CS 61C:
Great Ideas in Computer Architecture
Performance
Iron Law, Amdahl's Law

Instructors:

Nicholas Weaver& Vladimir Stojanovic

http://inst.eecs.berkeley.edu/~cs61c/

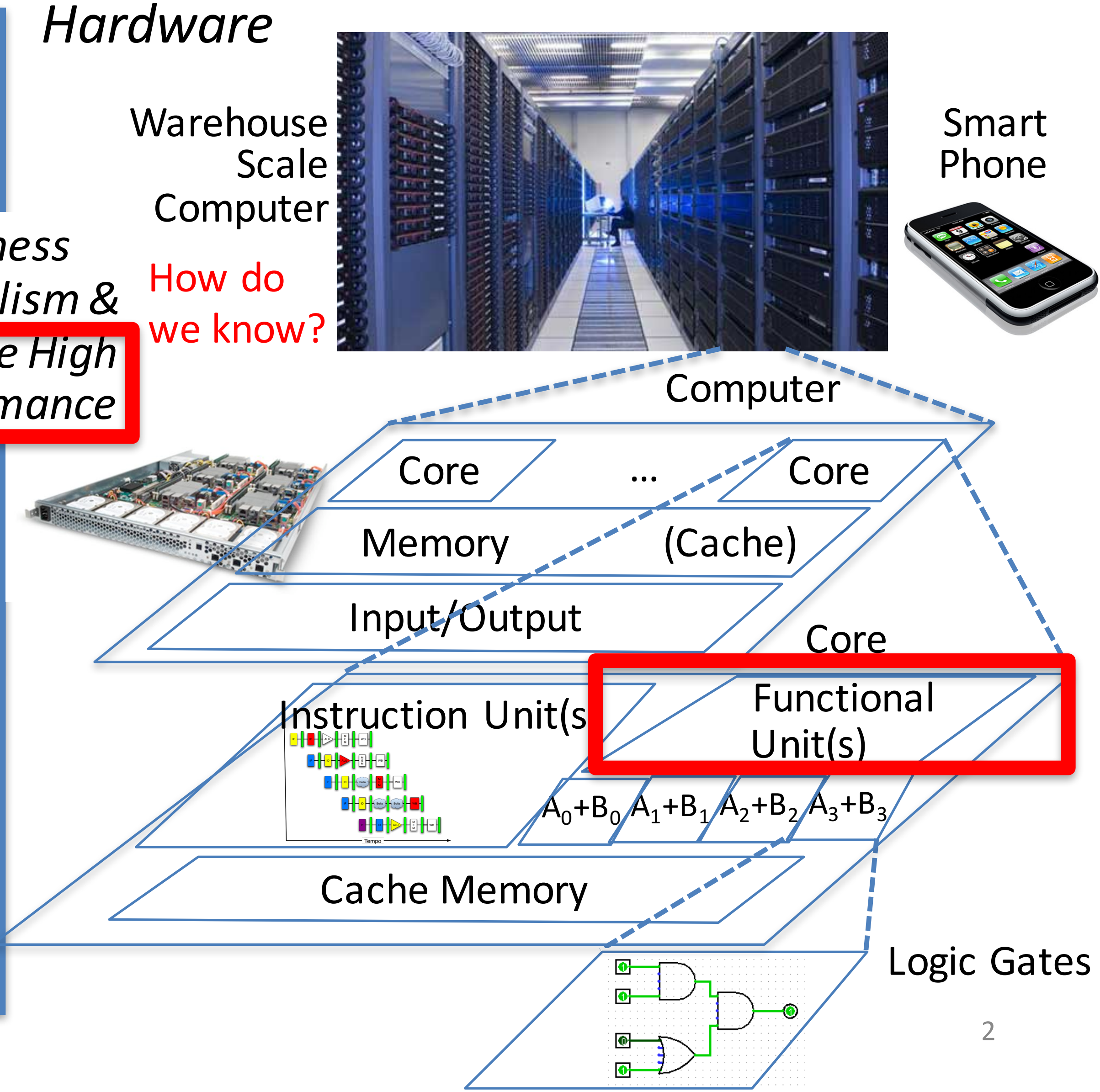
New-School Machine Structures (It’s a bit more complicated!)

