CS 223 Computer Organization & Architecture

Lecture 23 [12.03.2020]

Performance Measurement Techniques

John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Measuring Performance

- **Typical performance metrics:**
  - Response time
  - Throughput
- **Speedup of X relative to Y**
  - Execution time$_Y$ / Execution time$_X$
- **Execution time**
  - Wall clock time: includes all system overheads
  - CPU time: only computation time
- **Benchmarks**
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06, EEMBC, TPC-C)

Benchmark Suite

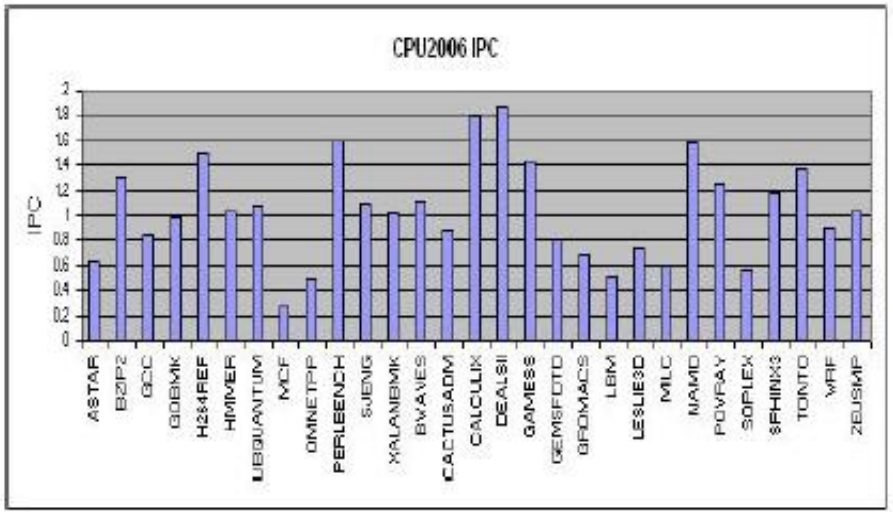
SPEC CPU2006 Programs

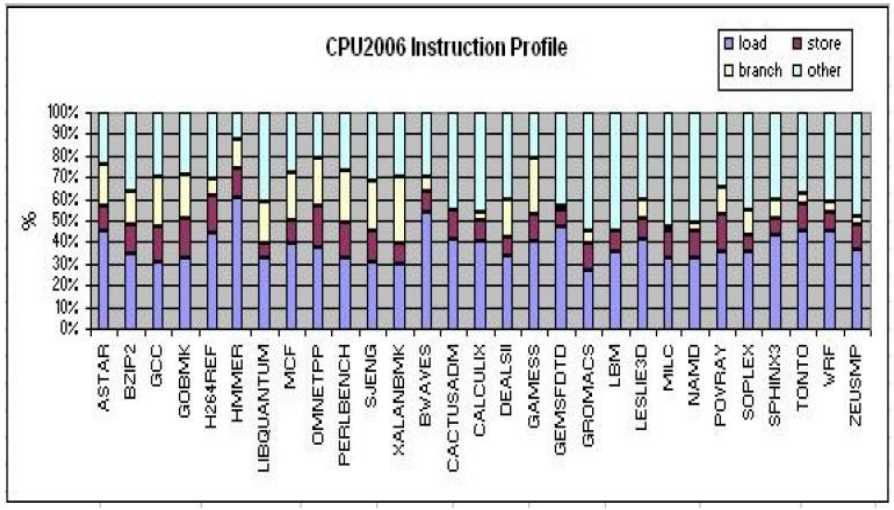
	Benchmark	Language	Descriptions
CINT2006 (Integer) 12 programs	400.perlbench	C	PERL Programming Language
	401.bzip2	C	Compression
	403.gcc	C	C Compiler
	429.mcf	C	Combinatorial Optimization
	445.gobmk	C	Artificial Intelligence: go
	456.hmmer	C	Search Gene Sequence
	458.sjeng	C	Artificial Intelligence: 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

