

Advanced Computer Architecture

Lecture 2

Performance Evaluation Methods



NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

Dr. John Jose

Assistant Professor

**Department of Computer Science & Engineering
Indian Institute of Technology Guwahati, Assam.**

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)

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

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

- ❖ $\text{Execution time}_Y / \text{Execution time}_X = n$

- ❖ $\text{Throughput}_X / \text{Throughput}_Y = n$

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

CINT2006 (Integer)

12 programs

Benchmark

400.perlbench
401.bzip2
403.gcc
429.mcf
445.gobmk
456.hmmer
458.sjeng
462.libquantum
464.h264ref
471.omnetpp
473.astar
483.Xalanbmk
410.bwaves
416.gamess
433.milc
434.zeusmp
435.gromacs
436.cactusADM
437.leslie3d
444.namd
447.dealII
450.soplex
453.povray
454.calculix
459.GemsFDTD
465.tonto
470.lbm
481.wrf
482.sphinx3

Language

C
C
C
C
C
C
C
C
C
C++
C++
C++
Fortran
Fortran
C
Fortran
C/Fortran
C/Fortran
Fortran
C++
C++
C++
C++
C/Fortran
Fortran
Fortran
C
C/Fortran
C

Descriptions

PERL Programming Language
Compression
C Compiler
Combinatorial Optimization
Artificial Intelligence: go
Search Gene Sequence
Artificial Intelligence: chess
Physics: Quantum Computing
Video Compression
Discrete Event Simulation
Path-finding Algorithms
XML Processing
Fluid Dynamics
Quantum Chemistry
Physics: Quantum Chromodynamics
Physics/CFD
Biochemistry/Molecular Dynamics
Physics/General Relativity
Fluid Dynamics
Biology/Molecular Dynamics
Finite Element Analysis
Linear Programming, Optimization
Image Ray-tracing
Structural Mechanics
Computational Electromagnetics
Quantum Chemistry
Fluid Dynamics
Weather Prediction
Speech recognition

CFP2006 (Floating Point)

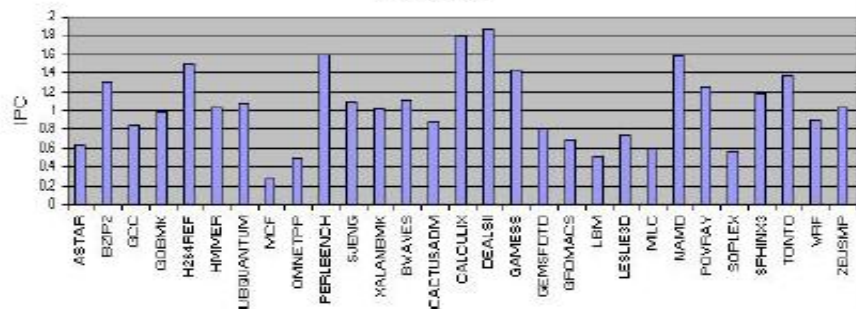
17 programs

NPTEL Online Course: Advanced Computer Architecture - Dr. John Jose, IIT Guwahati