Software

- Parallel Requests
  - Assigned to computer
  - e.g., Search “Katz”
- Parallel Threads
  - Assigned to core
  - e.g., Lookup, Ads
- Parallel Instructions
  - >1 instruction @ one time
  - e.g., 5 pipelined instructions
- Parallel Data
  - >1 data item @ one time
  - e.g., Add of 4 pairs of words
- Hardware descriptions
  - All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance

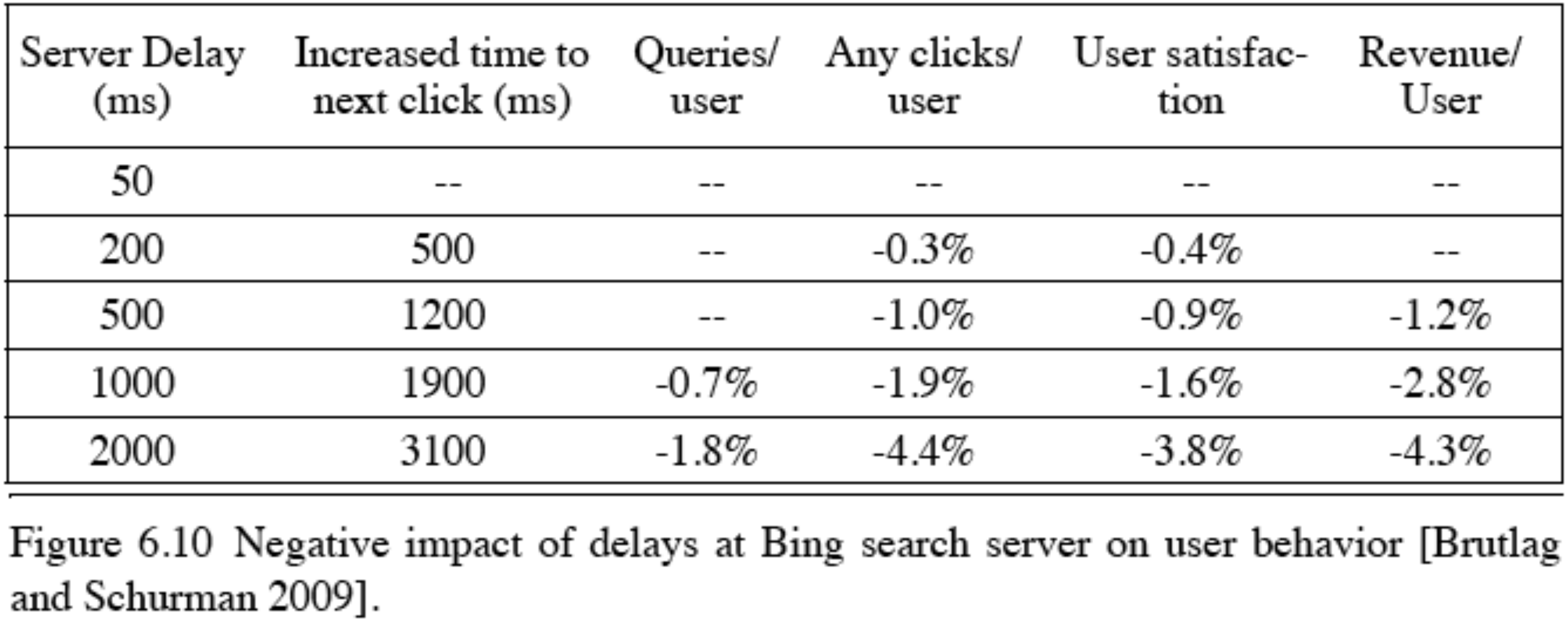
Hardware



What is Performance?

- *Latency* (or *response time* or *execution time*)
  - Time to complete one task
- *Bandwidth* (or *throughput*)
  - Tasks completed per unit time
    - If you have sufficient independent tasks, you can always throw more money at the problem: Throughput/$ often a more important metric than just throughput

Cloud Performance: Why Application Latency Matters



Server Delay (ms)	Increased time to next click (ms)	Queries/ user	Any clicks/ user	User satisfaction	Revenue/ User
50	--	--	--	--	--
200	500	--	-0.3%	-0.4%	--
500	1200	--	-1.0%	-0.9%	-1.2%
1000	1900	-0.7%	-1.9%	-1.6%	-2.8%
2000	3100	-1.8%	-4.4%	-3.8%	-4.3%

Figure 6.10 Negative impact of delays at Bing search server on user behavior [Brutlag and Schurman 2009].

- Key figure of merit: application responsiveness
  - Longer the delay, the fewer the user clicks, the less the user happiness, and the lower the revenue per user

Defining CPU Performance

- What does it mean to say
  X is faster than Y?
- Ferrari vs. School Bus?

