Performance (III): How to Evaluate Performance Fairly or ... Fool Others with Performance Metrics

Hung-Wei Tseng

NETFLIX

UNLIMITED TV SHOWS & MOVIES

N SERIES

GREAT PRETENDER

Great Pretender

2020 | TV-MA | 1 Season | Drama Anime

Supposedly Japan's greatest swindler, Makoto Edamura gets more than he bargained for when he tries to con Laurent Thierry, a real world-class crook.

Starring: Chiaki Kobayashi, Junichi Suwabe, Natsumi Fujiwara

Recap: von Neumman Architecture

509cbd23

00c2e800

Processor

Instructions	Data
0f00bb27	00c2e800
509cbd23	00000008
00005d24	00c2f000
0000bd24	00000008
2ca422a0	00c2f800
130020e4	00000008
00003d24	00c30000
2ca4e2b3	00000008

Memory

Program

Instructions	Data
0f00bb27	00c2e800
509cbd23	00000008
00005d24	00c2f000
0000bd24	00000008
2ca422a0	00c2f800
130020e4	00000008
00003d24	00c30000
2ca4e2b3	00000008

Storage

Recap: Summary of CPU Performance Equation

$$Performance = \frac{1}{Execution\ Time}$$

$$Execution\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

$$ET = IC \times CPI \times CT$$

- IC (Instruction Count)
  - ISA, Compiler, algorithm, programming language, **programmer**
- CPI (Cycles Per Instruction)
  - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language, **programmer**
- Cycle Time (Seconds Per Cycle)
  - Process Technology, microarchitecture, **programmer**

Recap: Speedup

- The relative performance between two machines, X and Y. Y is n times faster than X

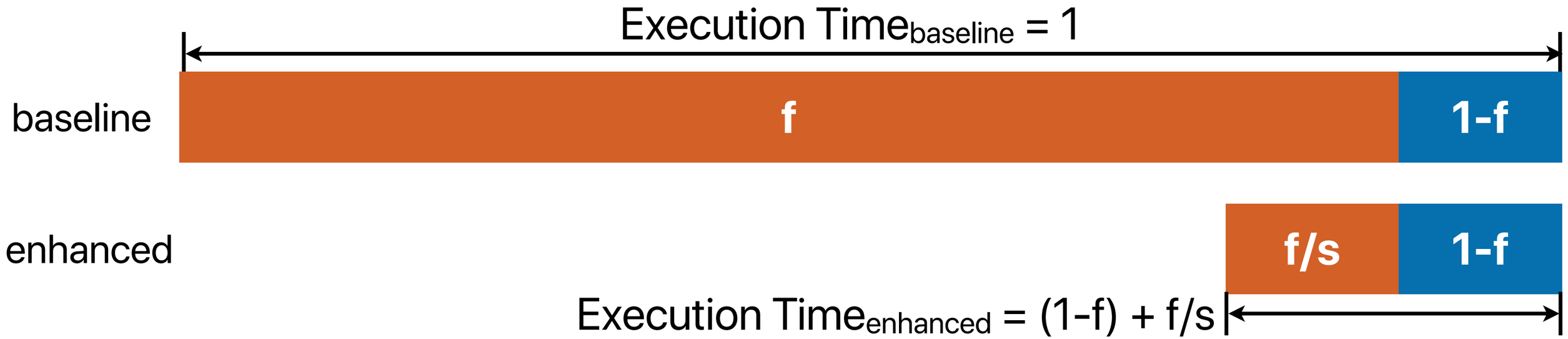
$$n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- The speedup of Y over X

$$Speedup = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

Recap: Amdahl's Law

$$Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$$



$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}} = \frac{1}{(1-f) + \frac{f}{s}}$$

Recap: Amdahl's Law Corollary #1

- The maximum speedup is bounded by

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f) + \frac{f}{\infty}}$$

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

Recap: Corollary #1 on Multiple Optimizations

- If we can pick just one thing to work on/optimize

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1 - f_1)}$$
$$Speedup_{max}(f_2, \infty) = \frac{1}{(1 - f_2)}$$
$$Speedup_{max}(f_3, \infty) = \frac{1}{(1 - f_3)}$$
$$Speedup_{max}(f_4, \infty) = \frac{1}{(1 - f_4)}$$

The biggest f_x would lead to the largest $Speedup_{max}$!

Recap: Corollary #2 — make the common case fast!

- When f is small, optimizations will have little effect.
- Common == **most time consuming** not necessarily the most frequent
- The uncommon case doesn’t make much difference
- The common case can change based on inputs, compiler options, optimizations you’ve applied, etc.