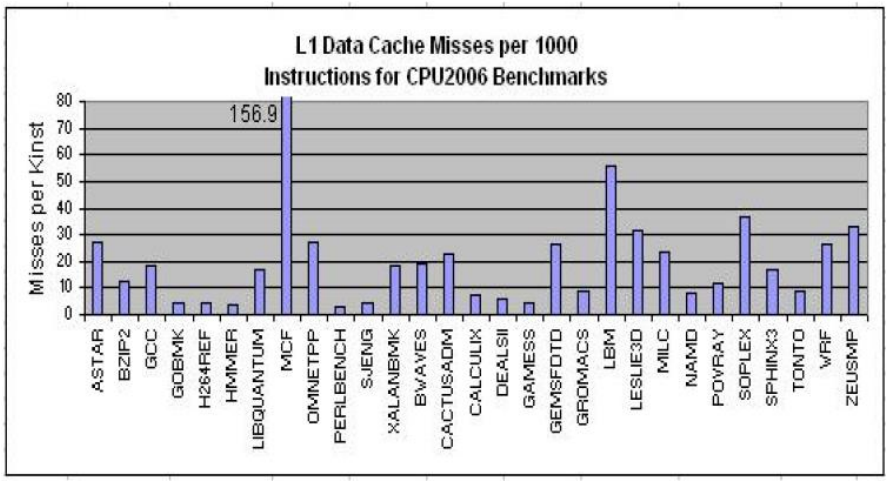
Target Programs application domain: Engineering and scientific computation

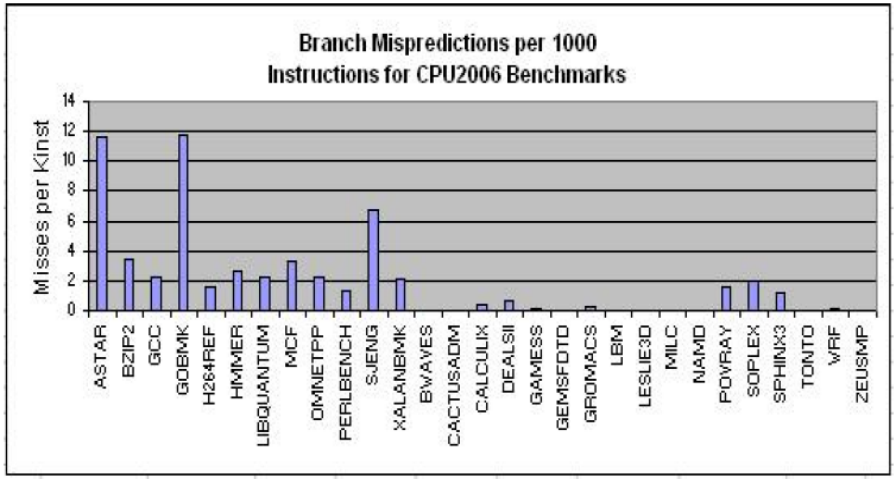
Source: http://www.spec.org/cpu2006/

Benchmark based evaluation









SPEC Ratio

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

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

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

Principles of Computer Design

- All processors are driven by clock.
- Expressed as clock rate in GHz or clock period in ns

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

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

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

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

Principles of Computer Design

- Different instruction types having different CPIs

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

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

$$\text{CPI} = \frac{\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i}{\text{Instruction count}} = \sum_{i=1}^{n} \frac{\text{IC}_i}{\text{Instruction count}} \times \text{CPI}_i$$

Example: Basic Performance Analysis

Consider two programs A and B that solves 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 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?

A on P1 (1GHz → 1ns)	B on P2 (1.4 GHz→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

Example: Basic Performance Analysis

A on P1 (1GHz → 1ns)	B on P2 (1.4 GHz→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=(0.2x5 + 0.4x2 + 0.4x1.5) = 2.4

ExT = 2.4 x10000x1= 24000 ns

(a) (b) CPI B_P2=(0.25x5 + 0.5x3 + 0.25x1.5) = 3.125

ExT = 3.125 x12000x0.714= 26775 ns

Hence A on P1 is faster.

