Computer Architecture

Performance Evaluation Methods

Dr. Mohammad Reza Selim

Lecture Outline

- Typical Performance Metrics
- Benchmarks
- Amdahl's Law
- Principle of Computer Design

Computer Architecture

- The architecture of a system refers to its structure in terms of separately specified components of that system and their interrelationships.
- Computer architecture is **a set of rules and methods that describe the functionality, organization, and implementation of computer systems.**

Computer Architecture

- **“Old” view of computer architecture:**
  – Instruction Set Architecture (ISA) design
  – i.e. decisions regarding: registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- **“Real” computer architecture:**
  - Given specific requirements of the target machine
  - Design goal is to maximize performance within constraints: cost, power, and availability
  – Computer Architecture covers all three aspects of computer design—instruction set architecture, organization or micro-architecture, and hardware.
- AMD Opteron and the Intel Core i7, both processors implement the x86 instruction set (same ISA), but they have very different pipeline and cache organizations (different microarchitecture).
- ISA and micro architectures of the Intel Core i7 and the Intel Xeon 7560 are nearly identical but offer different clock rates and different memory systems, making the Xeon 7560 more effective for server computers (hardware implementation is different).

Typical Performance Metrics

- **Response Time / Latency** - time between the start and the completion of an event, such as milliseconds for a disk access.
- **Throughput / Bandwidth** - total amount of work done in a given time, such as megabytes per second for a disk transfer. In
- **CPU Time** – Time to execute a program without I/O time. Only computation time.
- **Wall Clock Time** - Time to execute a program including I/O time.
- **Speedup** – Execution time(old) / Execution time (new)

Typical Performance Metrics

When Can we say that one computer/ architecture/ design is better than other?

- For PC/Laptop: Execution time of a program
- For Servers: Transactions/unit time called Throughput

Typical Performance Metrics

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

- Execution time$_Y$ / Execution time$_X$ =n
- Throughput $_x$/ Throughput $_y$ =n

Benchmarks

Standard Programs used for comparing two machines/architecture.
Types:

- **Kernels**, which are small, key pieces of real applications
- **Toy programs,** which are 100-line programs from beginning programming assignments, such as quicksort
- **Synthetic benchmarks,** which are fake programs invented to try to match the profile and behavior of real applications, such as Dhrystone
- **Benchmark suites,** are a popular measure of performance of processors with a variety of applications, such as SPEC06, SPLASH, etc.

