CALIFORNIA STATE UNIVERSITY FULLERTON

CPSC-440 Computer System Architecture

Lecture 2
Performance Assessment

Performance

- What we care most about…
  - How fast the computer 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

$$CPU\ Time = \frac{Clock\ cycles\ for\ a\ program\ (cycles)}{Clock\ Freq\ (cycles/sec)}$$

- If we know…
  - Total Instruction Counts (I_c)
  - Cycles Per Instruction (CPI)
  - Clock Frequency (f)
  - Cycle Time (τ) - the inverse of the clock frequency ($1/f$)
  - CPU Time (T):

$$T = \frac{I_c \times CPI}{f} = I_c \times CPI \times \tau$$

What if different instructions have different CPIs?

- CPU Time

$$T = \left(\sum_{i=1}^{n} (I_i \times CPI_i) \right) \times \tau$$

  - Where i is the instruction type
- CPI

$$CPI = \frac{\sum_{i=1}^{n} (I_i \times CPI_i)}{I_c}$$

- IPC (Instructions Per Cycle)
  - Inverse of CPI

MIPS and MFLOPS Rates

- MIPS (Millions of Instructions Per Second) Rate

$$MIPS\ Rate = \frac{I_C}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

- MFLOPS (Millions of Floating Point Operations Per Second) Rate

$$MFLOPS\ Rate = \frac{\#\ of\ executed\ floating\ point\ operations\ in\ a\ program}{Execution\ time \times 10^6}$$

Example

- 2 million instructions on a 400 MHz processor
- 4 major types of instructions
- What's the MIPS rate?

$$CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$$

$$MIPS\ Rate = (400 \times 10^6)/(2.24 \times 10^6) \approx 178$$

Instruction Type	CPI_i	I_i (%)
Arithmetic and Logic	1	60
Load/Store with Cache Hit	2	18
Branch	4	12
Memory Reference with Cache Miss	8	10

Improve CPU time

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

Benchmarks

- MIPS and MFLOPS rates are inadequate to evaluate performance of processors
  - Because of differences in instruction sets, these rates are not valid means of comparing the performance of different architectures

Example

$$A = B + C$$

Assume all quantities in main memory

Complex Instruction Set Computer (CISC)	Reduced Instruction Set Computer (RISC)

Complex Instruction Set Computer (CISC)

- Can be compiled into one instruction
- Rated at 1 MIPS

```
add    mem(B), mem(C), mem(A)
```

Reduced Instruction Set Computer (RISC)

- Rated at 4 MIPS

```
load  mem(B), reg(1)
load  mem(C), reg(2)
add   reg(1), reg(2), reg(3)
store reg(3), mem(A)
```

Standard Performance Evaluation Corporation (SPEC) Benchmark

- Benchmark Suite
  - Collection of programs
  - Provides a representative test of a computer in a particular application or area

Performance Comparison

Which One is Faster?
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

Total Execution Rates

- Both program A and B have equal number of instructions
- Below shows the execution rates

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

Average Execution Rate

- What if Program A and B have a different number of instructions?
- If there are m different benchmark programs

$$R_A = \frac{1}{m}\sum_{i=1}^{m} R_i$$

  - Where R_i is the high-level language instruction execution rate for the i^{th} benchmark program
- The throughput of a machine carrying out a number of tasks
  - The higher the rate (R_A) the better

Harmonic Mean

- Alternative to average execution rate

$$R_H = \frac{m}{\sum_{i=1}^{m} \frac{1}{R_i}}$$

- The reciprocal of the arithmetic mean of the reciprocals
- Gives the inverse of the average execution rate
- Again, the higher the rate (R_H) the better

Total Execution Time Example

The top table shows the execution rates. Assume each program has equal weight.

