TEXAS TECH UNIVERSITY
Department *of* Computer Science

CS5375 Computer Systems Organization and Architecture

Lecture 4

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Review of Last Lecture

- Trends in Technology
  - IC, DRAM, flash, magnetic disk, etc. => gap between CPU/processor and data-access enlarging
  - Power Wall
  - Techniques for reducing power

- Measuring, Reporting and Summarizing Performance

Review of Last Lecture (cont.)

- Performance = 1/Execution Time
- Elapsed time (wall-clock time, response time) v.s. CPU execution time (CPU time)
- CPU execution time for a program can be written as:

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

Or: $$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Three key factors that affect performance
  - Instruction Count, CPI, Clock Cycle Time or Clock Rate

Outline

- Measuring, Reporting and Summarizing Performance (cont.)
- Amdahl's Law and Scaled Computing

Measuring Performance

- To evaluate and compare two computers, the workload matters (when comparing the execution time on two computers)

- Workload
  - A set of programs run on a computer that is either the actual collection of applications run by a user or constructed from real programs to approximate such a mix

- Benchmarks
  - A set of programs (standard, agreed upon) chosen specifically to measure performance
  - From a user's perspective, need to know which programs are commonly run, so that the right benchmarks can be chosen to predict the performance of the actual workload

Measuring Performance (cont.)

- Benchmarks
  - Kernels: small, key pieces of real applications, e.g. matrix multiply
  - Toy programs: small or tiny real programs, e.g. sorting
  - Synthetic benchmarks: “fake” programs invented to match the profile and behavior of real applications
  - Benchmark suites: a collection of benchmark applications, e.g. SPEC (Standard Performance Evaluation Corporation) CPU2017, TPC-C (Transaction-Processing Council)

SPEC CPU Benchmark

- SPEC CPU2017 benchmark suite
  - https://www.spec.org/benchmarks.html
  - A set of Integer (called SPECspeed® 2017 Integer) and Floating-Point (SPECspeed® 2017 Floating Point) benchmarks for comparing the execution time
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Where Execution time ratio$_i$ is the execution time, normalized to the reference computer, for the *i*-th program of a total of *n* in the workload, and

$$\prod_{i=1}^{n} a_i \text{ means the product } a_1 \times a_2 \times \ldots \times a_n$$

SPECspeed 2017 Integer benchmarks on a 1.8 GHz Intel Xeon E5-2650L

$$\text{SPECRatio} = \text{Execution Time}_{\text{reference}} / \text{Execution Time}_{\text{target}}$$

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean	–	–	–	–	–	–	2.36

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Outline

- Measuring, Reporting and Summarizing Performance (cont.)
- Amdahl's Law and Scaled Computing

A Pitfall and Amdahl's Law

- Pitfall: improving one aspect of a computer and expecting a proportional improvement in overall performance

$$T_{improved} = \frac{T_{affected}}{\text{improvement factor}} + T_{unaffected}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

Can't be done!

Example: Laundry Room

Dirty Laundry | Washing Machine | Drying Machine | Clean Laundry

30 minutes washing

90 minutes drying

Total Execution Time: 30+90 = 120 minutes
Washing Portion: 30/120 = ¼
Drying Portion: 90/120 = ¾