Benchmarks Suite

SPEC CPU2006 Programs

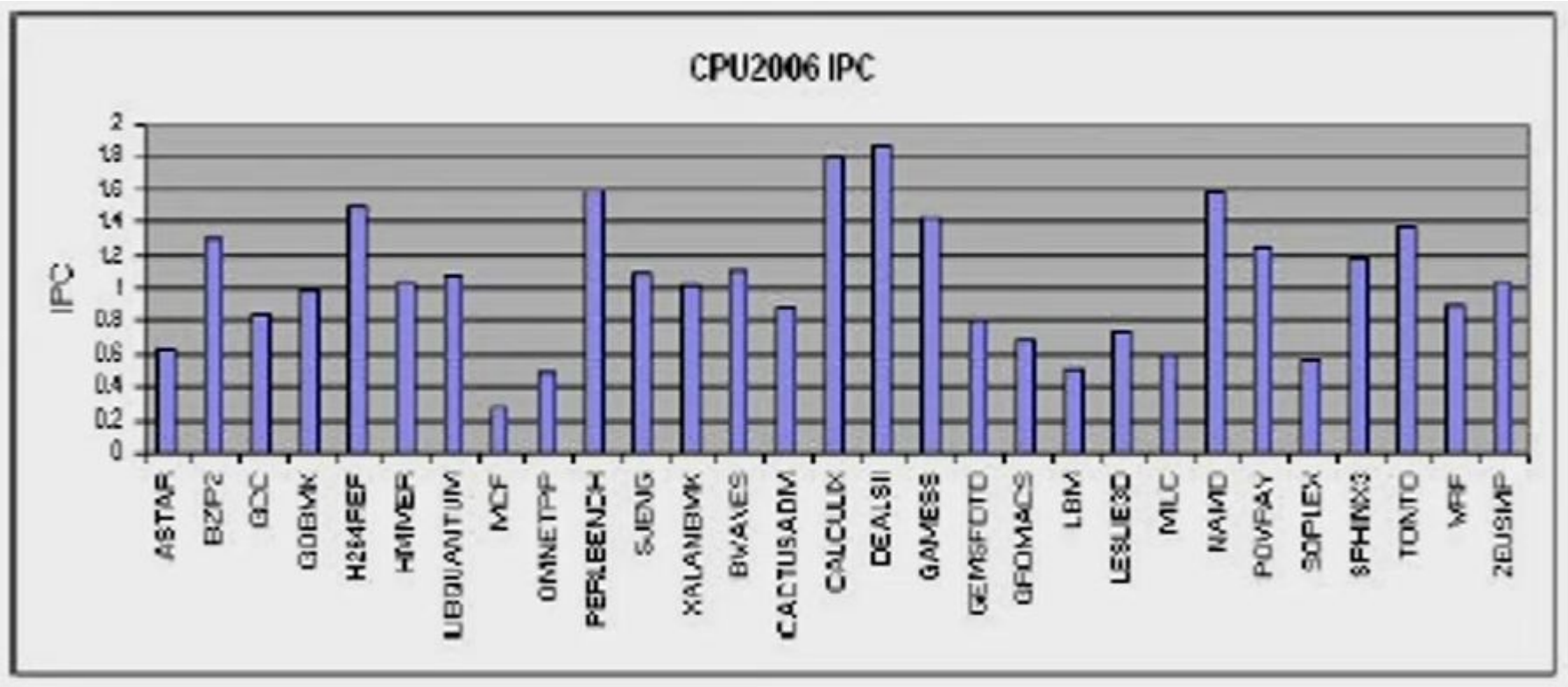
	Benchmark	Language	Descriptions
CINT2006 (Integer) 12 programs	400.perlbench	C	PERL Programming Language
	401.bzip2	C	Compression
	403.gcc	C	C Compiler
	429.mcf	C	Combinatorial Optimization
	445.gobmk	C	Artificial Intelligence: go
	456.hmmer	C	Search Gene Sequence
	458.sjeng	C	Artificial Intelligence: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

