Performance: Measurement and Analysis

Chapter 2

Performance Measurement and Analysis

- Performance
  - Speedup
- Performance Measurement
  - Response Time and Throughput
  - Measurement Issues
- Benchmarks
  - SPEC Benchmark Suites (SPEC.org)
- Performance Comparison
- CPU Time
- Amdahl's Law
- Performance Improvement

Performance

- ★ Performance is usually context dependent (relative).
- ★ Performance is the inverse function of Execution Time.
- ★ How much faster is X than Y?
  - Depending on execution time

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

$$Speedup(n) = \frac{Performance_X(1/seconds)}{Performance_Y(1/seconds)}$$

$$Speedup(n) = \frac{Performance_x}{Performance_y} = \frac{\frac{1}{Execution\ Time_x}}{\frac{1}{Execution\ Time_y}} = \frac{Execution\ Time_Y}{Execution\ Time_X}$$

Hint:
1.High performance means short time. Low performance means long time.
2. X is n time faster than Y. n >1 means faster. n<1 means slower.

Performance (ctd.)

- ★ Computer A runs a program in 20 seconds.
- ★ Computer B runs a program in 25 seconds.
- ★ How much faster is A than B?

$$Speedup = \frac{[Execution\,Time_Y]}{[Execution\,Time_X]}$$

$$Speedup = \frac{[Execution\,Time_B]}{[Execution\,Time_A]} = \frac{25}{20} = 1.25$$

Performance Measurement

$$Elapse\ Time = CPU\ Time + I/O\ Time + Memory\ Time$$

- Response Time
  - Elapsed Time - Wall clock
  - CPU Time - Only CPU time (without the wait time) … we will revisit this later.
  - From O.S. class, we use: user time, real time, sys time.
- Throughput
  - Tasks per time unit
- Units for commercialization (Not a real performance)
  - MIPS
    - Millions of instructions per second
  - FLOPS
    - Floating Point Operations per second

Response Time

- Timer using clock on the wall (in second)
  - Can be measure by Unix time command.
  - For Windows use ptime (require powershell).

```
$ time gcc -o a.out test.c

real    0m0.180s

user    0m0.040s

sys             0m0.012s
```

Throughput

- Number of tasks that can be completed in a unit of time.
- Can be referred as parallelism at task level.

Computer A performs 5 tasks in 10 minutes. All tasks also finish at the same time.

Throughput 0.5 task per minute

Response Time 10 minutes.

Computer B performs each task in 3 minutes. However, the task must be executed sequentially.

Throughput 0.33 task per minute

Response Time 3 minutes.

Consider transferring 20 TB of data between two building using high-speed fiber-optic cable (1Gb/s), this can take up to 40 hours. However, a man carrying 20x1TB hard drive in a backpack can walk between the building in 15 minutes.

Though fiber-optic cable is faster, but can we say that a man has higher throughput?

Measurement Issues

- What to measure
  - What program(s) to run?
- Interference
  - Loaded / Unloaded jobs on multi-programmed system
- Reproducibility
  - Can the measurement be repeated?
- Comparability
  - A is good at multimedia processor, but is poor at others.
  - B is poor at multimedia processor, but is good at others.
  - Which one is better?

Benchmarks

- ★ Real benchmark - using real applications
  - Scripted application to reproduce interactive or multi-user behaviors
- ★ Microbenchmark - using a part of software
- ★ Kernels - only execute certain aspects of performance
- ★ Toy benchmark - small programs (e.g. quick sort, merge sort). Widely used in the past.
- ★ Synthetics benchmarks - try to mimic behavior of real programs (e.g. scientific workloads)
  - Wheatstone
  - Dhrystone

A unified set is required for comparison.

SPEC Benchmark Suites

- System Performance Evaluation Cooperation (SPEC.org) - An organization that create benchmark suites.
  - SPEC CPU 2017 - for CPU
  - SPECviewpref - for graphics calculation
  - SPECjbb2015 - for java virtual machine
  - SPEC SFS 2014 - for file server
  - .... visit SPEC.org for more details

SPECrate 2017 Integer	SPECspeed 2017 Integer	Language [1]	KLOC [2]	Application Area
500.perlbench_r	600.perlbench_s	C	362	Perl interpreter
502.gcc_r	602.gcc_s	C	1,304	GNU C compiler
505.mcf_r	605.mcf_s	C	3	Route planning
520.omnetpp_r	620.omnetpp_s	C++	134	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	96	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	33	General data compression

Performance Comparison

★ How do we compare machines using collections of execution times?

★ Choices
- Summation
- Arithmetic Mean/Weighted Arithmetic Mean
- Harmonic Mean/Weighted Harmonic Mean
- Geometric Mean/Weighted Geometric Mean

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

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

When to use each one?

$$\sqrt[n]{\prod^{n} x_i}$$ Geometric Mean

(Weighted) Arithmetic Mean

- ★ Which one is the fastest?
- ★ Which one will you buy?