If we can have two drying machines

Dirty Laundry | Washing Machine | 2 Drying Machines | Clean Laundry




30 minutes washing

90/2=45 minutes drying

Speedup? (30+90)/(30+45)=1.6

If we can have unlimited drying machines

Dirty Laundry

Washing Machine

∞ Drying Machines

Clean Laundry

30 minutes washing

90/∞ ≈ 0 minutes drying

Speedup? (30+90)/(30+0)=4

Amdahl's Law

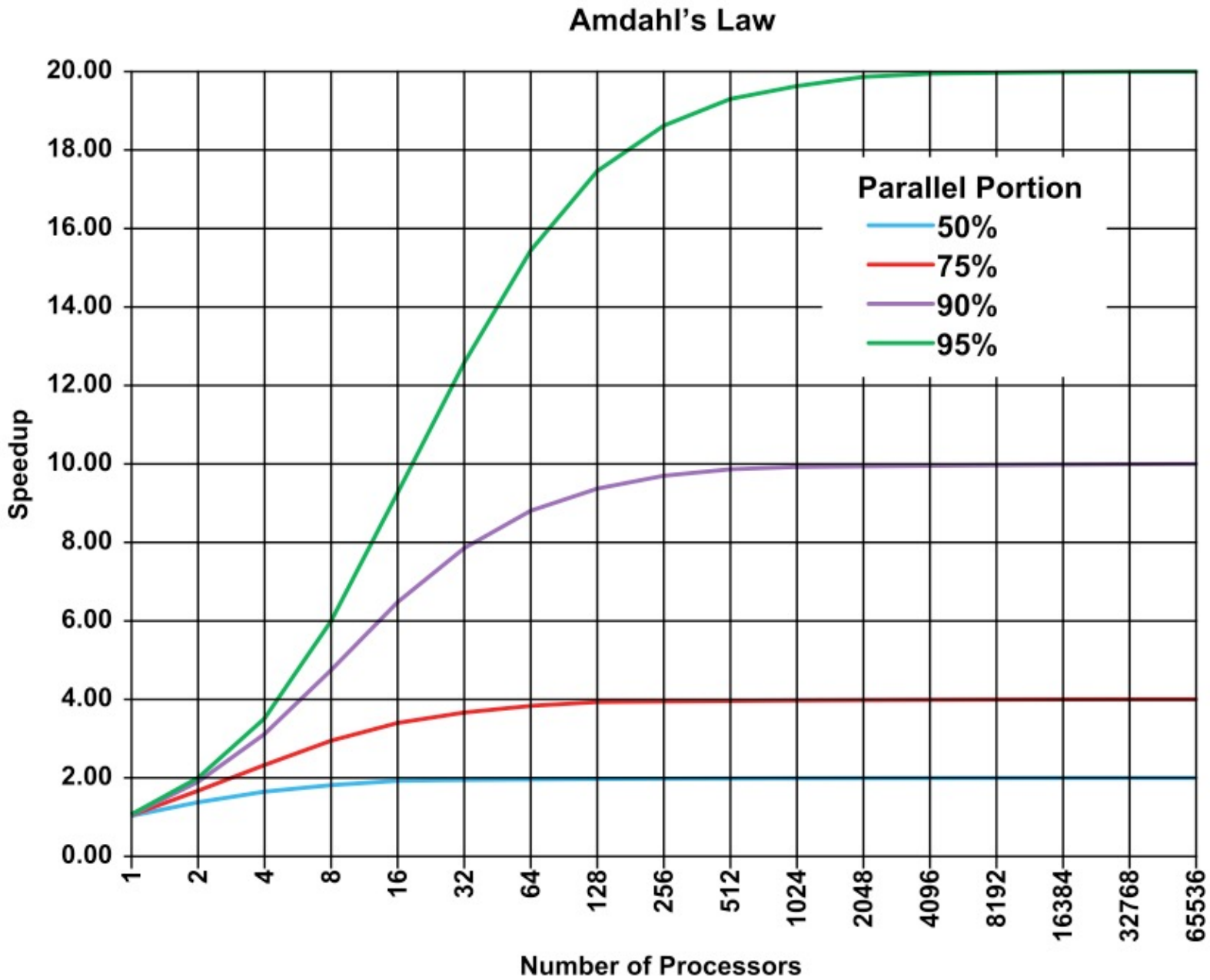
- Gene M. Amdahl, "*Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities*", 1967
- Amdahl's law (Amdahl's speedup model)

$$Speedup_{Amdahl} = \frac{1}{(1-f) + \frac{f}{n}}$$

$$\lim_{n \to \infty} Speedup_{Amdahl} = \frac{1}{1-f}$$

What is the speedup limit if f is 0.5 (50%) and 0.8 (80%), respectively?

- Implications
  - Limited speedup achievable by parallelism


Amdahl's Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Parallel Portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

Amdahl's Law in the Multicore Era

- Hill & Marty, "*Amdahl's Law in the Multicore Era*", IEEE Computer 2008

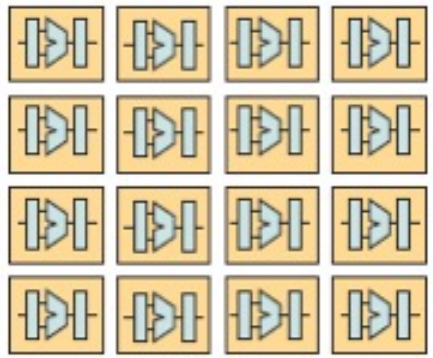
- Analyze the design choice and the performance

