds, how much faster is machine A than B?

$$\text{Performance Ratio} = \frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution Time}_B}{\text{Execution Time}_A}$$
$$= 15/10 = 1.5$$

Thus machine A is 1.5 times faster than B or
The execution time on B is 1.5 times longer than on A

CIT 595

8 - 13

Example

- Machine A has a clock cycle of 1ns and a CPI of 2.0 for some program, and machine B has a clock cycle of 2ns and CPI of 1.2 for the same program. Which machine is faster for this program and by how much?

Let "I" be the number instructions per program

$$\text{CPU Time}_A = 1 \times 2.0 \times I = 2 \times I \text{ ns}$$

$$\text{CPU Time}_B = 2 \times 1.2 \times I = 2.4 \times I \text{ ns}$$

Clearly Machine A is faster than B by :

$$\text{Performance Ratio} = \frac{\text{Time}_B}{\text{Time}_A} = \frac{2.4 \times I \text{ ns}}{2.0 \times I \text{ ns}} = 1.2$$

CIT 595

8 - 14

Aside: User vs. System CPU Time

- True CPU Time comprises of *user* CPU time and *system* CPU Time (O.S. performing tasks on user behalf)
 - This is because on most modern day computers, no program runs without some O.S running on the hardware
- However, comparing performance between machines with different O.S will be unfair or inaccurate, so in those cases we want to ignore the system CPU time

CIT 595

8 - 15

Aside: Program Runtime

- Curious to find how long it take to run your program?
- On Unix, command called "time" provides the following:
 - First do: gcc myprog.c -o progname
 - time progname
 - 90.7u 12.9s 2:39 65% - User CPU time is 90.7 u, system CPU time is 12.9 s, elapsed time is 159 s (2mins 39 sec) and the % of elapsed time that is CPU time
 - So more than 1/3 of elapsed time was spent waiting for I/O, running other programs or both

CIT 595

8 - 16

Aside: Program Runtime

Drawbacks of UNIX *time* command

- Has poor resolution
 - only milliseconds
- Sometimes one wants a higher precision
 - esp. if **performance** improvements are in the 1-2% range
- Times the whole code
 - Sometimes one is only interested in timing some part of the code
 - Sometimes one wants to compare the **execution time** of different sections of the code

CIT 595

8 - 17

Aside: Program Analysis

- Instead of using “time” command, use sophisticated tools that not only provide you time but time spent different sections of the program
- Profiler
 - Performance analysis tool that measures the behavior of a program as it runs particularly the frequency and duration of function calls
 - The output is a statistical summary of the events observed (a **profile**)

CIT 595

8 - 18

Maximizing Performance in relation to Throughput

- Throughput is amount of work that can be done in a given time
- Measured in tasks per time unit
 - E.g. x tasks per hour
- So higher the throughput, better the performance
- Also, *Throughput* and *Response Time* are inversely related
 - If system carries out a task in k seconds then its throughput is 1/k tasks per second
- Throughput can be increased:
 - Pipelining at instruction level (chp 5)
 - Timesharing a computer system (chp 7 - I/O & 8 - OS)
 - Achieving Parallelism (chp 9)
 - Superscalar Processor
 - Use multiprocessors (Shared Memory, Distributed, Chip Multiprocessors)

CIT 595

8 - 19

Evaluating Performance

- In general we want compare performance of computer systems, not just with one program, but many others
- So we perform tests with different programs (workload) and we arrive at an expected behavior
- We evaluate data gathered using statistical analysis
- The statistical method use depends on the nature of the data as well as distribution of the test results

CIT 595

8 - 20

Arithmetic Mean

$$\text{Arithmetic Mean} = \frac{\sum_{i=1}^n x_i}{n}$$

- The arithmetic mean can be misleading if the data are skewed or scattered
 - Consider the execution times given in the table below. The performance differences are hidden by the simple average

Program	System A Execution Time	System B Execution Time	System C Execution Time
v	50	100	500
w	200	400	600
x	250	500	500
y	400	800	800
z	5000	4100	3500
Average	1180	1180	1180

CIT 595

8 - 21

Weighted Average

- Weighted average* can be revealing if we know how frequently was each program run during daily processing on these systems
- Example: The weighted average for System A is:
 - $50 \times 0.5 + 200 \times 0.3 + 250 \times 0.1 + 400 \times 0.05 + 5000 \times 0.05 = 380$

Program	Execution Frequency	System A Execution Time	System C Execution Time
v	50%	50	500
w	30%	200	600
x	10%	250	500
y	5%	400	800
z	5%	5000	3500
Weighted Average		380 seconds	695 seconds

CIT 595

8 - 22

Problem with Weighted Average

- However, what if you decide to change the frequency:
 - System A performance degrades with different execution frequency