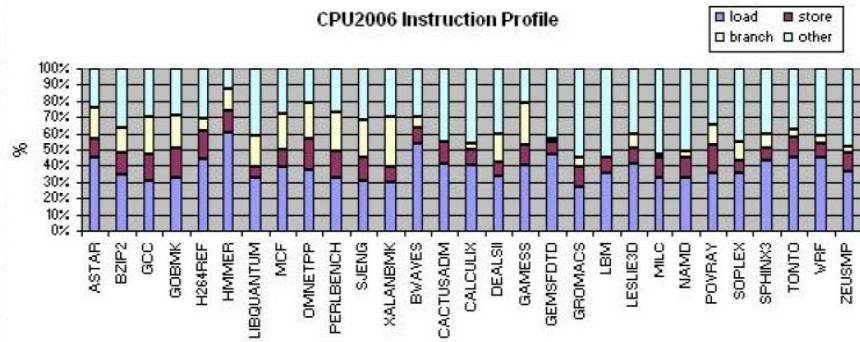
Source: <http://www.spec.org/cpu2006/>

Benchmark Based Evaluation

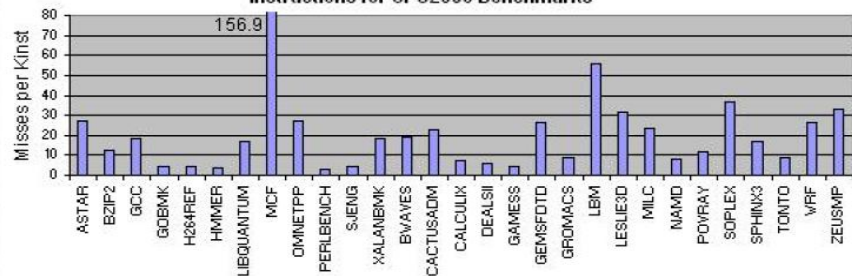
CPU2006 IPC



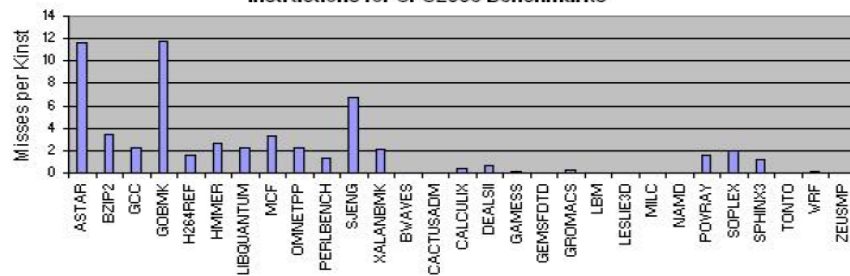
CPU2006 Instruction Profile



L1 Data Cache Misses per 1000 Instructions for CPU2006 Benchmarks



Branch Mispredictions per 1000 Instructions for CPU2006 Benchmarks



SPEC Ratio

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

Reference for SPEC 2006:

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

NPTFL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

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

$$\text{GeometricMean}(a_1, a_2, a_3, \dots, a_N) = \sqrt[N]{\prod_i a_i}$$

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.

$$ET_{\text{new}} = ET_{\text{old}} \times (1 - \text{fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}$$
$$\text{Speedup} = \frac{ET_{\text{old}}}{ET_{\text{new}}} = \frac{1}{(1 - \text{fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Amdahl's Law- Illustration

Example: Suppose that we want to enhance the floating point operations 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?

Solution:

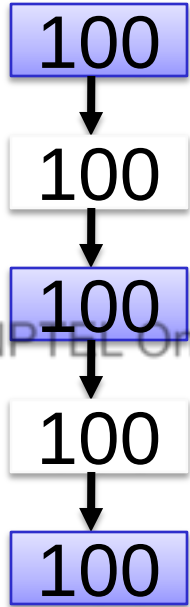
Fraction enhanced = 0.4

Speedup enhanced = 10

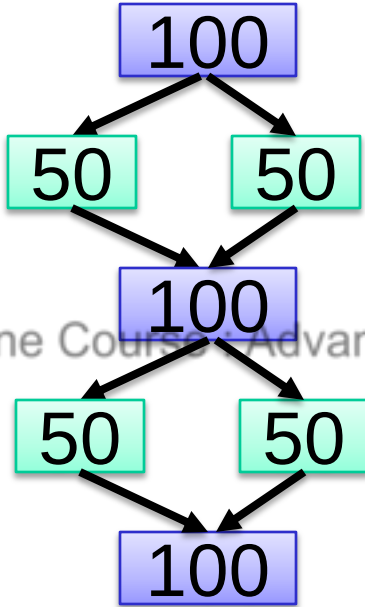
$$\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