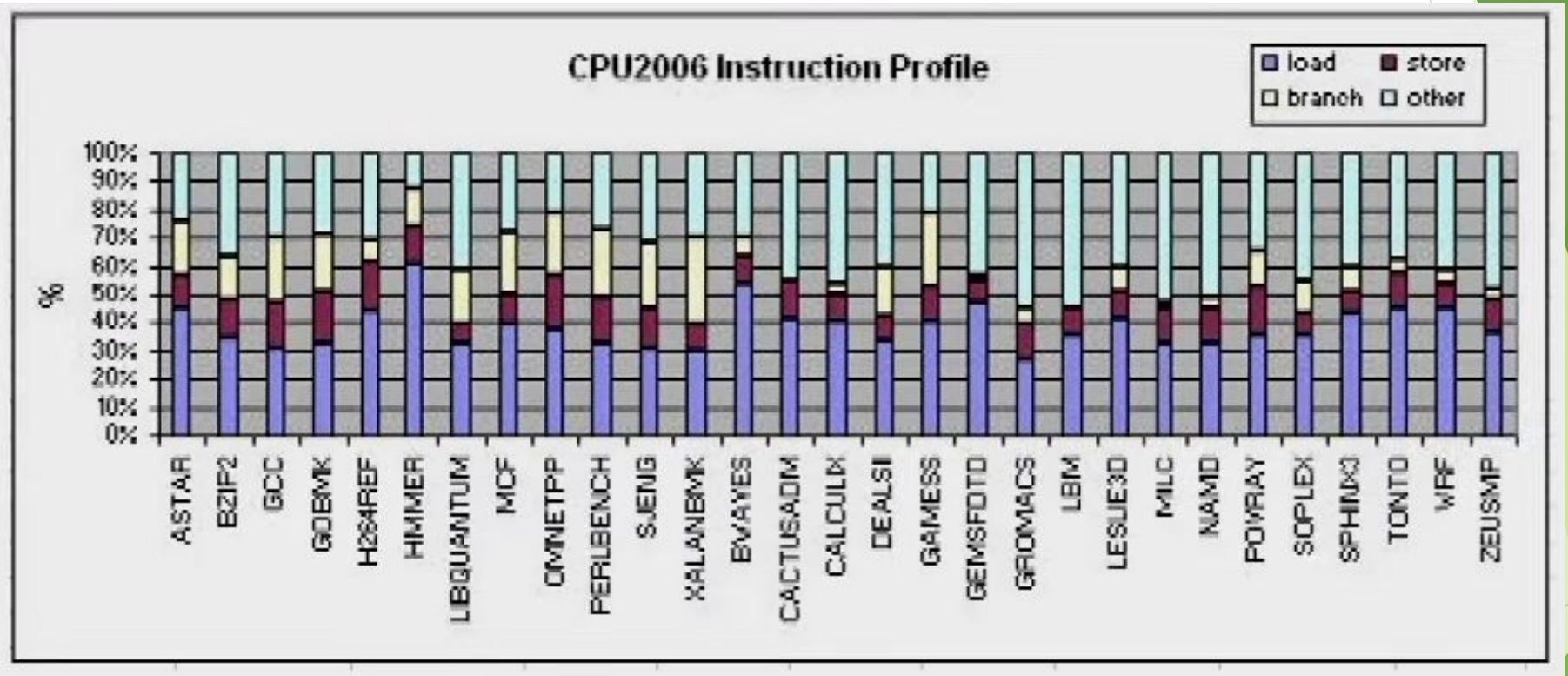
Architecture Simulators

- Used to design system-level architecture as well as processor microarchitecture. Can be created computer architectural designs and modified as required. Designs can be analyzed.
- Example: gem5
- gem5 is an open source computer architecture simulator used in academia and in industry.