	Computer A	Computer B	Computer C
Program 1	100	10	5
Program 2	0.1	1	5
Program 3	0.2	0.1	2
Program 4	1	0.125	1

	R_A	Rank	R_H	Rank
Computer A	25.325	1	0.25	2
Computer B	2.8	3	0.21	3
Computer C	3.25	2	2.1	1

SPEC Benchmark
Speed Metrics

- Measures the ability of a computer to complete a single task

$$r_i = \frac{Tref_i}{Tsut_i}$$

  - $Tref_i$ - execution time of benchmark program i on the reference system
  - $Tsut_i$ - execution time of benchmark program i on the system under test
- The larger the ratio, the higher the speed

SPEC Benchmark
Speed Metrics

- Example
  - A system executes a program in 934 sec.
  - The reference implementation requires 22,135 sec.

$$\frac{22{,}135\ sec}{934\ sec} = 23.7$$

SPEC Benchmark
Rate Metric

- Throughput/rate of a machine carrying out a number of tasks
- Multiple copies of benchmarks run simultaneously

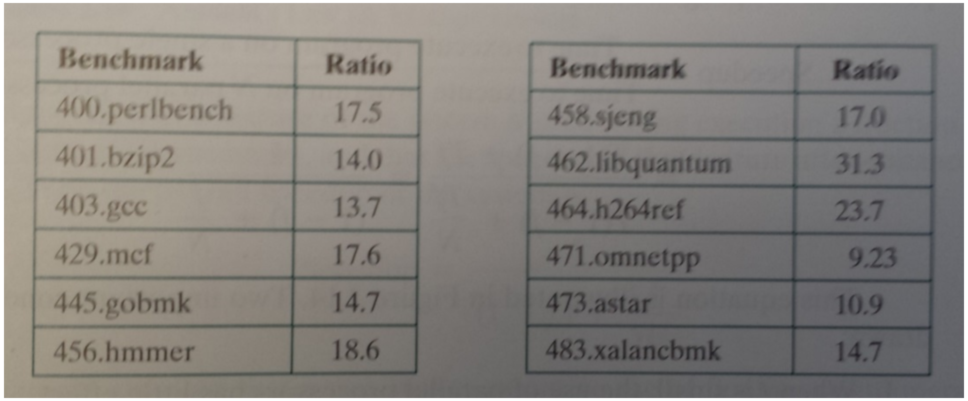
$$r_i = \frac{N \times Tref_i}{Tsut_i}$$

  - N – number of copies of the program that are run simultaneously

SPEC Benchmark
Geometric Mean

- Averages ratios for all 12 integer benchmarks
- Used to determine the overall performance measure

$$r_G = \left(\prod_{i=1}^{n} r_i \right)^{1/n}$$



Benchmark	Ratio
400.perlbench	17.5
401.bzip2	14.0
403.gcc	13.7
429.mcf	17.6
445.gobmk	14.7
456.hmmer	18.6

Benchmark	Ratio
458.sjeng	17.0
462.libquantum	31.3
464.h264ref	23.7
471.omnetpp	9.23
473.astar	10.9
483.xalancbmk	14.7

$$(17.5 \times 14 \times 13.7 \times 17.6 \times 14.7 \times 18.6 \times 17 \times 31.3 \times 23.7 \times 9.23 \times 10.9 \times 14.7)^{1/12} = 18.5$$

Amdahl's Law

- *Speedup in one aspect of technology/design does not result in a corresponding improvement in performance*

$$Speedup = \frac{Execution\ time\ before\ enhancement}{Execution\ time\ after\ enhancement}$$

Amdahl's Law Example

- Single vs. Multiple processors

$$Speedup = \frac{X}{Y} = \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

  - X: Time to execute a program on a single processor
  - Y: Time to execute a program on N parallel processors
  - T: Total execution time
  - f: Fraction of code executed on parallel processors (no scheduling overhead)
  - $(1-f)$: Fraction of code executed on a single processor

1. When f is small, the use of parallel processors has little effect
2. As $N \rightarrow \infty$, speedup bound by $1/(1-f)$
   - Diminishing returns for using more processors

Amdahl's Law Example