Performance Assessment

Electrical and Computer Engineering

Florida International University

Fall 2009

Performance

- What we care most about the computer performance
  - How fast it can run a program
  - Response time or throughput
    - Response time: time to finish one single program
    - Throughput: total amount of work done in unit time

CPU Performance Equation

- CPU Time

$$\text{CPU time} = \frac{\text{clock cycles for a program (cycles)}}{\text{clock rate (cycles/sec)}}$$

- If we know
  - Total instruction counts (IC)
  - Cycles per instruction (CPI)
  - Cycle times (the inverse of the clock rate)

$$\text{CPU time} = \frac{IC \times CPI}{\text{clock rate}} = IC \times CPI \times \text{cycle\_time}$$

What if different instructions have different CPIs

- CPU time

$$\text{CPU time} = \left(\sum_i IC_i \times CPI_i\right) \times cycle\_time$$

- Overall CPI

$$\text{CPI} = \frac{\sum_i IC_i \times CPI_i}{IC_{total}} = \sum_i \frac{IC_i}{IC_{total}} \times CPI_i$$

Improve CPU time

- Instruction count
  - ISA and compiler technology
- CPI
  - Organization and ISA
- Clock cycle time
  - Hardware technology and organization

Speed measurement

Speedup n:

$$n = \frac{\text{execution time of Y}}{\text{execution time of X}} = \frac{1/\text{performance Y}}{1/\text{performance X}} = \frac{\text{performance of X}}{\text{performance of Y}}$$

X performs n times better than Y

Example

- 400-mhz processor
- 2 million instructions
- CPU Time?

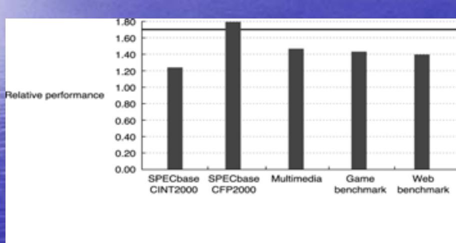
Instruction Type	CPI	Instruction Mix
ALU	1	60%
Load/Store with cache hit	2	18%
Load/store with cache miss	8	10%
Branch	4	12%

MIPS and MFLOPS

- MIPS (million instructions per second)
  - MIPS = IC/(CPU Time x 10^6)
  - Problems?
    - High MIPS ≠ shorter CPU time
- MFLOPS (million floating point operations per second)
  - MFLOPS = floating point operations in a program / (CPU Time x 10^6)
  - Problems?

True/False

- *Two processors with same ISA can be judged by clock rate or with single benchmark suite*



Performance Comparison

A is 10x faster than B for Prog P1
B is 10x faster than A for Prog P2
A is 20x faster than C for Prog P1
C is 50x faster than A for Prog P2
B is 2x faster than C for Prog P1
C is 5x faster than B for Prog P2

Which one is faster ?

Using total execution time

	Computer 1	Computer 2	Computer 3
Program A	1	10	20
Program B	1000	100	20
Total	1001	110	40

Both program A and B run equal number of times.

Arithmetic Mean

$$\frac{1}{n}\sum_{i=1}^{n} time_i$$

What if Program A and B run different times?

Weighted arithmetic mean example

Arithmetic mean (weighted)

$$\sum_{i=1}^{n} weight_i * time_i$$

	Comp A	Comp B	Comp C
Prog 1	1	10	20
Prog 2	1000	100	20

	W_1=.5, W_2=.5	W_1=.909,W_2=.091	W_1=.999,W_2=.001
Computer A	500.5	91.909	1.999
Computer B	55	18.10	10.09
Computer C	20	20	20

Normalized Execution Time

- Normalize to a particular machine by dividing all execution times by chosen machine's time
- Example

Program P1 has the following execution times:

On machine A: 10 secs

On machine B: 100 secs

On machine C: 150 secs

Normalized to A: A=1, B=10, C=15

Normalized to B: A=.1, B=1, C=1.5

Execution time ratio

Normalized Mean

Taking the average of the normalized times

Normalized geometric mean

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Normalized Geometric Mean Example

- Two programs and three machines

	Comp A	Comp B	Comp C
Prog 1	1	10	20
Prog 2	1000	100	20

- ETR (Execution time ratio)

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
ETR P1	1	10	20	0.1	1	2	.05	.5	1
ETR P2	1	.1	.02	10	1	.2	50	5	1

- NGM (Normalized geometric mean)

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
NGM	1	1	.63	1	1	.63	1.58	1.58	1

Amdahl's Law

Improvement by the faster mode is limited by the fraction of time the faster mode can be used

Execution time of any code has two portions

$C_{total} = C_{p1} + C_{p2}$

$C_{p1} = (1-\alpha) * C_{total}$: not affected by enhancement
$C_{p2} = \alpha * C_{total}$: affected by enhancement

Let n be the speedup factor for C_{p2}, then

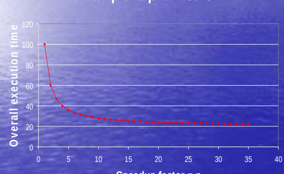
$C_{new} = C_{p1} + C_{p2}/n = ((1-\alpha) + \alpha/n) * C_{total}$

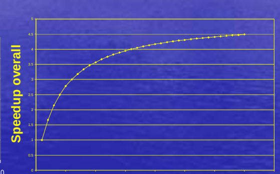
As n -> infinity, C_{new} -> $(1-\alpha) * C_{total}$

Amdahl's Law

$$\text{execution time}_{new} = (1-\alpha) * \text{execution time}_{old} + (\alpha) * \frac{\text{execution time}_{old}}{n}$$

Example: alpha = 80%





$$\text{Speedup}_{overall} = \frac{\text{execution time}_{old}}{\text{execution time}_{new}} = \frac{1}{(1-\alpha) + \frac{\alpha}{n}}$$

Example

- Enhancement: Vector mode
- Portions of code containing computations run 20x faster in vector mode.
- What % of original code must be vectorizable to achieve $speedup_{overall} = 2$?

Example

- Enhancement: Vector mode
- Portions of code containing computations run 20x faster in vector mode.
- What % of original code must be vectorizable to achieve $speedup_{overall} = 2$?

$$\text{Speedup}_{\text{overall}} = \frac{\text{execution time}_{\text{old}}}{\text{execution time}_{\text{new}}} = \frac{1}{(1-\alpha)+\frac{\alpha}{n}}$$

$$2 = \frac{1}{(1-\alpha)+\frac{\alpha}{20}} \qquad \alpha = .5263$$

Example

- FP operations =25%
- FP operation AVG CPI = 4.0
- AVG CPI for others = 1.33
- FP operatios for FPSQR = 2%
- CPI of FPSQR = 20
- Design 1: decrease the CPI of FPSQR to 2
- Deisgn 2: decrease average CPI of all FP to 2.5.
- Which one is better?

Summary

- Measure performance
  - Execution time/throughput
  - CPU time
- Fair comparison of performance
  - Weighted mean,
  - normalization, geometric mean
- Principles in architecture design
  - Amdahl's law