System and Devices (SYS 3)

Lecture 4: Principles of Computer Design

Dr Pengcheng Liu

COM00018I

Measuring Performance

- When can we say one computer / architecture design is better than others?

  - Desktop PC – (execution time of a program)

  - Server (transactions / unit time)

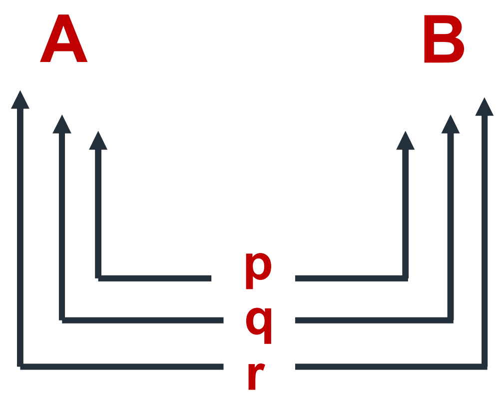
- When can we say X is n times faster than Y?

  - Execution $time_Y$ / Execution $time_X$ = n

  - $Throughput_X$ / $Throughput_Y$ = n

Measuring Performance

- Typical performance metrics:
  - Response time
  - Throughput
  - CPU time
  - Wall clock time
  - Speedup
- Benchmarks
  - Toy programs (e.g., sorting, matrix multiply)
  - Synthetic benchmarks (e.g., Dhrystone)
  - Benchmark suites (e.g., SPEC06, SPLASH)