- 2013 Ferrari 599 GTB
  - 2 passengers, quarter mile in 10 secs
- 2013 Type D school bus
  - 50 passengers, quarter mile in 20 secs

- *Response Time* (*Latency*): e.g., time to travel ¼ mile
- *Throughput* (*Bandwidth*): e.g., passenger-mi in 1 hour

Defining Relative CPU Performance

- $\text{Performance}_X = 1/\text{Program Execution Time}_X$
- $\text{Performance}_X > \text{Performance}_Y \Rightarrow$
  $1/\text{Execution Time}_X > 1/\text{Execution Time}_y \Rightarrow$
  $\text{Execution Time}_Y > \text{Execution Time}_X$
- Computer X is N times faster than Computer Y
  $\text{Performance}_X / \text{Performance}_Y = N$ or
  $\text{Execution Time}_Y / \text{Execution Time}_X = N$

- Bus to Ferrari performance:
  - Program: Transfer 1000 passengers for 1 mile
  - Bus: 3,200 sec, Ferrari: 40,000 sec

Measuring CPU Performance

- Computers use a clock to determine when events takes place within hardware
- *Clock cycles:* discrete time intervals
  - aka clocks, cycles, clock periods, clock ticks
- *Clock rate* or *clock frequency:* clock cycles per second (inverse of clock cycle time)
- 3 GigaHertz clock rate
  => clock cycle time = $1/(3\text{x}10^9)$ seconds
  clock cycle time = 333 picoseconds (ps)

CPU Performance Factors

- To distinguish between processor time and I/O, *CPU time* is time spent in processor

-
  ```
  CPU Time/Program
     = Clock Cycles/Program
       x Clock Cycle Time
  ```
- Or

  ```
  CPU Time/Program
   = Clock Cycles/Program ÷ Clock Rate
  ```

Iron Law of Performance

by Emer and Clark

- A program executes instructions
- CPU Time/Program
  = Clock Cycles/Program x Clock Cycle Time
  = Instructions/Program
  x Average Clock Cycles/Instruction
  x Clock Cycle Time
- 1st term called *Instruction Count*
- 2nd term abbreviated *CPI* for average *Clock Cycles **Per Instruction***
- 3rd term is 1 / Clock rate

Restating Performance Equation

- $$\text{Time} = \frac{\text{Seconds}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$

What Affects Each Component?
A)Instruction Count, B)CPI, C)Clock Rate

	Affects What? (click in letter of component not affected)
Algorithm	
Programming Language	
Compiler	
Instruction Set Architecture	

What Affects Each Component? Instruction Count, CPI, Clock Rate

	Affects What?
Algorithm	Instruction Count, CPI
Programming Language	Instruction Count, CPI
Compiler	Instruction Count, CPI
Instruction Set Architecture	Instruction Count, Clock Rate, CPI

Clickers

Computer	Clock frequency	Clock cycles per instruction	#instructions per program
A	1GHz	2	1000
B	2GHz	5	800
C	500MHz	1.25	400
D	5GHz	10	2000

- Which computer has the highest performance for a given program?

Workload and Benchmark

- *Workload:* Set of programs run on a computer
  - Actual collection of applications run or made from real programs to approximate such a mix
  - Specifies programs, inputs, and relative frequencies
- *Benchmark:* Program selected for use in comparing computer performance
  - Benchmarks form a workload
  - Usually standardized so that many use them

SPEC
(System Performance Evaluation Cooperative)

- Computer Vendor cooperative for benchmarks, started in 1989
- SPECCPU2006
  - 12 Integer Programs
  - 17 Floating-Point Programs
- Often turn into number where bigger is faster
- *SPECratio*: reference execution time on old reference computer divide by execution time on new computer to get an effective speed-up

SPECINT2006 on AMD Barcelona

Description	Instruction Count (B)	CPI	Clock cycle time (ps)	Execution Time (s)	Reference Time (s)	SPEC-ratio
Interpreted string processing	2,118	0.75	400	637	9,770	15.3
Block-sorting compression	2,389	0.85	400	817	9,650	11.8
GNU C compiler	1,050	1.72	400	724	8,050	11.1
Combinatorial optimization	336	10.0	400	1,345	9,120	6.8
Go game	1,658	1.09	400	721	10,490	14.6
Search gene sequence	2,783	0.80	400	890	9,330	10.5
Chess game	2,176	0.96	400	837	12,100	14.5
Quantum computer simulation	1,623	1.61	400	1,047	20,720	19.8
Video compression	3,102	0.80	400	993	22,130	22.3
Discrete event simulation library	587	2.94	400	690	6,250	9.1
Games/path finding	1,082	1.79	400	773	7,020	9.1
XML parsing	1,058	2.70	400	1,143	6,900	6.0

Summarizing Performance …

System	Rate (Task 1)	Rate (Task 2)
A	10	20
B	20	10

Clickers: Which system is faster?