	Computer A (secs)	Computer B (secs)	Computer C (secs)
Program 1 (80%)	10	4	5
Program 2 (20%)	2	10	5
Total	12	14	10
Arithmetic Mean	6	7	5
Weighted Arithmetic Mean	8.4	5.2	5

Geometric Mean and Speed Up

★ Since performance measure is context dependent (relative), it is usually measured as a ratio (speed up).

★ For a ratio (no unit), Geometric mean should be used.

B is a base line.

	A (secs)	B (secs)	A/B (X)	C (secs)	A/C (Y)	C/B	X/Y
Program 1	10	4	2.5	5	2	1.25	1.25
Program 2	2	10	0.2	5	0.4	0.5	0.5
Arithmetic Mean	6	7	?	5	?	?	?
Geometric Mean			**0.71**		**0.89**	**0.79**	**0.79**

$$\sqrt[n]{\prod^{n} x_i}$$

Geometric Mean

Reporting Benchmark

- ★ With speed up and geometric mean, we can ignore workload.
- ★ For example,
  Improvement from 5 sec to 3 sec
  is equal to
  Improvement from 500 sec to 300 sec.
- ★ Nonetheless, Geometric mean (as well as speed up) cannot predict actual performance.

CPU Time

- ★ If we ignore the I/O (wait) time, CPU time can be calculated.

$$CPU\ Time = Instruction \times CPI_{average} \times Cycle\ Time$$

$$Cycle\ Time(seconds) = \frac{1}{Clock\ Rate(Hz)}$$

$$CPI_{average} = \sum_{i=1}^{n} CPI_i \times Weight_i$$

CPI = Clock Per Instruction

CPI and Cycle Time

- ★ Processor is based on a clock rate (usually in MHz or GHz). We can easily calculate a period of each clock cycle (Cycle Time).
- ★ For RISC processor, the cycle per instruction is usually fixed (e.g. 1 cycle for each instruction).

$$Cycle\ Time(seconds) = \frac{1}{Clock\ Rate(Hz)}$$

Average CPI

★ If each instruction class has different CPI, we can use weight to find an average CPI.

$$CPI_{average} = \sum_{i=1}^{n} CPI_i \times Weight_i$$

	CPI	Rate
class A	5	30%
class B	3	50%
class C	8	20%

$$CPI_{average} = (5 \times 0.3) + (3 \times 0.5) + (8 \times 0.2)$$

$$\therefore \quad CPI_{average} = 4.6$$

CPU TIME calculation

★ A 1 GHz processor execute 2000 millions instructions in 20 seconds. If the architect can double the clock to 2 GHz in exchange for lower average CPI by a factor of 1.2, how long does it take for this program to execute on a new architecture.

$$Clock\ Cycle\ Time = \frac{1}{(1\times 10^{9})}\ seconds$$

$$CPU\ Time(seconds) = Instruction \times CPI_{average} \times Cycle\ Time$$

$$CPI_{\text{เดิม}} = \frac{(20\ seconds)\times(1\times 10^{9}\ seconds\ per\ cycle)}{(2000\times 10^{6}\ instructions)}$$

$$CPI_{\text{เดิม}} = 10$$

$$CPI_{\text{ใหม่}} = 10\times 1.2 = 12$$

$$CPU\ TIME_{\text{ใหม่}} = \frac{(2000\times 10^{6}\times 1.2\times 10)}{(2\times 10^{9})}\ seconds$$

$\therefore$ CPU TIME $_{\text{ใหม่}}$ = 12 วินาที

MIPS

Despite the lower MIPS,
B is faster than A.

- ★ Bad unit for performance measured. Used by several architects.
- ★ Computer A is 10 MIPS running at 33 MHz. Computer B is 9 MIPS running at 20 MHz. If a same program uses 12,000,000 instructions and 10,000,000 instructions on A and B respectively, which computer is faster?

$$Cycle\ Time_A(seconds)=\frac{1}{Clock\ Rate}=\frac{1}{33\times10^6(Hz)}$$

$$CPU\ Time_A=Instruction\ Count_A\times CPI_A\times CycleTime_A$$

$$CPU\ Time_A=12\times10^6\times\left(\frac{1}{10\times10^6\times\frac{1}{33\times10^6}}\right)\times\frac{1}{3\times10^6}(seconds)$$

$$CPU\ Time_A=\frac{12\times10^6}{10\times10^6}=1.2(seconds)$$

$$Cycle\ Time_B(seconds)=\frac{1}{Clock\ Rate}=\frac{1}{20\times10^6(Hz)}$$

$$CPU\ Time_B=10\times10^6\times\left(\frac{1}{9\times10^6\times\frac{1}{20\times10^6}}\right)\times\frac{1}{20\times10^6}(seconds)$$