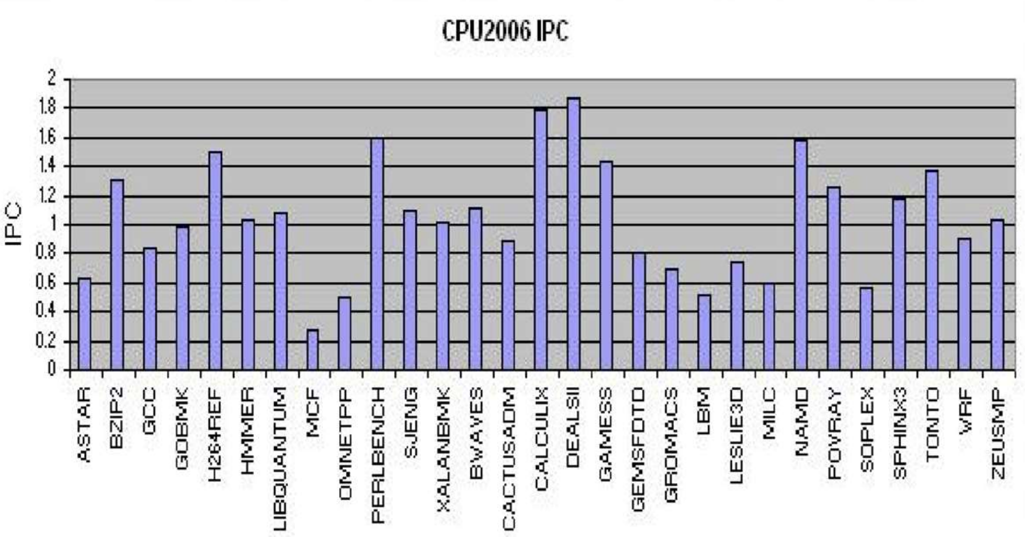
Benchmark Suite

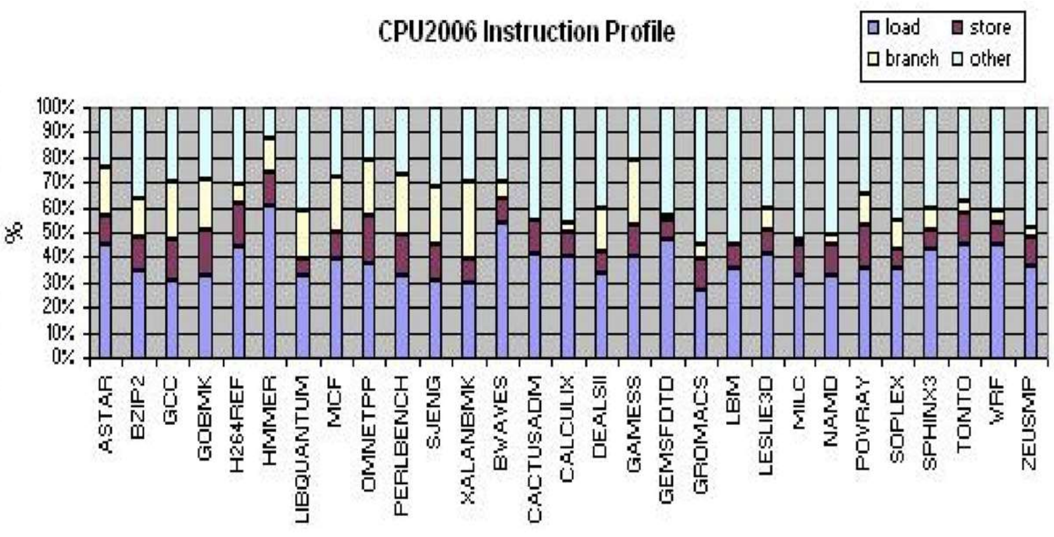
SPEC CPU2006 Programs

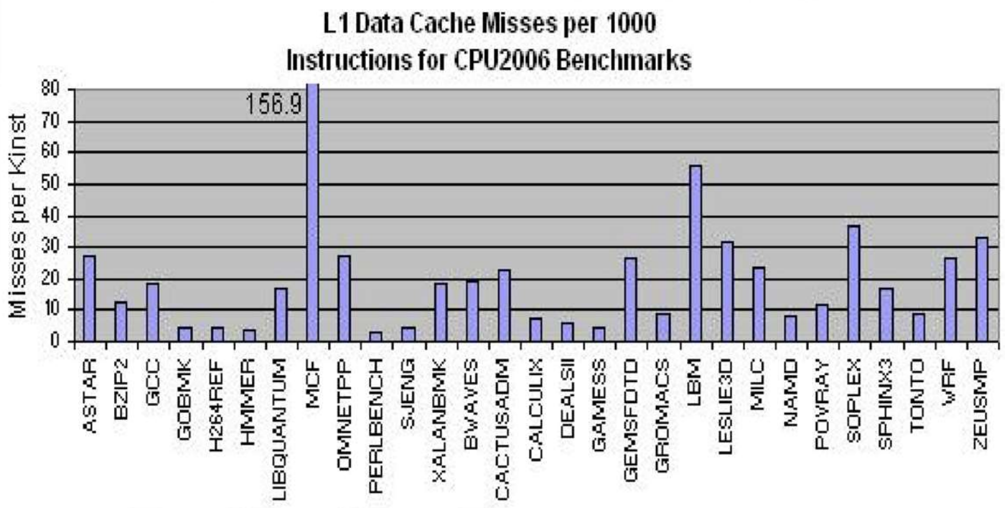
	Benchmark	Language	Descriptions
CINT2006 (Integer) 12 programs	400.perlbench	C	PERL Programming Language
	401.bzip2	C	Compression
	403.gcc	C	C Compiler
	429.mcf	C	Combinatorial Optimization
	445.gobmk	C	Artificial Intelligence: go
	456.hmmer	C	Search Gene Sequence
	458.sjeng	C	Artificial Intelligence: chess
	462.libquantum	C	Physics: Quantum Computing
	464.h264ref	C	Video Compression
	471.omnetpp	C++	Discrete Event Simulation
	473.astar	C++	Path-finding Algorithms
	483.Xalancbmk	C++	XML Processing
CFP2006 (Floating Point) 17 programs	410.bwaves	Fortran	Fluid Dynamics
	416.gamess	Fortran	Quantum Chemistry
	433.milc	C	Physics: Quantum Chromodynamics
	434.zeusmp	Fortran	Physics/CFD
	435.gromacs	C/Fortran	Biochemistry/Molecular Dynamics
	436.cactusADM	C/Fortran	Physics/General Relativity
	437.leslie3d	Fortran	Fluid Dynamics
	444.namd	C++	Biology/Molecular Dynamics
	447.dealII	C++	Finite Element Analysis
	450.soplex	C++	Linear Programming, Optimization
	453.povray	C++	Image Ray-tracing
	454.calculix	C/Fortran	Structural Mechanics
	459.GemsFDTD	Fortran	Computational Electromagnetics
	465.tonto	Fortran	Quantum Chemistry
	470.lbm	C	Fluid Dynamics
	481.wrf	C/Fortran	Weather Prediction
	482.sphinx3	C	Speech recognition