- Assumptions
  - Resources are bounded due to area, power, or cost, etc.
  - A Simple Base Core
    - Consumes 1 Base Core Equivalent (BCE) resources
    - Provides performance normalized to 1
  - An Enhanced Core
    - Consumes *r* BCEs
    - Performance as a function *perf(r)*
    - *perf(r)* < *r* in general, and is Square Root of *r* following Pollack's rule

Performance of Different Architectures

- Analyze performance of three multicore architecture organizations
  - Symmetric

Each Chip Bounded to *n* BCEs (for all cores)



n/r Cores per Chip

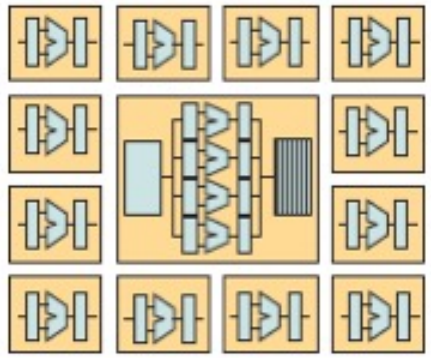
Symmetric/homogeneous

16 1-BCE cores

Performance of Different Architectures

- Analyze performance of three multicore architecture organizations
  - Asymmetric

Each Chip Bounded to *n* BCEs (for all cores)



Asymmetric/heterogeneous

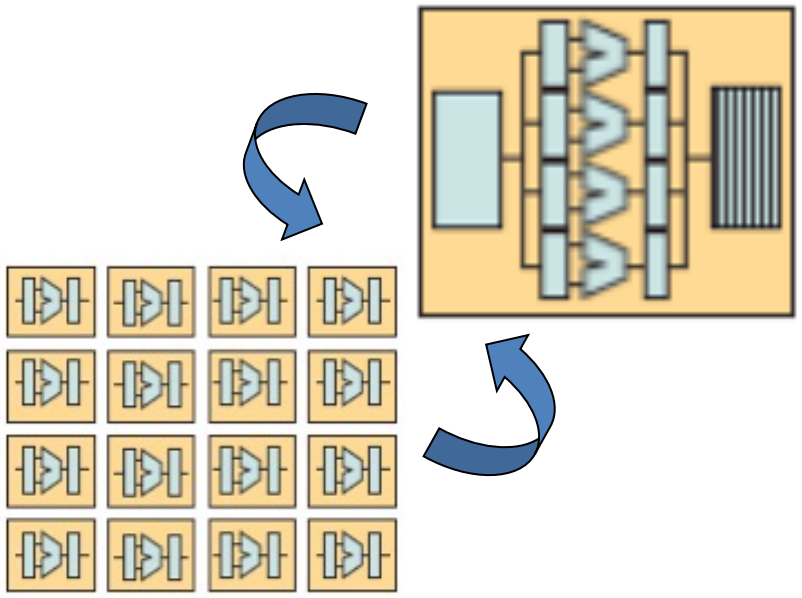
12 1-BCE cores + 1 4-BCE core

One r-BCE Core leaves *n-r* BCEs
Use *n-r* BCEs for *n-r* Base Cores
Therefore, 1 + *n* - *r* Cores per Chip

Performance of Different Architectures

- Analyze performance of three multicore architecture organizations
  - Dynamic

Each Chip Bounded to *n* BCEs (for all cores)



n Base Cores for Best Parallel Performance
Harness *r* Cores Together for Serial Performance

