Measuring Performance

November 17, 2008

Outline

1 Introduction

2 CPU Peformance and Its Factors

3 Evaluating Performance

Some measures of Peformance

- High throughput
- Short respone time
- Scalability

Performance

- "How fast?"

Performance

- "How fast?"
- Measure, evaluate

Performance

- "How fast?"
- Measure, evaluate
- Tough: complex software + optimized hardware
- Hardware:
  - CPU: many metrics

Performance

- "How fast?"
- Measure, evaluate
- Tough: complex software + optimized hardware
- Hardware:
    - CPU: many metrics
    - Memory
    - Graphics

Performance

- "How fast?"
- Measure, evaluate
- Tough: complex software + optimized hardware
- Hardware:
  - CPU: many metrics
  - Memory
  - Graphics
- Software: needs vary

Performance

- "How fast?"
- Measure, evaluate
- Tough: complex software + optimized hardware
- Hardware:
  - CPU: many metrics
  - Memory
  - Graphics
- Software: needs vary
- Consumer: **GHz**, **MB** . . . ?

Our Concern

- From inside
- What determines computer performance?

Our Concern

- From inside
- What determines computer performance?
- Answer:
  - Why is a software slow / fast?
  - Why implementations of ISs can perform better / worse?
  - How hardware pieces affect performance?

Defining Performance

$$\texttt{Performance}_{\texttt{ComputerX}} > \texttt{Computer}_{\texttt{ComputerY}}$$

Defining Performance

$$\texttt{Performance}_{\texttt{ComputerX}} > \texttt{Computer}_{\texttt{ComputerY}} \; ?$$

Judging Performance

Aircraft	Passenger Capacity	Fuel Capacity	Cruising Range	Cruising Speed	Throughput	Cost
Boeing 747-400	421	216,847	10,734	920	387,320	0.048
Boeing 767-300	270	91,380	10,548	853	230,310	0.032
Airbus 340-300	284	139,681	12,493	869	246,796	0.039
Airbus 319-100	120	23,859	4,442	837	100,440	0.045
BAE-146-200	77	11,750	2,406	708	54,516	0.063
Concorde	132	119,501	6,230	2180	287,760	0.145
Dash-8	50	3,202	1,389	531	26,550	0.046
My car	5	60	700	100	500	0.017

Figure: Statistics of some aircraft models

Basic Metrics

- **Response time**: time for single task
- **Throughput**: amount of work done per unit time

Throughput and Response Time

- Which is improved in these cases:
    - Upgrading to a faster processor?
    - Upgrading to a multi-core processor?

Response Time

- Main metric

Response Time

- Main metric
- `max(Performance)` ⇒ min(respTime)

Response Time

- Main metric
- $\mathtt{max(Performance)} \Rightarrow \min(\text{respTime})$

$$\mathtt{Performance} = \frac{1}{\mathtt{ExecutionTime}}$$

Response Time

- Main metric
- $\texttt{max(Performance)} \Rightarrow \min(\text{respTime})$

$$\texttt{Performance} = \frac{1}{\texttt{ExecutionTime}}$$

$$\therefore, \text{if } \texttt{Performance}_{\texttt{X}} > \texttt{Performance}_{\texttt{Y}}$$
$$\implies \texttt{ExecutionTime}_{\texttt{Y}} > \texttt{ExecutionTime}_{\texttt{X}}$$

Relative Performance

X is $\mathtt{n}$ times faster than Y

$$\Longrightarrow \frac{\mathtt{Performance}_{\mathtt{X}}}{\mathtt{Performance}_{\mathtt{Y}}} = \frac{\mathtt{ExecutionTime}_{\mathtt{Y}}}{\mathtt{ExecutionTime}_{\mathtt{X}}} = n$$

Relative Performance

X is n times faster than Y

$$\implies \frac{\texttt{Performance}_\texttt{X}}{\texttt{Performance}_\texttt{Y}} = \frac{\texttt{ExecutionTime}_\texttt{Y}}{\texttt{ExecutionTime}_\texttt{X}} = n$$

$\therefore$, *improve performance* $\implies$ *descrease execution time*

Measuring Performance

- Settled on **time** as metric
- Execution time / program

Measuring Performance

- Settled on **time** as metric
- Execution time / program
- Formally *wall-clock/execution/response* time
  - Total time to complete everything for a task (CPU + I/O + Memory + OS + . . .)