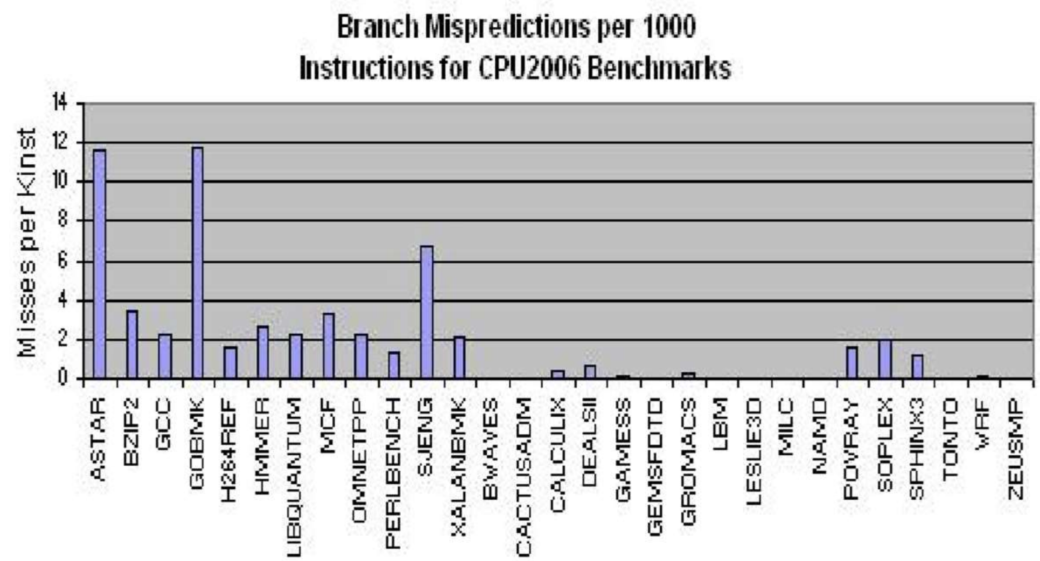
Source: http://www.spec.org › cpu2006

Benchmark Based Evaluation









SPEC Ratio

Reference for SPEC 2006:

Sun Ultra Enterprise 2 workstation with a 296-MHz UltraSPARC II processor

$$\text{SPECRatio}_A = \frac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_A}$$

$$\frac{\text{SPECRatio}_A}{\text{SPECRatio}_B} = \frac{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_A}}{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_B}} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

$$\text{Geometric mean} = \sqrt[n]{\prod_{i=1}^{n} sample_i}$$

Amdahl's Law

- Amdahl's Law defines the speedup that can be gained by improving some portion of a computer
- The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Amdahl's Law - Illustration

Example: Suppose that we want to enhance the floating-point operation of a processor by introducing a new advanced FPU unit. Let the new FPU is 10 times faster on floating point computations than the original processor. Assuming a program has 40% floating point operations, what is the overall speedup gained by incorporating the enhancement?

$$Speedup_{overall} = \frac{1}{(1 - Fractionenhanced_{\text{部}}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

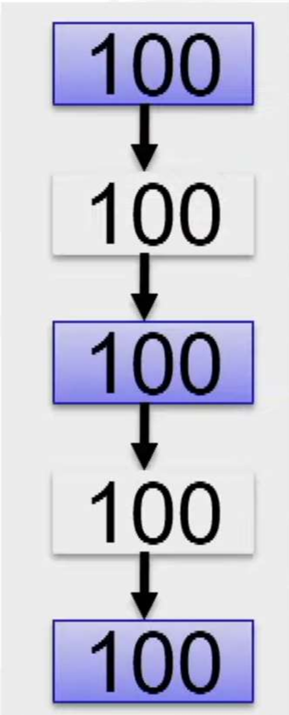
Solution:

Fraction enhanced = 0.4

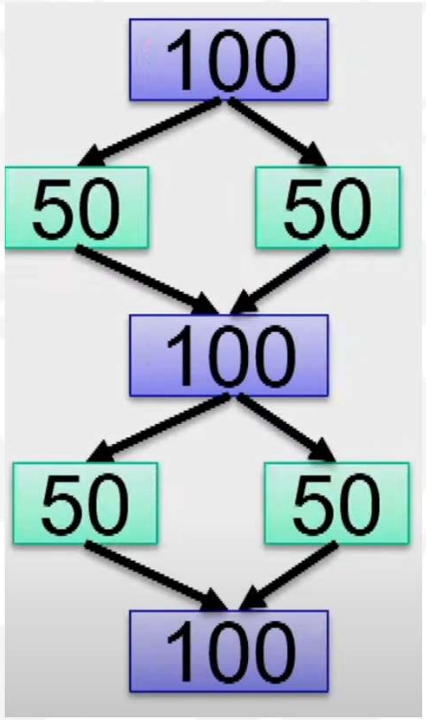
Speedup enhanced = 10

$$Speedup_{overall} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

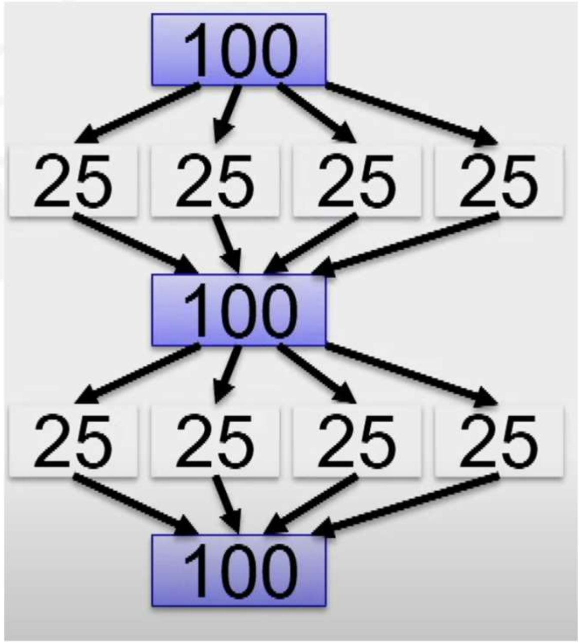
Amdahl’s Law - Illustration



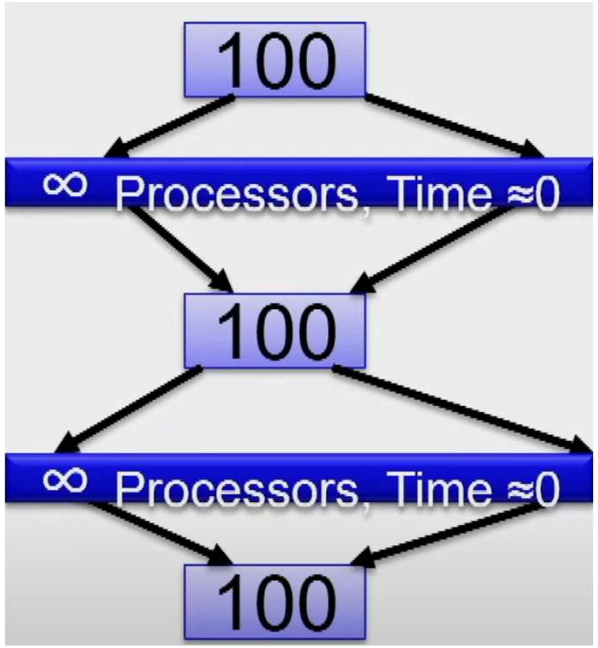
Work 500,
Time 500,
S_p=1X



Work 500,
Time 400,
S_p=1.25X



Work 500,
Time 350,
S_p=1.4X

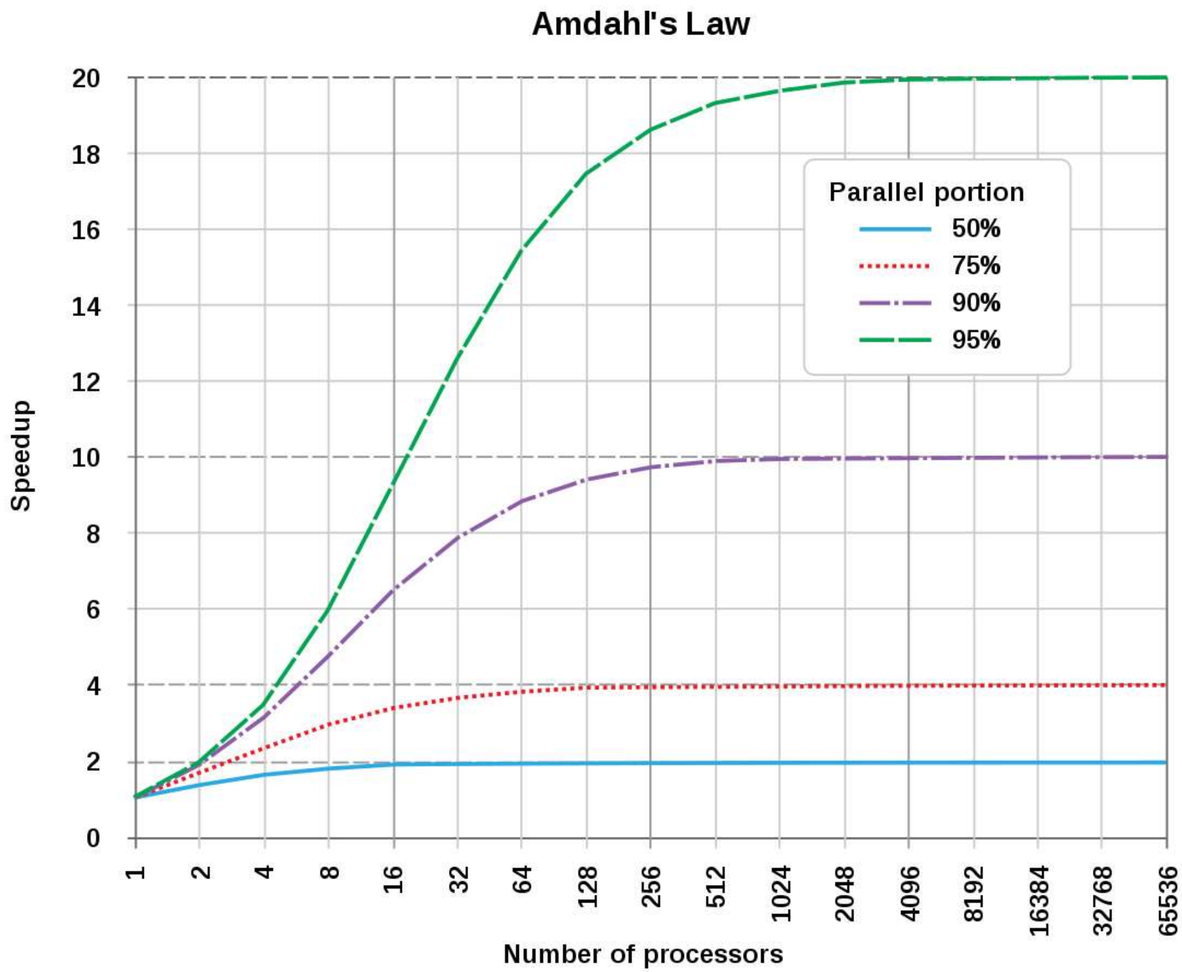


Work 500,
Time 300,
S_p=1.7X

How much Speedup you can achieve?

Amdahl's Law:

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



Design Example

A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark.

One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10. The other alternative is just to try to 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. Compare these two design alternatives using Amdahl's Law.

Design Example

Case A: FPSQR hardware optimization

$$S = \frac{1}{(1-f)+\frac{f}{N}}$$