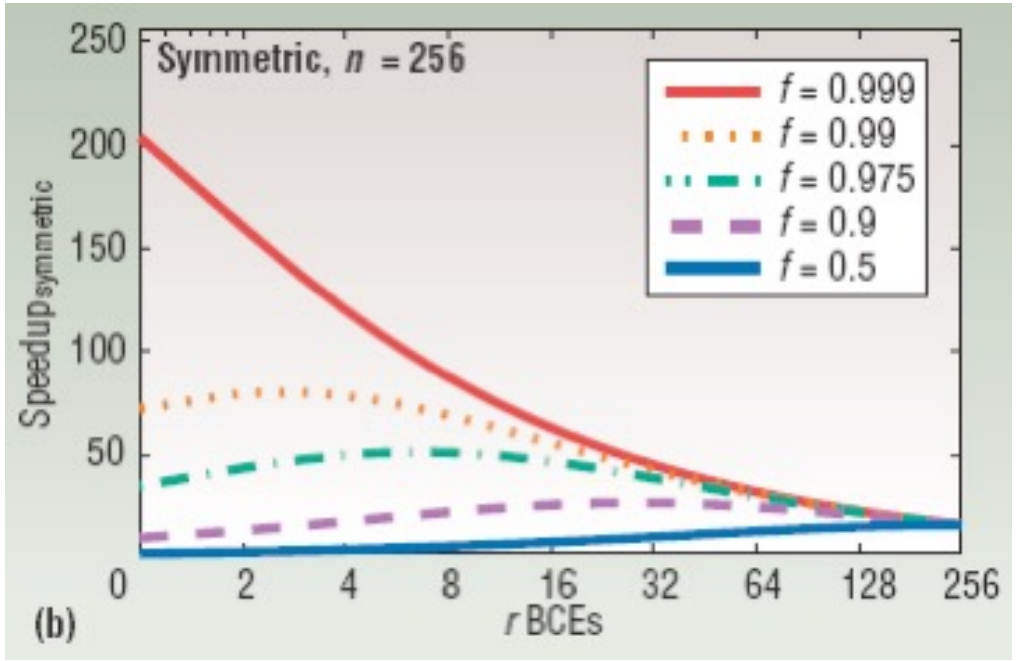
Dynamic

16 1-BCE cores or 1 16-BCE core

Performance of Different Architectures

- Speedup of symmetric architecture

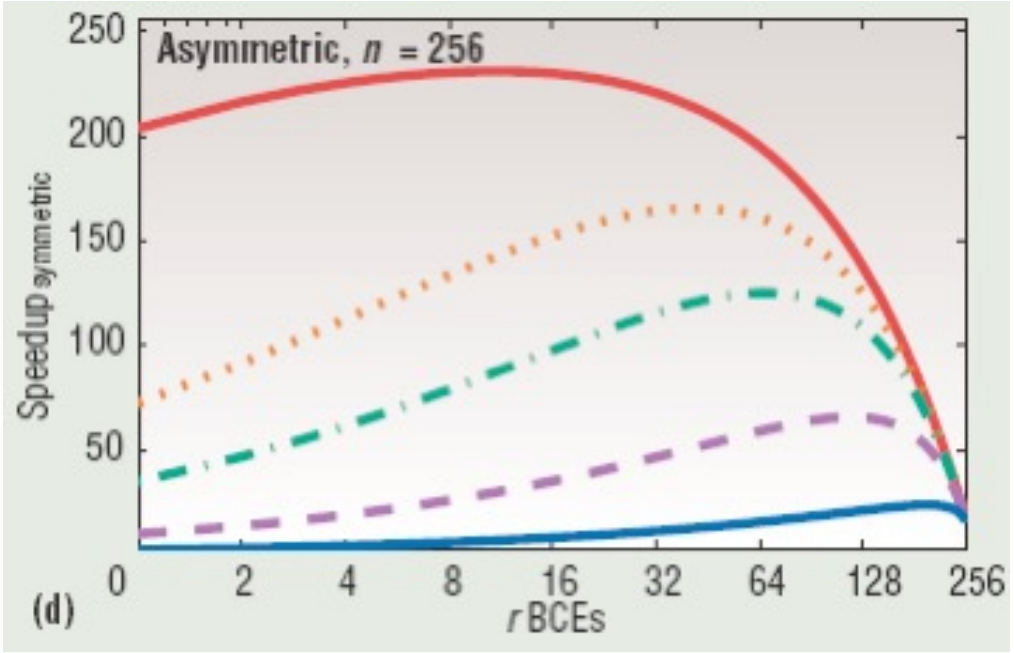
$$Speedup_{symmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}$$