Program	Execution Time	Execution Frequency #1	Execution Frequency #2
v	50	50%	25%
w	200	30%	5%
x	250	10%	10%
y	400	5%	5%
z	5000	5%	55%
Weighted Average		380 seconds	2817.5 seconds

CIT 595

8 - 23

Geometric Mean

- Gives a consistent number with which to perform comparisons regardless of the distribution of the data
- Represented as:

$$G = \left[x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n \right]^{\frac{1}{n}}$$

- Performance results are stated in relation to the performance of a common machine used as reference (i.e. systems under evaluation are normalized to the reference machine)

CIT 595

8 - 24

Geometric Mean Example

Program	System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
v	50	2	100	1	500	0.2
w	200	2	400	1	600	0.6667
x	250	2	500	1	500	1
y	400	2	800	1	800	1
z	5000	0.82	4100	1	3500	1.1714
Geometric Mean		1.6733		1		0.6898

Geometric Mean For A normalized with respect to System B:

$$= (100/50 \times 400/200 \times 500/250 \times 800/400 \times 4100/5000)^{1/5}$$

$$= 1.6733$$

Also the Geometric Mean ratios are consistent no matter which system we choose for the reference machine

CIT 595

8 - 25

Geometric Mean Consistency

- The results that we got when using System B and System C as reference machines are given below
- We find that Geometric Mean of A to Geometric Mean of B to be consistent regardless which system is taken as reference
 - $1.6733/1 = 2.4258/1.4497$

System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
Geometric Mean	1.6733		1		0.6898

System A Execution Time	Execution Time Normalized to C	System B Execution Time	Execution Time Normalized to C	System C Execution Time	Execution Time Normalized to C
Geometric Mean	2.4258		1.4497		1

CIT 595

8 - 26

Harmonic Mean

- Provides us with a way to compare execution times that are expressed as a rate
 - E.g. operations per second
- The harmonic mean allows us to form a mathematical expectation of throughput, and to compare the relative throughput of systems and system components
- To find the harmonic mean, we add the reciprocals of the rates and divide them into the number of rates:

$$H = n \div (1/x_1 + 1/x_2 + 1/x_3 + \dots + 1/x_n)$$

CIT 595

8 - 27

Harmonic Mean

- Good predictor of the expected behavior
 - Without comparing it to another machine (geometric mean)
- The slowest rates have the greatest influence on the result
 - Hence it identifies areas where performance can be improved

CIT 595

8 - 28

Objective Assessment

- The objective assessment of computer performance is most critical when deciding which one to buy
 - For enterprise-level systems, this process is complicated, and the consequences of a bad decision are grave
- Unfortunately, computer sales are as much dependent on good marketing as on good performance
- The wary buyer will understand how objective performance data can be slanted to the advantage of anyone giving a sales pitch

CIT 595

8 - 29

Objective Assessment (contd..)

- The most common deceptive practices include:
 - Selective statistics: Citing only favorable results while omitting others
 - Citing only peak performance numbers while ignoring the average case
 - Vagueness in the use of words like “almost,” “nearly,” “more,” and “less,” in comparing performance data
 - The use of inappropriate statistics or “comparing apples to oranges.”
 - Implying that you should buy a particular system because “everyone” is buying similar systems

CIT 595

8 - 30

Benchmarking

- Performance benchmarking is the science of making objective assessments concerning the performance of one system over another
- For a while, there were no definitive benchmark that could tell you which system will run *your* applications better than other based on the metric you desire (throughput or CPU time)

CIT 595

8 - 31

SPEC Benchmarks

- In 1988 the Standard Performance Evaluation Corporation (SPEC) was formed to address the need for objective benchmarks
- SPEC produces benchmark suites for various classes of computers and computer applications
- Their most widely known benchmark suite is the SPEC CPU benchmark
- The SPEC CPU2000 benchmark consists of two parts, CINT2000, which measures integer arithmetic operations, and CFP2000, which measures floating-point processing

CIT 595

8 - 32

SPEC Benchmarks

- The SPEC benchmarks consist of a collection of kernel programs
- These are programs that carry out the core processes involved in solving a particular problem
 - Activities that do not contribute to solving the problem, such as I/O routines are removed
- CINT2000 consists of 12 applications (11 written in C and one in C++)
- SPEC also has Benchmark suite that measures performance of Java Virtual Machine called JVM98

CIT 595

8 - 33

Transactional Processing Benchmarks

- The SPEC CPU benchmark evaluates only CPU performance
- If we interested in systems that involve exchange of items of value (transaction), such as information, goods, services and money
 - The processing of such information is known as transaction processing
- When the performance of the entire system under high transaction loads is a greater concern, the *Transaction Performance Council* (TPC) benchmarks are more suitable
- TPC-C suite models the transactions typical of a programs dealing with warehousing and distribution of items

CIT 595

8 - 34