CPU Execution Time

- Often, multi-processing OS
- $\therefore$, goal: throughput

CPU Execution Time

- Often, multi-processing OS
- ∴, goal: throughput
- **CPU (Execution) time**: CPU's devotion to a single program
  - exclude I/O or other programs

CPU Execution Time

- Often, multi-processing OS
- $\therefore$, goal: throughput
- **CPU (Execution) time**: CPU's devotion to a single program
  - exclude I/O or other programs
- = user CPU time + system CPU time

Clock Cycles

- More precise metric
- **Clock cycles**
  - Also, tick, clock tick, period, clock period
- Measured by a crystal oscillator
- **Clock rate/period**

Outline

CPU Performance

- Relate user and designer metrics
- For a program:

$$\text{CPU Execution time} = \text{CPU clock cycles} * \text{Clock cycle time}$$
$$= \frac{\text{CPU clock cycles}}{\text{Clock rate}}$$

- Execution time: user experience
- Cycles and clock rate: designer metrics

Example

- Program execution on CPU **A** (4 GHz)
  - 10 seconds
- Same program on future CPU **B**
  - 6 seconds (hopefully)
- Improved design will cost 1.2 times more cycles for same program
- $\texttt{ClockSpeed}_{B}$?
  - 8 GHz

Cycles Per Instruction

- Relate to number of instructions in program
- **Clock Cycles per Instruction** (CPI)
  - average for all instructions
  - examples for some instructions costlier than some others?
- CPI: Good measure of implementations of same architecture
- $\therefore$, for a program:

$$\text{Cycles} = \text{No. of instructions} \times \text{Average CPI}$$

Rephrasing the Performance Equation

$$\begin{aligned}\text{CPU time} &= \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}\\ &= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}\end{aligned}$$

Measuring the factors

- **CPU time**: use a watch
- **Clock rate/period**: comes with CPU specs
- **Instructions Count**:
  - Software: profilers, hardware simulators
  - Hardware: CPU counters
- **CPI**:
  - hardware counters

Measuring the factors

- **CPU time**: use a watch
- **Clock rate/period**: comes with CPU specs
- **Instructions Count**:
  - Software: profilers, hardware simulators
  - Hardware: CPU counters
- **CPI**:
  - hardware counters
- No. of cycles
  - $\sum_{i=1}^{n} CPI_i \times C_i$

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program
  - Instruction count, CPI

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program
  - Instruction count, CPI
- Compiler for the program

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program
  - Instruction count, CPI
- Compiler for the program
  - Instruction count, CPI

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program
  - Instruction count, CPI
- Compiler for the program
  - Instruction count, CPI
- Instruction Architecture for the program

Component Effect on Performance

- What are the factors that affect performance of a program and on what metrics?
- Algorithm of the program
  - Instruction count, (possibly) CPI
- Programming language of the program
  - Instruction count, CPI
- Compiler for the program
  - Instruction count, CPI
- Instruction Architecture for the program
  - all: instruction count, CPI, clock rate