Performance of Different Architectures

- Speedup of asymmetric architecture

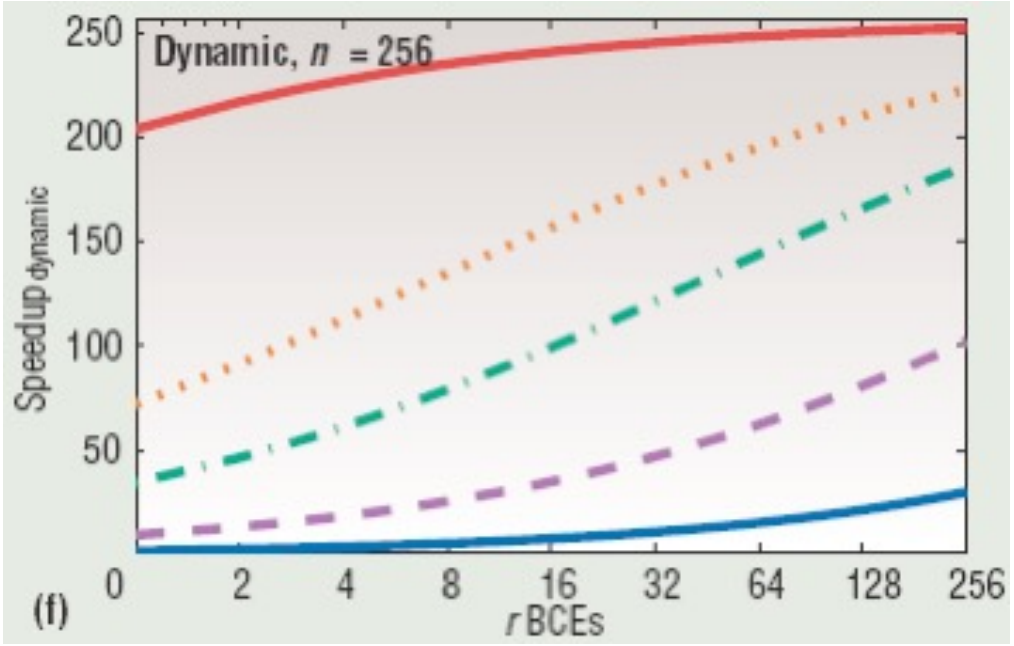
$$Speedup_{asymmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}}$$



Performance of Different Architectures

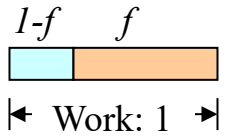
- Speedup of dynamic architecture

$$Speedup_{dynamic}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$



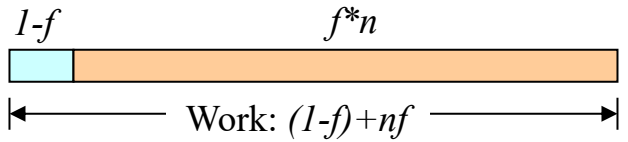
Scaled Computing Concept

- Tacit assumption in Amdahl's law
  - The problem size is fixed
  - The speedup emphasizes time reduction



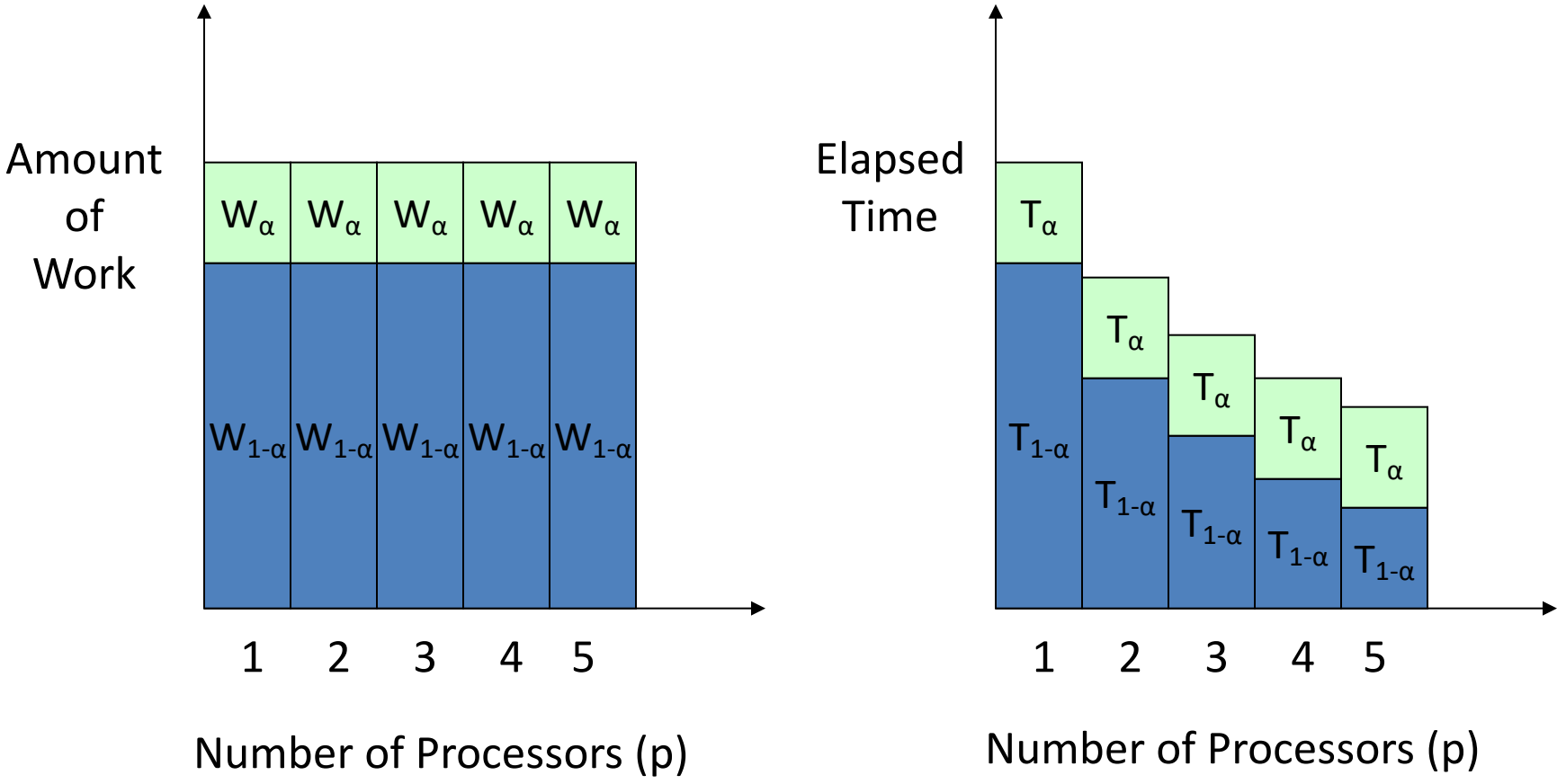
- Gustafson's Law, 1988
  - Fixed-time speedup model