$$= \frac{1}{(1-0.2)+\frac{0.2}{10}}$$

$$= \frac{1}{0.8+0.02} = 1.219\text{times}$$

Case B: FP instruction optimization

$$S = \frac{1}{(1-f)+\frac{f}{N}}$$

$$= \frac{1}{(1-0.5)+\frac{0.5}{16}}$$

$$= \frac{1}{0.5+0.3125} = 1.23\text{times}$$

Principles of Computer Design

- All processors are driven by clock
- Expressed as clock rate in GHz or clock period in ns
- CPU Time = CPU clock cycles × clock cycle time

$$CPI = \frac{CPU\ clock\ cycles\ for\ a\ program}{Instruction\ Count}$$

$$CPU\ Time = IC \times CPI \times CCT$$

$$CPU\ Time = \frac{Instructions}{Program} \times \frac{Clock\ Cycles}{Instructions} \times \frac{Seconds}{Clock\ Cycle}$$

Principles of Computer Design

$$CPU\ Time = IC \times CPI \times CCT$$

$$CPU\ Time = \frac{Instructions}{Program} \times \frac{Clock\ Cycles}{Instructions} \times \frac{Seconds}{Clock\ Cycle}$$

- Clock cycle time - hardware technology
- CPI – organization and ISA
- IC - ISA and compiler technology