A: System A
B: System B
C: Same performance
D: Unanswerable question!

… Depends Who's Selling

System	Rate (Task 1)	Rate (Task 2)	Average
A	10	20	15
B	20	10	15

Average throughput

System	Rate (Task 1)	Rate (Task 2)	Average
A	0.50	2.00	1.25
B	1.00	1.00	1.00

Throughput relative to B

System	Rate (Task 1)	Rate (Task 2)	Average
A	1.00	1.00	1.00
B	2.00	0.50	1.25

Throughput relative to A

Summarizing SPEC Performance

- Varies from 6x to 22x faster than reference computer
- *Geometric mean* of ratios: N-th root of product of N ratios

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

  - Geometric Mean gives same relative answer no matter what computer is used as reference
- Geometric Mean for Barcelona is 11.7

Administrivia

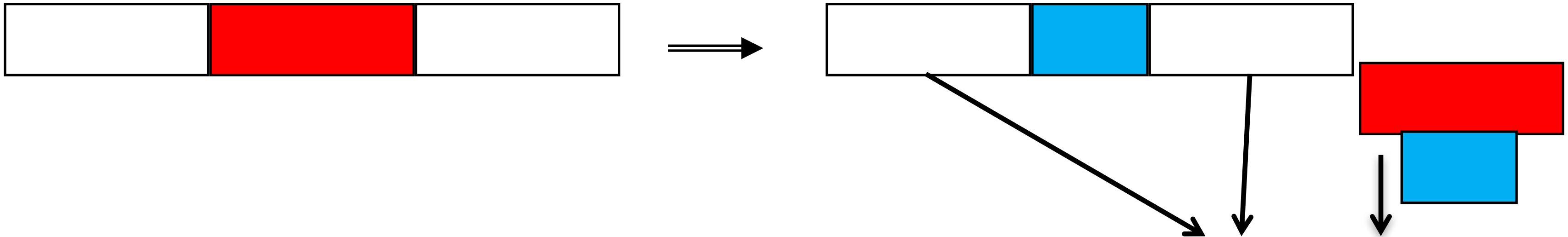
- Midterm 2 in the evening next Monday
- Project 2.1 grades in

Big Idea: Amdahl's (Heartbreaking) Law

- Speedup due to enhancement E is

$$\text{Speedup w/ E} = \frac{\text{Exec time w/o E}}{\text{Exec time w/ E}}$$

- Suppose that enhancement E accelerates a fraction F (F <1) of the task by a factor S (S>1) and the remainder of the task is unaffected



$$\text{Execution Time w/ E} = \text{Execution Time w/o E} \times [\,(1\text{-}F) + F/S\,]$$

$$\text{Speedup w/ E} = 1 / [\,(1\text{-}F) + F/S\,]$$

Big Idea: Amdahl’s Law

$$\text{Speedup} = \frac{1}{(1 - F) + \frac{F}{S}}$$

Non-speed-up part

Speed-up part

Example: the execution time of half of the program can be accelerated by a factor of 2.
What is the program speed-up overall?

$$\frac{1}{0.5 + \frac{0.5}{2}} = \frac{1}{0.5 + 0.25} = 1.33$$

Example #1: Amdahl's Law

Speedup w/ E = 1 / [(1-F) + F/S]

- Consider an enhancement which runs 20 times faster but which is only usable 25% of the time