$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload} = (1-f)+nf$$

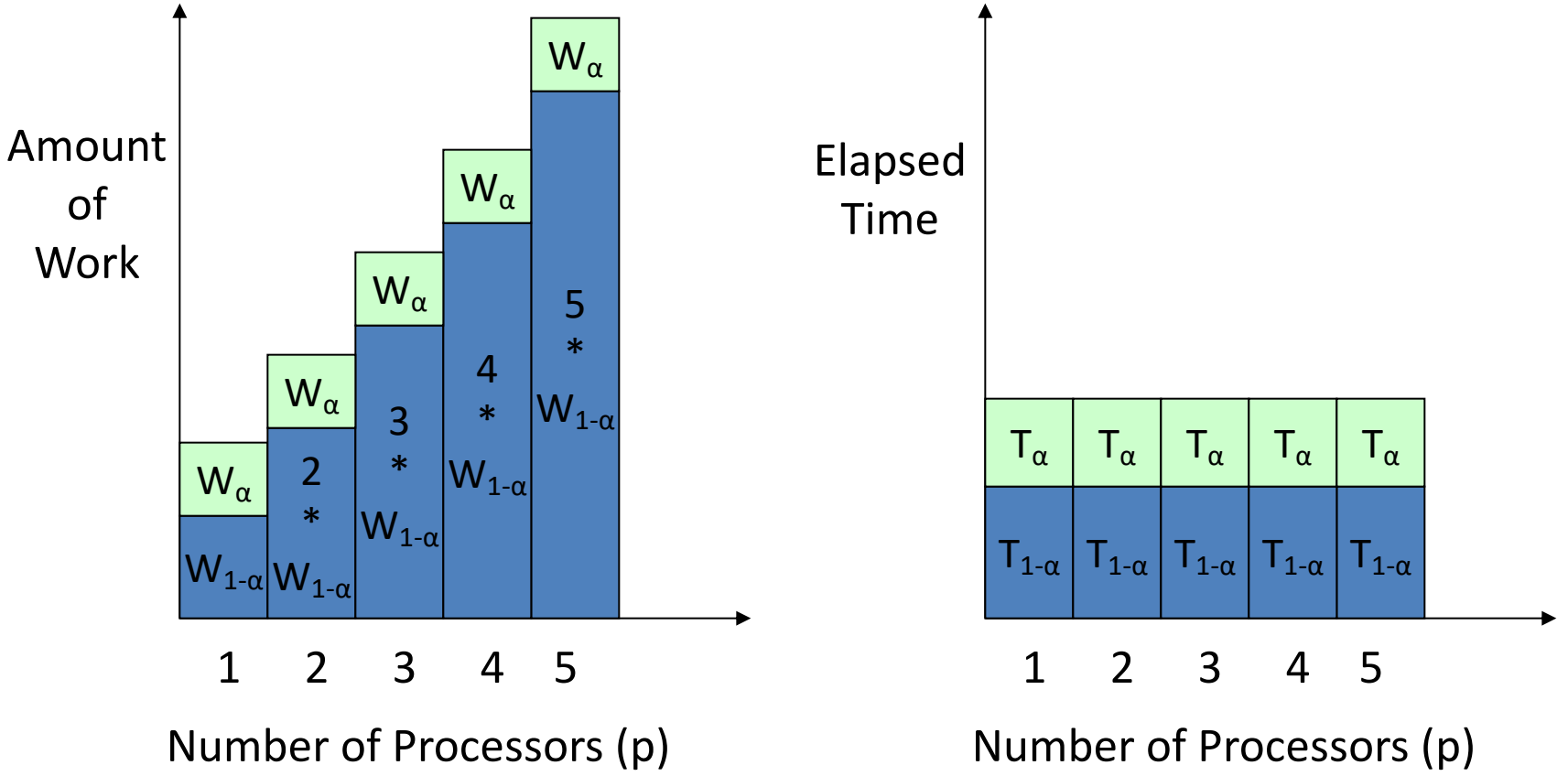


  - Shows the scaled speedup is a linear function of *n*
- Demonstrates great promise of parallelism