$$CPU\ Time_B=\frac{10\times10^6}{9\times10^6}=1.11\ (seconds)$$

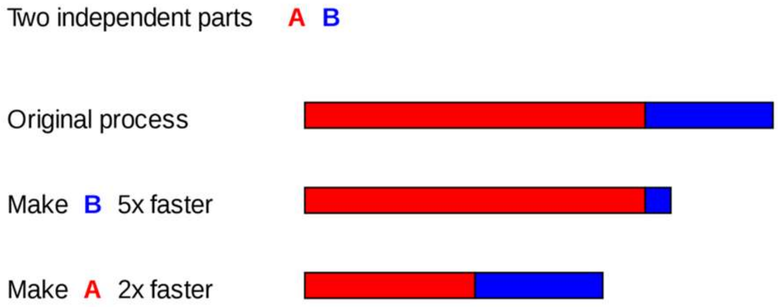
$$Speedup=\frac{1.2(seconds)}{1.1(seconds)}=1.09$$

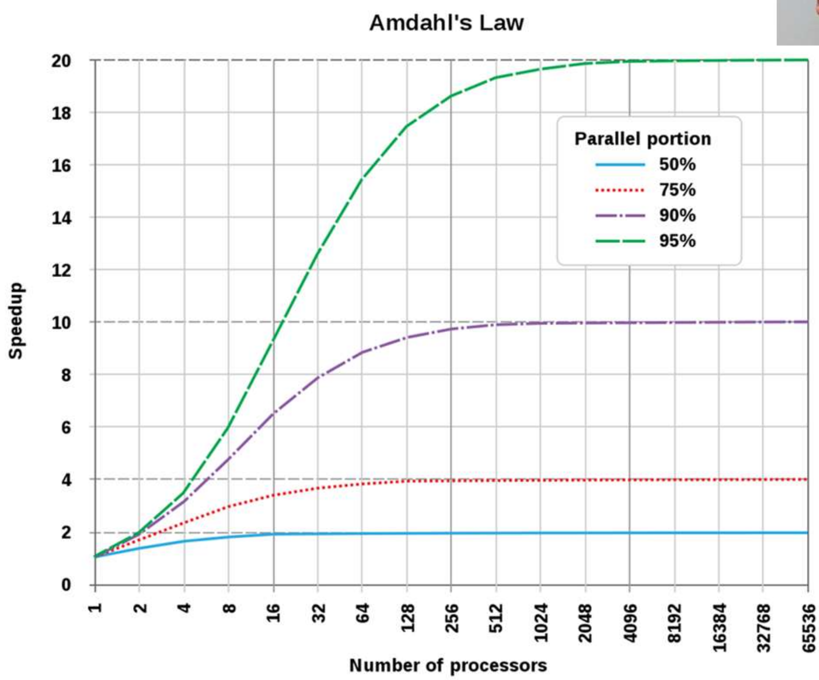
Amdahl’s Law

★ Law of diminish return

Gene M. Amdahl
1922-2015

To do grocery shopping, there are two parts: traveling (A), shopping (B).
Which optimization will be faster?
1. Travel 2 times faster
2. Shop 5 times faster





Images are taken from wikipedia

Amdahl's Law (ctd.)

$$Overall\ Speedup = \frac{1}{(1-P)+\frac{P}{n}}$$

★ If time required for buying meal (select and pay) is accounting for ⅓ of the meal time, what is the ideal speedup when we can buying 3-time faster.

$$Overall\ Speedup = \frac{1}{(1-(\frac{1}{3}))+(\frac{(\frac{1}{3})}{3})} = \frac{1}{\frac{2}{3}+\frac{1}{9}} = \frac{1}{\frac{6+1}{9}} = \frac{9}{7} \approx 1.29$$

Performance Improvement

- ★ With CPU Time function as a guideline, we can improve performance by minimizing the function. $CPU\ Time = Instruction \times CPI_{average} \times Cycle\ Time$
- ★ Reducing no. of Instruction
  - Redesign ISA?
  - Compiler Optimization?
- ★ Reducing (average) CPI
  - Redesign ISA?
  - Redesign internal CPU organization?
- ★ Reducing Cycle Time
  - Redesign internal CPU organization?
  - Improve transistor technology to make it faster?

Exercises

CPI and SpeedUp

- Suppose that we want to enhance the processor used for Mail Server. The new server is 10 times faster on computation in the mail server application.
  Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speed up gained by incorporating the enhancement?

CPI and SpeedUp

★ If we can make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application, calculate the speedup.

CPI and SpeedUp

- ★ Given the following measurements:
  - Frequency of FP operations = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
- ★ If we can decrease the average CPI of all FP operations to 2.5, calculate the speedup.

End of Chapter 2