Principles of Computer Design

- Different instruction types having different CPIs

$$CPU\ clock\ cycles = \sum_{i=1}^{n} IC_i \times CPIi$$

$$CPU\ Time = \left(\sum_{i=1}^{n} IC_i \times CPIi\right) \times CCT$$

Example: Basic Performance Analysis

Consider two programs A and B that solve a given problem. A is scheduled to run on a processor P1 operating at 1 GHz, and B is scheduled to run on processor P2 running at 1.4 GHz. A has total 10000 instructions, out of which 20% are branch instructions, 40% load store instructions and rest are ALU instructions. B is composed of 25% branch instructions. The number of load store instructions in B is twice the count of ALU instructions. Total instruction count of B is 12000. In both P1 and P2 branch instructions have an average CPI of 5 and ALU instructions has an average CPI of 1.5. Both the architectures differ in CPI of load-store instruction. They are 2 and 3 for P1 and P2, respectively. Which mapping (A on P1 or B on P2) solves the problem faster, and by how much?

Example: Basic Performance Analysis

A on P1 (1GHz → CCT = 1ns)	B on P2 (1.4 GHz→ CCT = 0.714ns)
IC=10000	IC=12000
Fraction BR: L/S: ALU = 20: 40: 40	Fraction BR: L/S: ALU = 25: 50: 25
CPI of BR: L/S: ALU = 5: 2: 1.5	CPI of BR: L/S: ALU = 5: 3 : 1.5

(a) $CPI\ A_{P1} = (0.2 \times 5 + 0.4 \times 2 + 0.4 \times 1.5) = 2.4$

$$ExT = 2.4 \times 10000 \times 1ns = 24000ns$$

(b) $CPI\ B_{P2} = (0.25 \times 5 + 0.5 \times 3 + 0.25 \times 1.5) = 3.125$

$$ExT = 3.125 \times 12000 \times 0.714ns = 26775ns$$

So, A on P1 is faster.

Example: Amdahl's Law

A company is releasing 2 latest versions (beta and gamma) of its basic processor architecture named alpha. Beta and gamma are designed by making modifications on three major components (X, Y and Z) of the alpha. It was observed that for a program A the fractions of the total execution time on these three components, X, Y, and Z are 40%, 30% and 20%, respectively. Beta speeds up X and Z by 2 times but slows down Y by 1.3 times, whereas gamma speeds up X, Y and Z by 1.2, 1.3 and 1.4 times, respectively.

(1) How much faster is gamma over alpha for running A?

(2) Whether beta or gamma is faster for running A? Find the speedup factor.

$$S = \frac{1}{(1-f_x-f_y-f_z)+\left(\frac{f_x}{N_x}+\frac{f_y}{N_y}+\frac{f_z}{N_z}\right)}$$

f_x=0.4, $f_y = 0.3$, $f_z = 0.2$

Beta: N_x=2, $N_y = \frac{1}{1.3}$, $N_z = 2$

Gamma: N_x=1.2, $N_y = 1.3$, $N_z = 1.4$

Example: Amdahl's Law

$$S = \frac{1}{(1-f_x-f_y-f_z)+\left(\frac{f_x}{N_x}+\frac{f_y}{N_y}+\frac{f_z}{N_z}\right)}$$

f_x=0.4, $f_y = 0.3$, $f_z = 0.2$

Beta: N_x=2, $N_y = \frac{1}{1.3}$, $N_z = 2$

Gamma: N_x=1.2, $N_y = 1.3$, $N_z = 1.4$

$$S_{\text{beta/alpha}} = \frac{1}{(0.1)+\left(\frac{0.4}{2}+\frac{0.3}{0.77}+\frac{0.2}{2}\right)} = 1.267\text{times}$$

$$S_{\text{alpha/}}\text{beta} = \frac{1}{(0.1)+\left(\frac{0.4}{1.2}+\frac{0.3}{1.3}+\frac{0.2}{1.4}\right)} = 1.239\text{times}$$

(1) Gamma is 1.239 times faster over alpha.

(2) Beta is faster than gamma, 1.267/1.239 = 1.022 times

Summary

Suggested Readings

Computer Architecture: A Quantitative Approach (5th Edition) by John Hennessy and David A Patterson:

- pp. 17 – 33: Sections 1.4 - 1.6 Trends in Technology, Power & Energy, and Cost
- pp. 36 – 52: Sections 1.8 & 1.9 Measuring Performance & Quantitative Principles