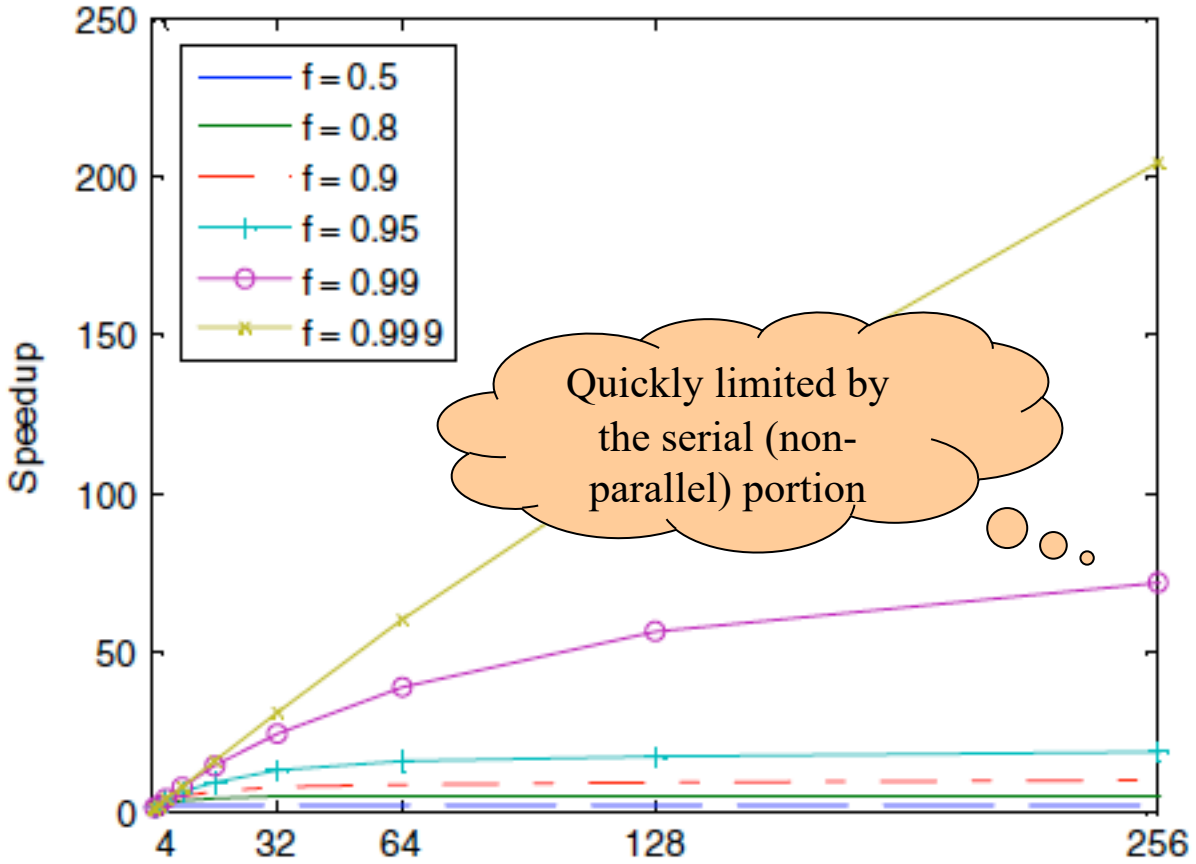
Fixed-Size Speedup



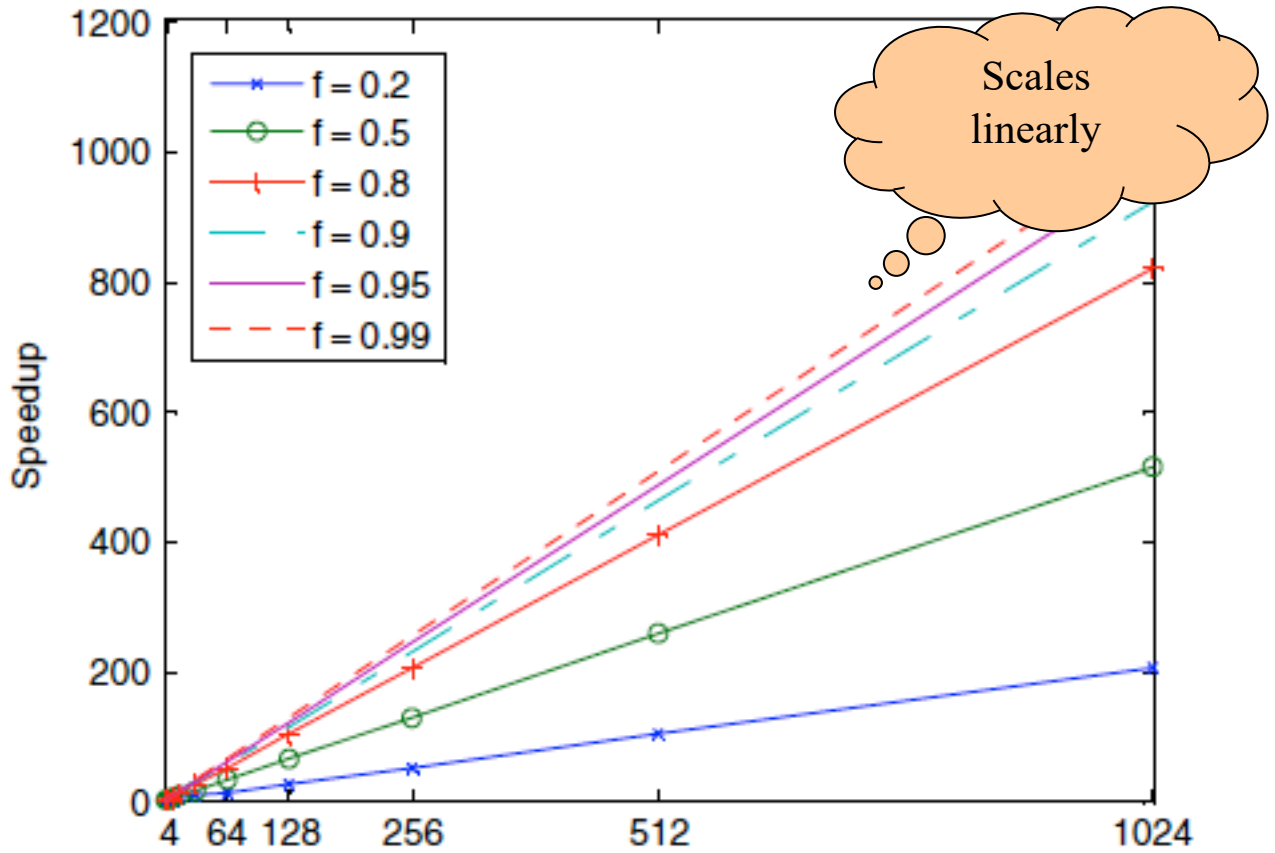
Fixed-Time Speedup



Fixed-size Speedup



Fixed-time Speedup



Readings

- Chapter 1, 1.4-1.9
- Gene M. Amdahl, “Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities”, 1967
- John L. Gustafson, “Reevaluating Amdahl’s Law”, 1988
- Hill & Marty, “Amdahl’s Law in the Multicore Era”, IEEE Computer 2008
- X.-H. Sun and Y. Chen, "Reevaluating Amdahl's Law in the Multicore Era,” Journal of Parallel and Distributed Computing, vol. 70, no. 2, pp. 183-188, Feb 2010.