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 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:

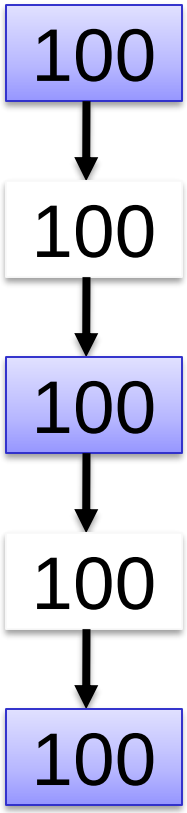
Fraction enhanced = 0.4

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

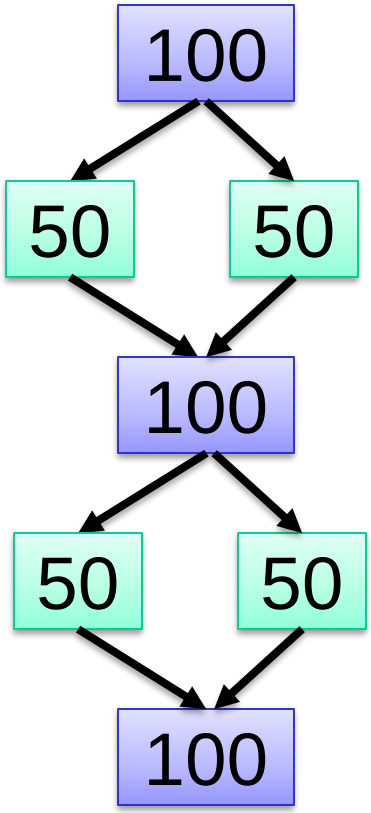
Speedup enhanced = 10

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

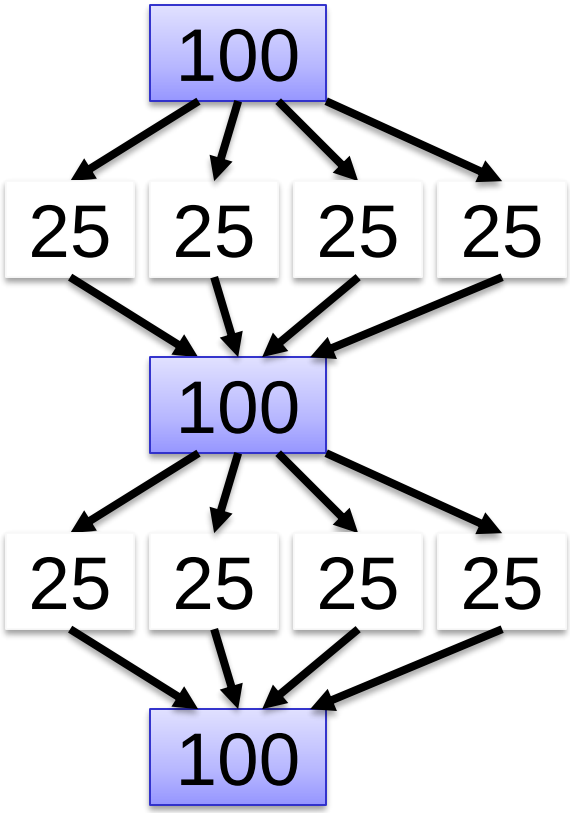
Amdahl's Law for multi-cores



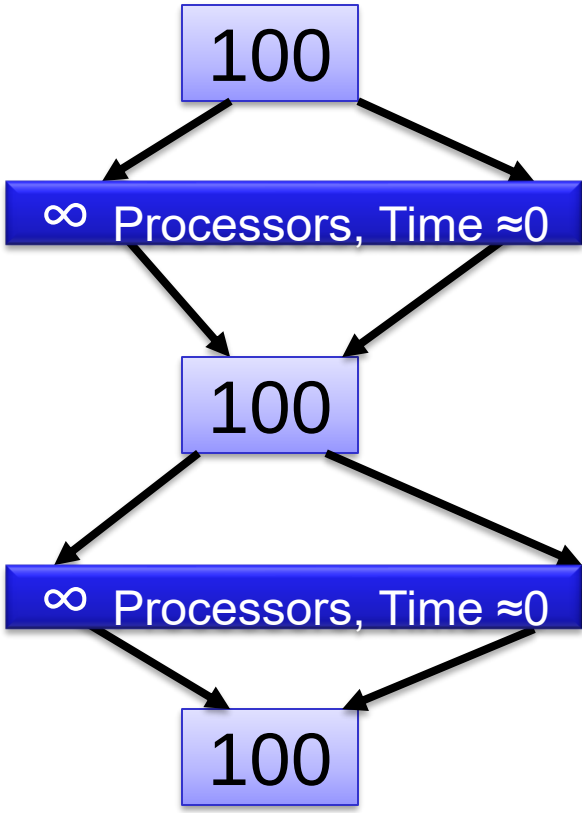
Work 500,
Time 500
Sp=1X



Work 500,
Time 400
Sp=1.25X



Work 500,
Time 350
Sp=1.4X

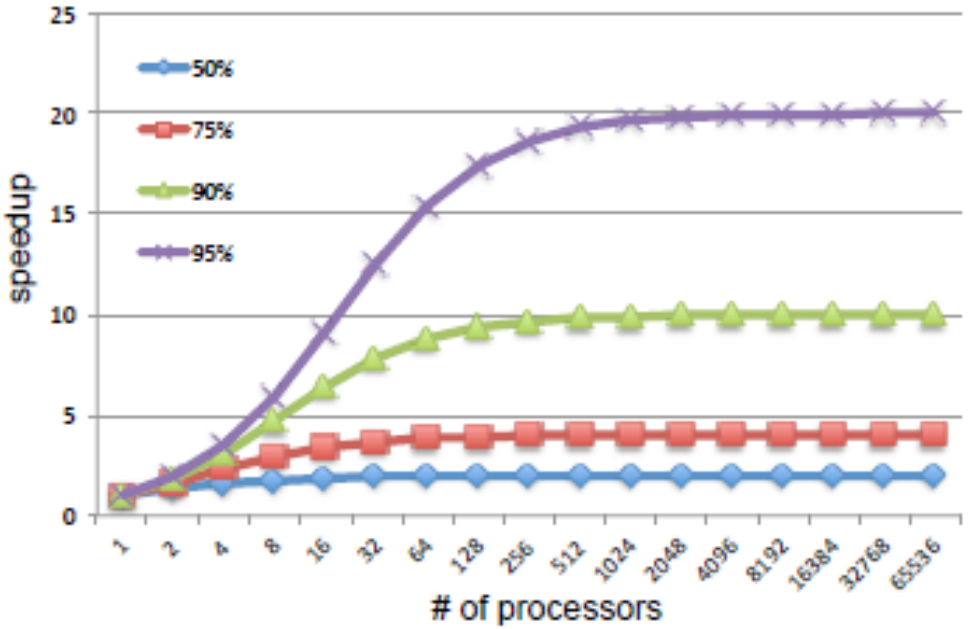


Work 500,
Time 300
Sp=1.7X

How much Speed up you can achieve ?

Amdahl's Law:

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



Example: Amdahl's Law

A new floating-point unit speeds up floating point operations by two times. In an application one fifth of the instructions are floating-point operations.

(a) What is the overall speedup? (Ignore the penalty to other instructions).

(b) Assume that the speeding up of the floating-point unit mentioned above slowed down data cache accesses resulting in a 1.5x slowdown. Assume the load instructions constitute 15% and store instructions constitute 9% of the total instruction what is the effective overall speedup now?

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

(a) S = 1/ { (1-f) + (f/N) } = 1 / { (1- 0.2) + (0.2/2) } = 1.11 times

(b) S = 1/ { (1-f1-f2) + (f1/N1) + (f2/N2) }

= 1 / { (1- 0.2-0.24) + (0.2/2) + (0.24/0.67) }

= 0.98 times

Reading Exercises

- **Computer Architecture-A Quantitative Approach** (5th edition),

  John L. Hennessy, David A. Patterson, Morgan Kaufman.

  - Chapter 1.8 – Measuring, Reporting and summarizing Performance
  - Chapter 1.9 – Quantitative Principles of Computer Design

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/