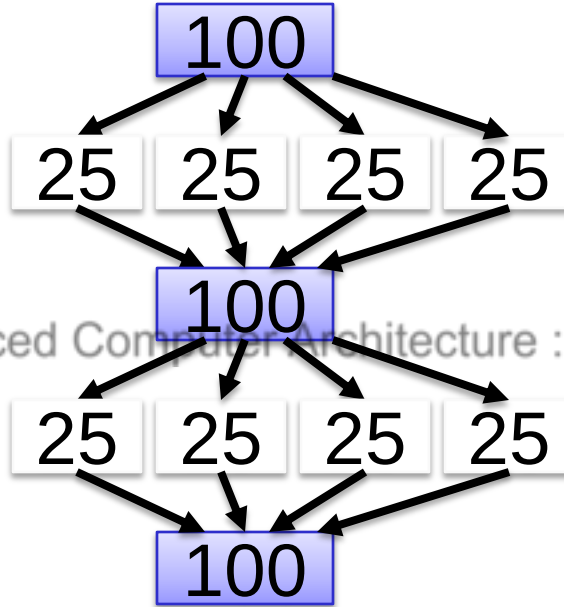
Amdahl's Law for Parallel Processing



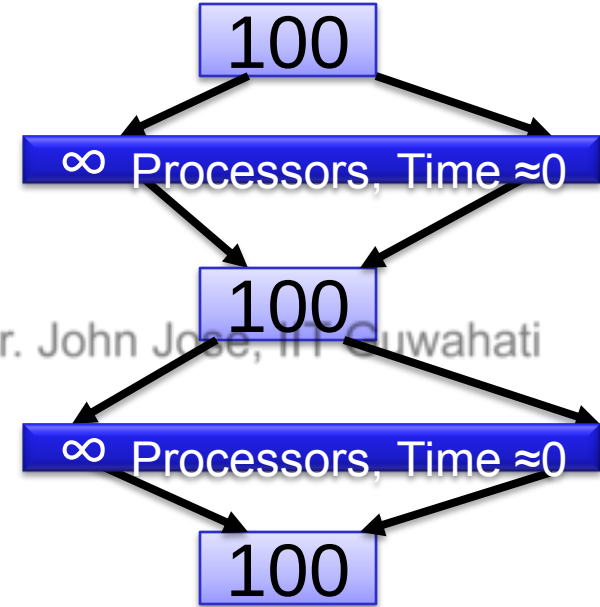
Work 500,
Time 500
Sp=1X



Work 500,
Time 400
Sp=1.25X



Work 500,
Time 350
Sp=1.4X

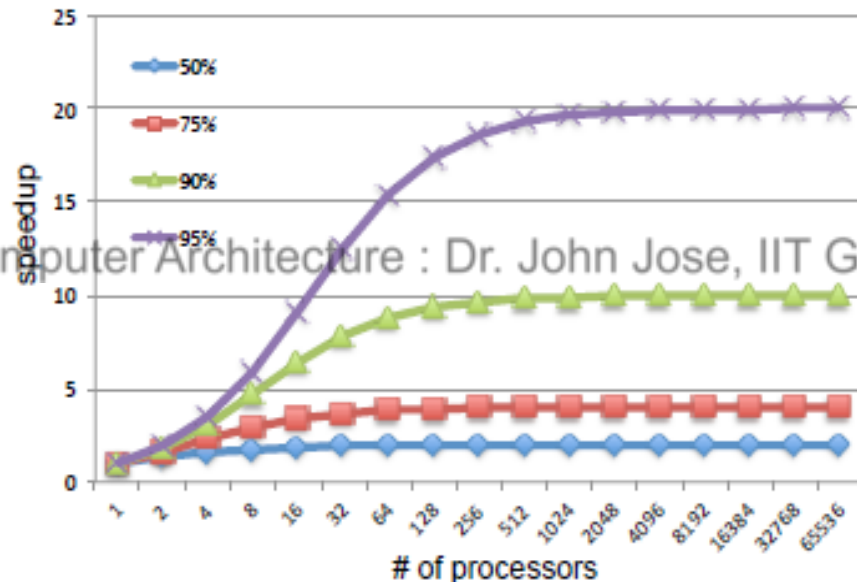


Work 500,
Time 300
Sp=1.7X

How much Speed up you can achieve ?

Amdahl's Law:

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



NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

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

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

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

$$\frac{1}{(1-0.2) + \frac{0.2}{10}} = \frac{1}{(0.8) + 0.02}$$

= 1.219 times.

Case B: FP instructions optimization

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

$$\frac{1}{(1-0.5) + \frac{0.5}{1.6}} = \frac{1}{(0.5) + 0.3125}$$

= 1.23 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 x clock cycle time

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

$$\text{CPU Time} = \text{IC} \times \text{CPI} \times \text{CCT}$$

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

Principles of Computer Design

$$\text{CPU Time} = \text{IC} \times \text{CPI} \times \text{CCT}$$

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

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

Principles of Computer Design

- ❖ Different instruction types having different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^n \text{IC}_i \times \text{CPI}_i$$

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

$$\text{CPU time} = \left(\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i \right) \times \text{CCT}$$

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 have an average CPI of 1.5. Both the architectures differ in the 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) $CPI_{A_P1} = (0.2 \times 5 + 0.4 \times 2 + 0.4 \times 1.5) = 2.4$

$ExT = 2.4 \times 10000 \times 1 \text{ ns} = 24000 \text{ ns}$

(b) $CPI_{B_P2} = (0.25 \times 5 + 0.5 \times 3 + 0.25 \times 1.5) = 3.125$

$ExT = 3.125 \times 12000 \times 0.714 \text{ ns} = 26775 \text{ ns}$

Hence 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, where as gamma speeds up X, Y and Z by 1.2, 1.3 and 1.4 times, respectively.

- (a) How much faster is gamma over alpha for running A?
- (b) Whether beta or gamma is faster for running A? Find the speedup factor

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

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

$$\text{Beta: } N_x=2 : N_y=1/1.3 : N_z=2$$

$$\text{Gamma: } N_x=1.2 : N_y=1.3 : N_z=1.4$$

Example: Amdahl's Law

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

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

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

(a) Gamma is 1.239 times faster over alpha

(b) Beta is faster than gamma $\rightarrow 1.267/1.239 = 1.022$ times

**THANK
YOU!**

FOR

YOUR

PRESENCE

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

johnjose@iitg.ac.in

<http://www.iitg.ac.in/johnjose/>