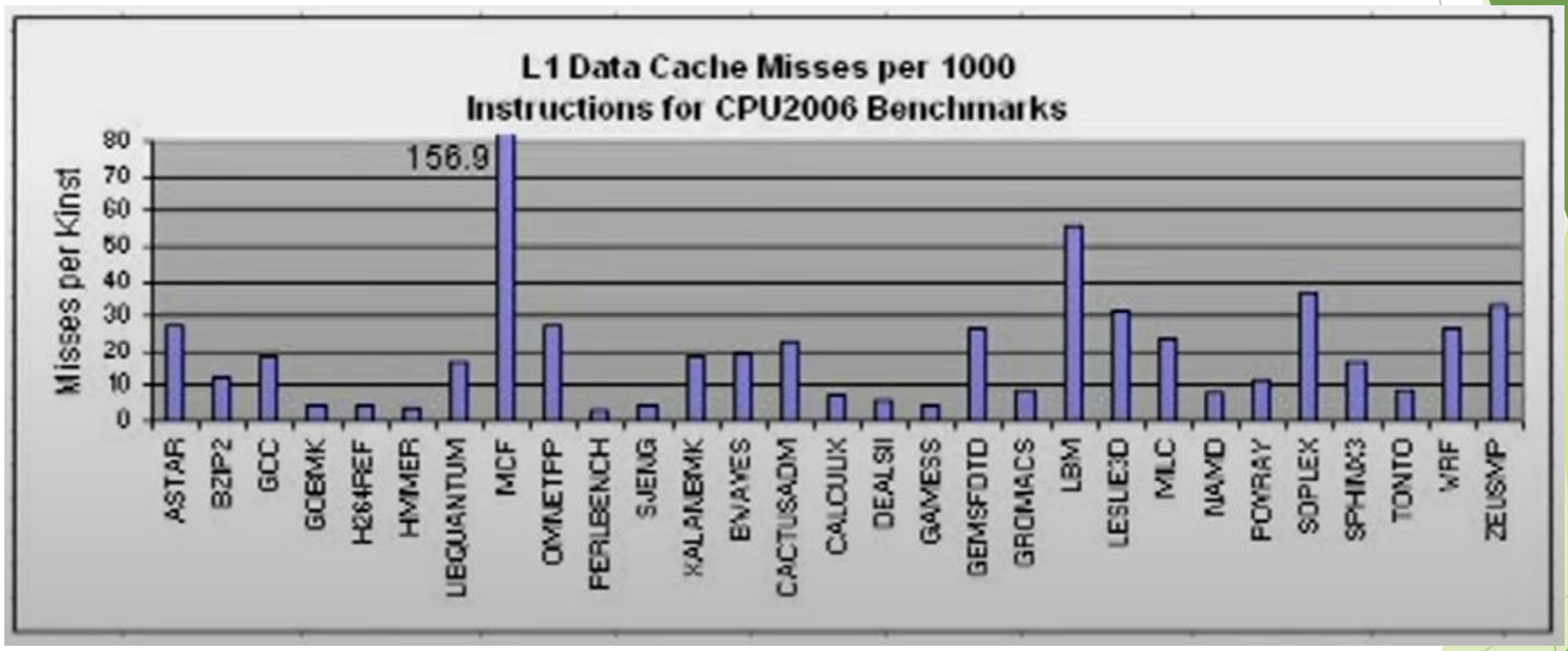
Benchmarks based Evaluation



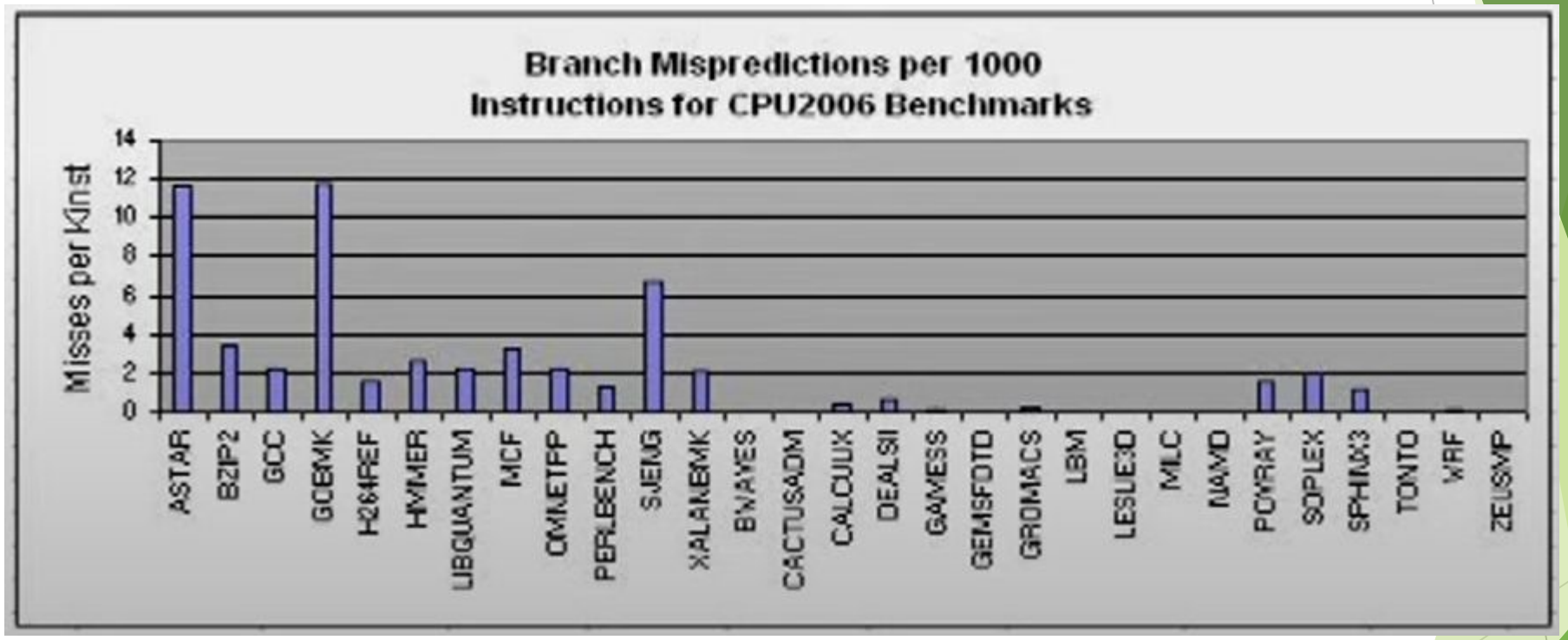
Benchmarks based Evaluation



Benchmarks based Evaluation



Benchmarks based Evaluation



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

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

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

Amdahl's Law

- One of the principle of computer design is to ***make the common case faster***
- Amdahl's law states that 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.