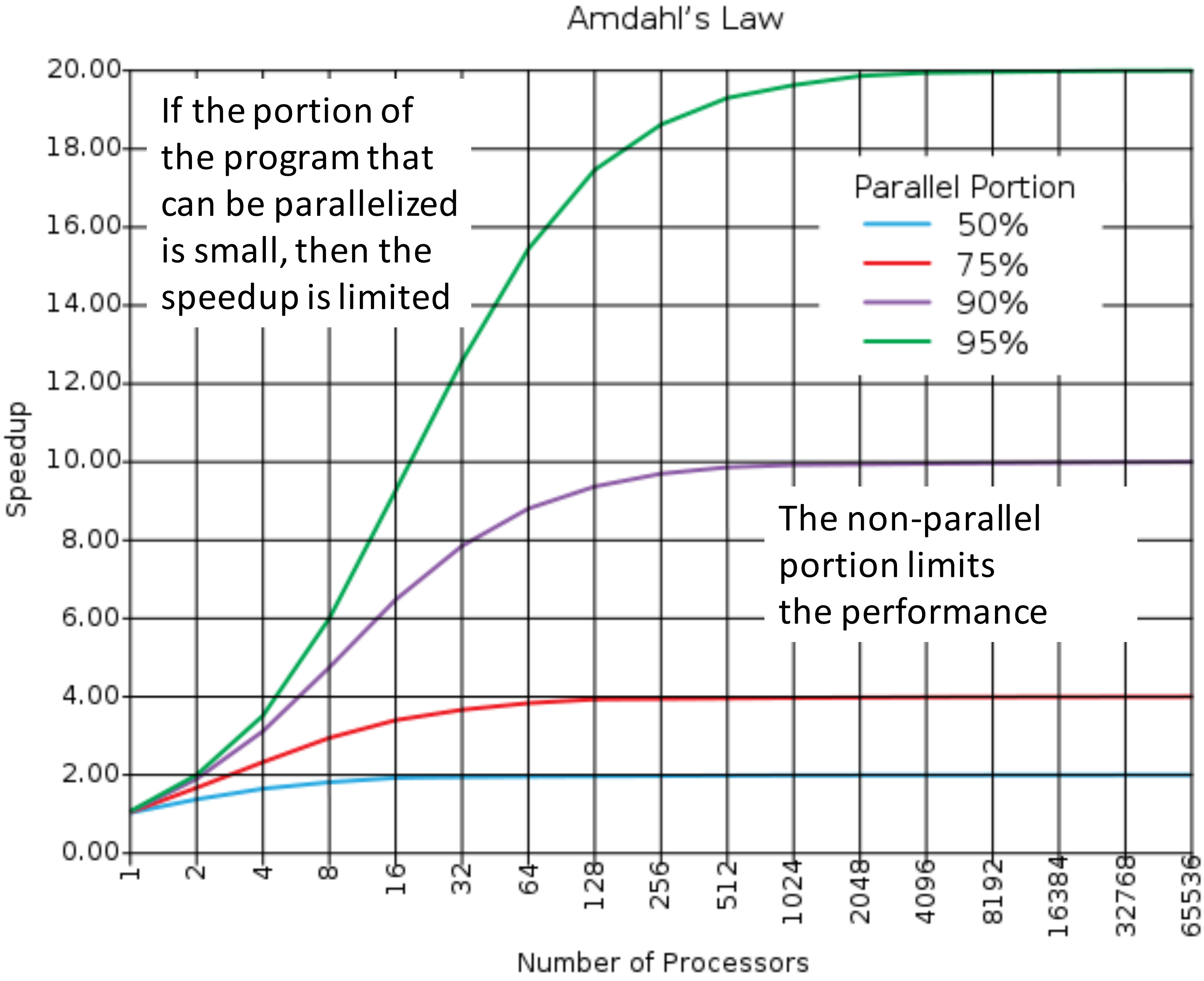
  Speedup w/ E = 1/(.75 + .25/20) = 1.31

- What if its usable only 15% of the time?

  Speedup w/ E = 1/(.85 + .15/20) = 1.17

- Amdahl's Law tells us that to achieve linear speedup with 100 processors, none of the original computation can be scalar!
- To get a speedup of 90 from 100 processors, the percentage of the original program that could be scalar would have to be 0.1% or less

  Speedup w/ E = 1/(.001 + .999/100) = 90.99



If the portion of the program that can be parallelized is small, then the speedup is limited

The non-parallel portion limits the performance

Strong and Weak Scaling

- To get good speedup on a parallel processor while keeping the problem size fixed is harder than getting good speedup by increasing the size of the problem.
  - *Strong scaling*: when speedup can be achieved on a parallel processor without increasing the size of the problem
  - *Weak scaling*: when speedup is achieved on a parallel processor by increasing the size of the problem proportionally to the increase in the number of processors
- Load balancing is another important factor: every processor doing same amount of work
  - Just one unit with twice the load of others cuts speedup almost in half

Clickers/Peer Instruction

Suppose a program spends 80% of its time in a square root routine. How much must you speedup square root to make the program run 5 times faster?

Speedup w/ E = $1 / [(1-F) + F/S]$

A: 5

B: 16

C: 20

D: 100

E: None of the above

And In Conclusion, …

- Time (seconds/program) is measure of performance

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$

- Floating-point representations hold approximations of real numbers in a finite number of bits