Example of Code Segment Performance

- Decision for a compiler designer
- Consider following CPI's

	CPI for this instruction class		
	A	**B**	**C**
CPI	1	2	3

Example of Code Segment Performance

- Decision for a compiler designer
- Consider following CPI's

	CPI for this instruction class		
	A	**B**	**C**
CPI	1	2	3

- For a HLL statement, alternative AL code sequences

	Instruction counts for instruction class		
	A	**B**	**C**
x	2	1	2
y	4	1	1

Example of Code Segment Performance

- Decision for a compiler designer
- Consider following CPI's

	CPI for this instruction class		
	A	**B**	**C**
CPI	1	2	3

- For a HLL statement, alternative AL code sequences

	Instruction counts for instruction class		
	A	**B**	**C**
x	2	1	2
y	4	1	1

- Calculate clock cycles for both

Example of Code Segment Performance

- Decision for a compiler designer
- Consider following CPI's

	CPI for this instruction class		
	A	**B**	**C**
CPI	1	2	3

- For a HLL statement, alternative AL code sequences

	Instruction counts for instruction class		
	A	**B**	**C**
x	2	1	2
y	4	1	1

- Calculate clock cycles for both
- Calculate CPI for both sequences
- ∴, look at all 3 factors when evaluating performance

Another Example

- A Java program runs in 12 seconds on a PC
- New Java compiler will need 0.6 times the instructions
  - increases CPI by 1.1
- Compare performance of program with new compiler

Effect of Improvement

- Consider a program with: 25% floating point + 75 % rest

Effect of Improvement

- Consider a program with: 25% floating point + 75 % rest
- If FP improves by factor of 5, total improvement?

Effect of Improvement

- Consider a program with: 25% floating point + 75 % rest
- If FP improves by factor of 5, total improvement?
- Amdahl's law:

$$\text{ET after improvement} = \text{ET of unimproved part} + \frac{\text{ET of improved part}}{\text{amount of improvement}}$$

Outline

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers
- **Benchmark**
  - set of representative programs run on different programs

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers
- **Benchmark**
  - set of representative programs run on different programs
- Benchmark applications should come from real world
  - User-environment specific
- Reproducible

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers
- **Benchmark**
  - set of representative programs run on different programs
- Benchmark applications should come from real world
  - User-environment specific
- Reproducible
  - Choose inputs that trigger commonly-used instructions

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers
- **Benchmark**
  - set of representative programs run on different programs
- Benchmark applications should come from real world
  - User-environment specific
- Reproducible
  - Choose inputs that trigger commonly-used instructions
  - Input affects memory most

Performance Evaluation

- **Workload**: set of programs run on a computer
  - To evaluate, user runs workload on different computers
- **Benchmark**
  - set of representative programs run on different programs
- Benchmark applications should come from real world
  - User-environment specific
- Reproducible
  - Choose inputs that trigger commonly-used instructions
  - Input affects memory most
- Optimization (compiler, IA, hardware, . . .) may be benchmark-specific

Comparing and Summarizing Performance

Results of a Benchmark

	Computer A	Computer B
Program 1 (secs)	1	10
Program 2	1000	100
Total Time	1001	110

Artithmetic Mean

- If running multiple programs

$$AM = \frac{1}{n}\sum_{i=1}^{n} \texttt{Time}_\texttt{i}$$

where, $Time_i$ is execution time for ith program

Artithmetic Mean

- If running multiple programs

$$AM = \frac{1}{n}\sum_{i=1}^{n} \mathtt{Time_i}$$

where, $Time_i$ is execution time for ith program

- Weighted

$$AM = \frac{1}{n_{total}}\sum_{i=1}^{n} \mathtt{Time_i} \times \mathtt{freq_i}$$

SPEC benchmark

- Weighted running time for programs
- Source code
  - Fortran or C
  - Compiled
- Web servers, floating point, file transfer

MIPS

- **M**iilion **I**nstructions **P**er **S**econd

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6}$$

MIPS

- **M**iilion **I**nstructions **P**er **S**econd

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time } \times 10^6}$$

- Intuitive

MIPS

- **M**iilion **I**nstructions **P**er **S**econd

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time } \times 10^6}$$

- Intuitive
- Problems
  - cannot compare different IAs
  - differs for different programs