Amdahl's law defines the *speedup* that can be gained by using a particular feature. What is speedup? Suppose that we can make an enhancement to a computer that will improve performance when it is used. Speedup is the ratio:

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively,

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

Speedup tells us how much faster a task will run using the computer with the enhancement as opposed to the original computer.

Amdahl's Law

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

The overall speedup is the ratio of the execution times:

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

Amdahl's Law

Example Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. 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 speedup gained by incorporating the enhancement?

Answer $\text{Fraction}_{\text{enhanced}} = 0.4$; $\text{Speedup}_{\text{enhanced}} = 10$; $\text{Speedup}_{\text{overall}} = \dfrac{1}{0.6 + \dfrac{0.4}{10}} = \dfrac{1}{0.64} \approx 1.56$

Amdahl's Law

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. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Amdahl's Law

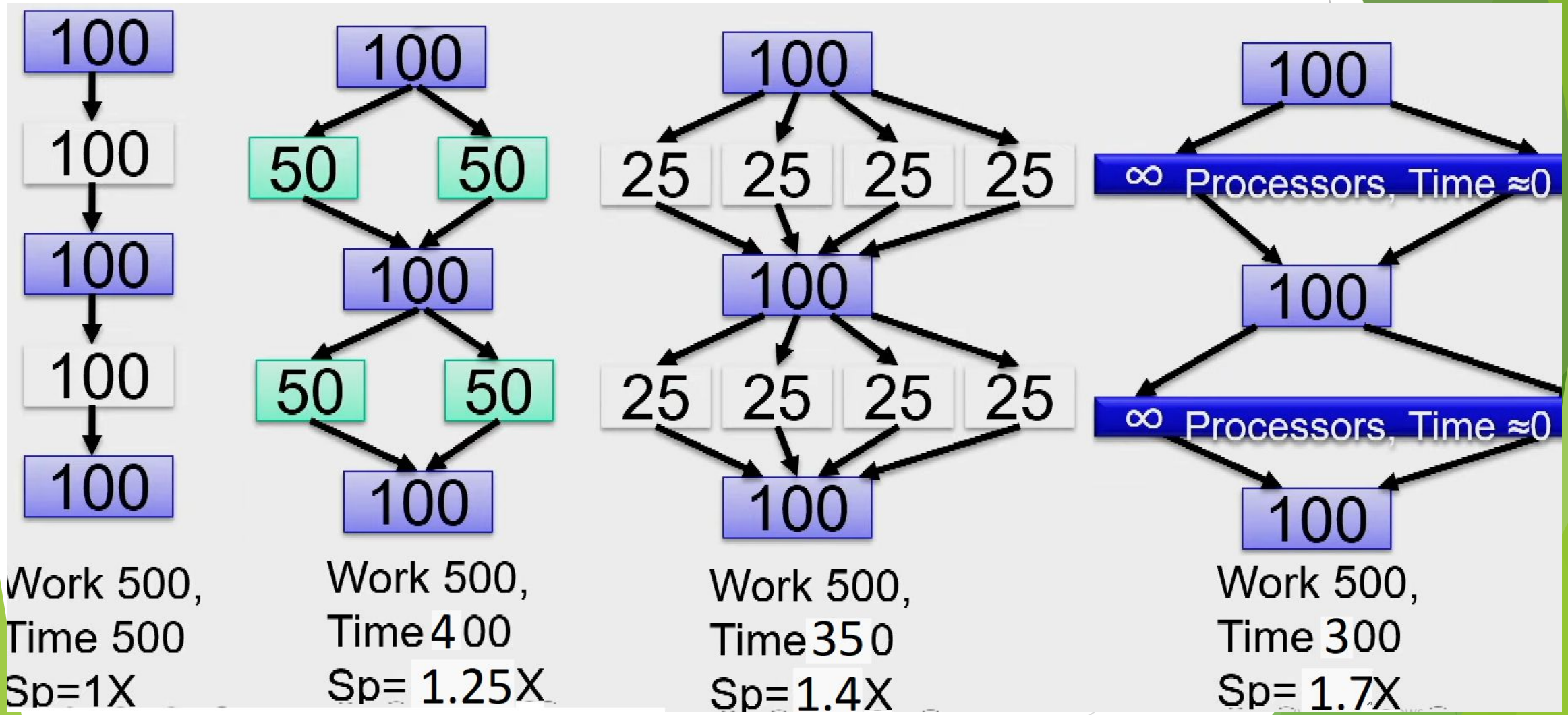
Answer We can compare these two alternatives by comparing the speedups:

$$\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

Improving the performance of the FP operations overall is slightly better because of the higher frequency.

Amdahl's Law for parallel processing

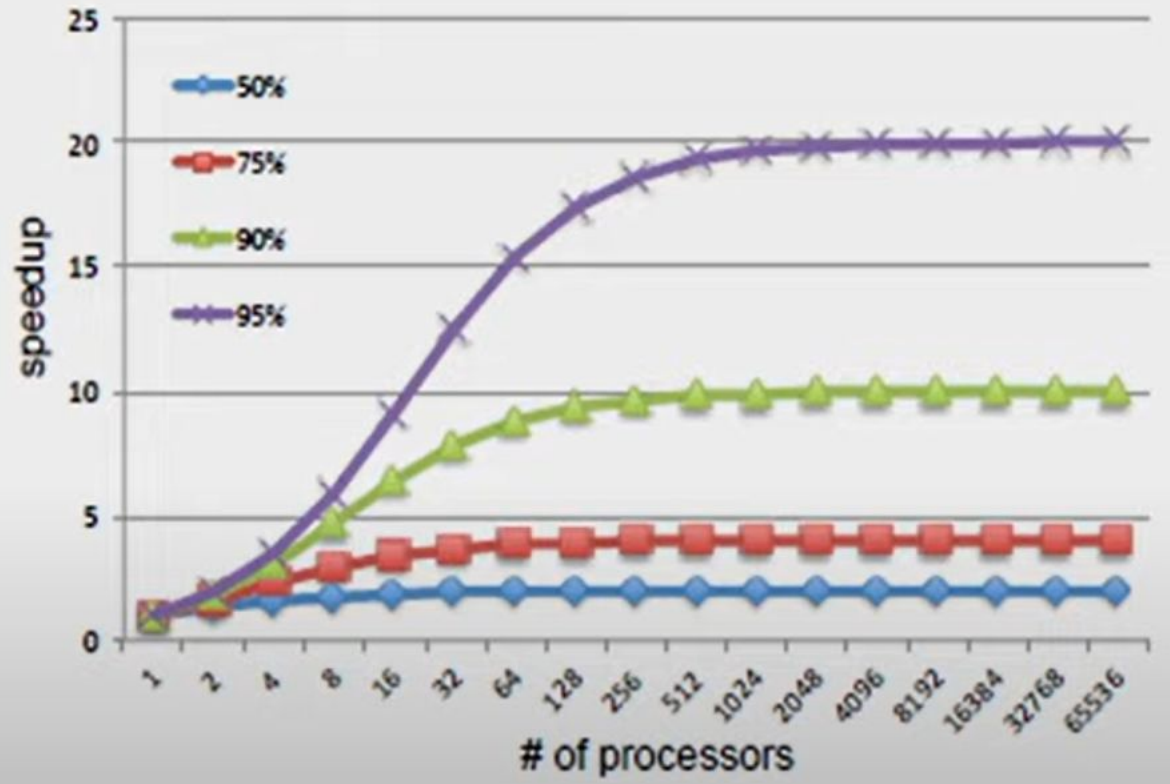


Amdahl's Law for parallel processing

How much speedup you can get from parallel processing?

Amdahl's Law:

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



Principles of Computer Design

- Take Advantage of Parallelism
  - e.g. multiple processors, disks, memory banks, pipelining, multiple functional units
- Make use of Principle of Locality
  - Reuse of data and instructions
  - Rules of thumb: 10/90, 20/80
  - Temporal vs. Spatial
- Focus on the Common Case - favour the frequent case over the infrequent case.

Principle 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

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

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

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

Principle 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

Principle of Computer Design

- Different instruction types having different CPIs

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

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

Principle of Computer Design

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?

Principle of Computer Design

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=(0.2x5 + 0.4x2 + 0.4x1.5) = 2.4

ExT = 2.4 x10000x1 ns= **24000 ns**

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

ExT = 3.125 x12000x0.714 ns = **26775 ns**