Team scores

0	1	2	2

Outline

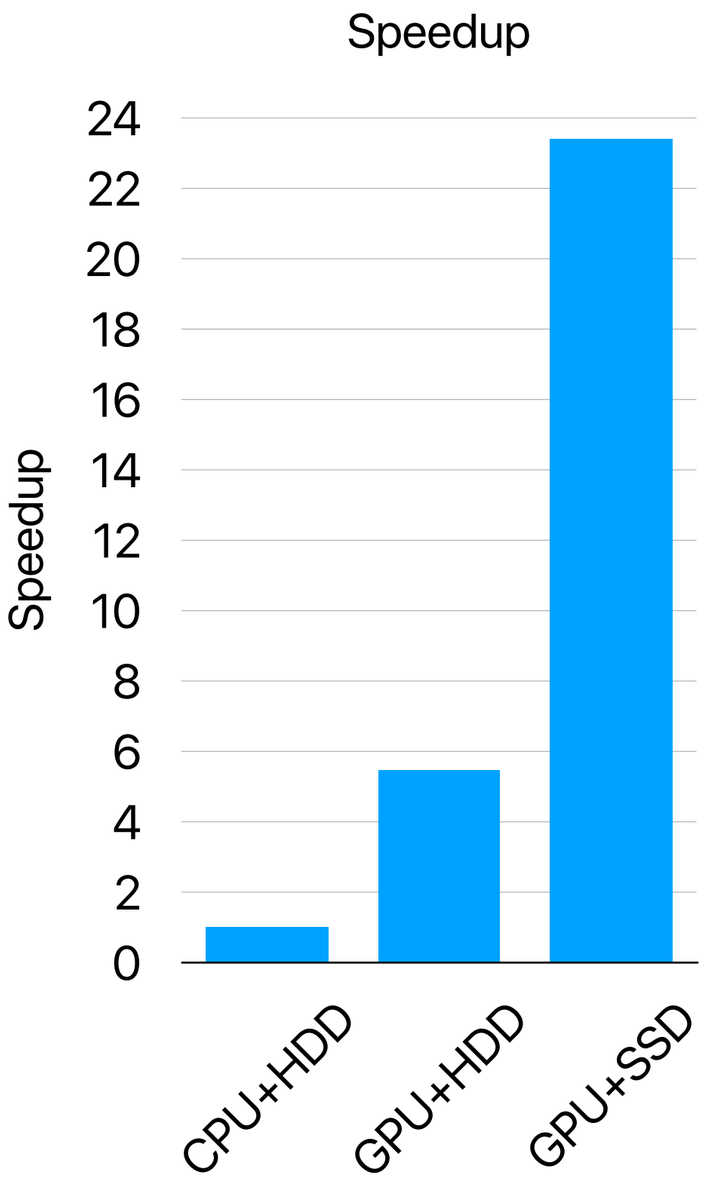
- Amdahl's law and it's implications (cont.)
- How to fairly present/compare your system performance or … fool others…
- High-level view of your memory hierarchy (if we have time)

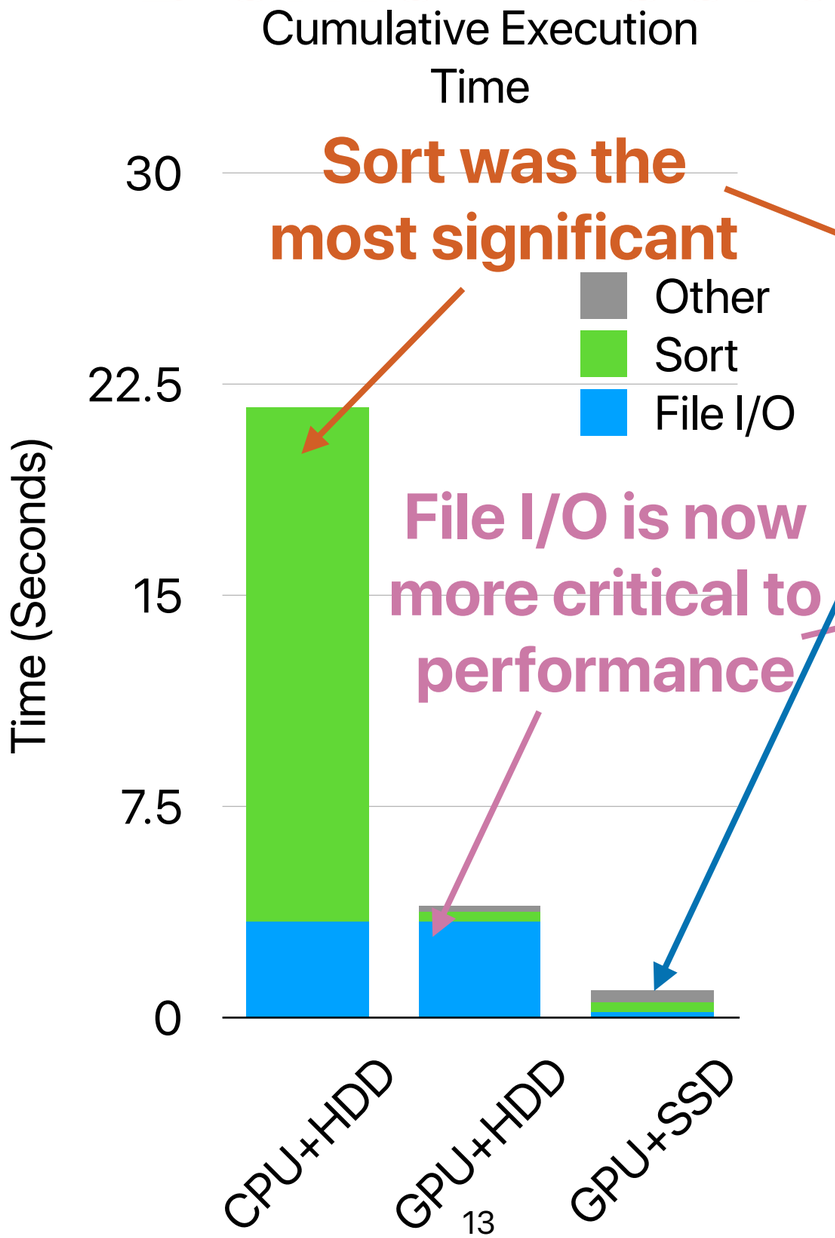
Amdahl's Law — and It's Implication in the Multicore Era (cont.)

H&P Chapter 1.9

M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. In Computer, vol. 41, no. 7, pp. 33-38, July 2008.

Demo — sort

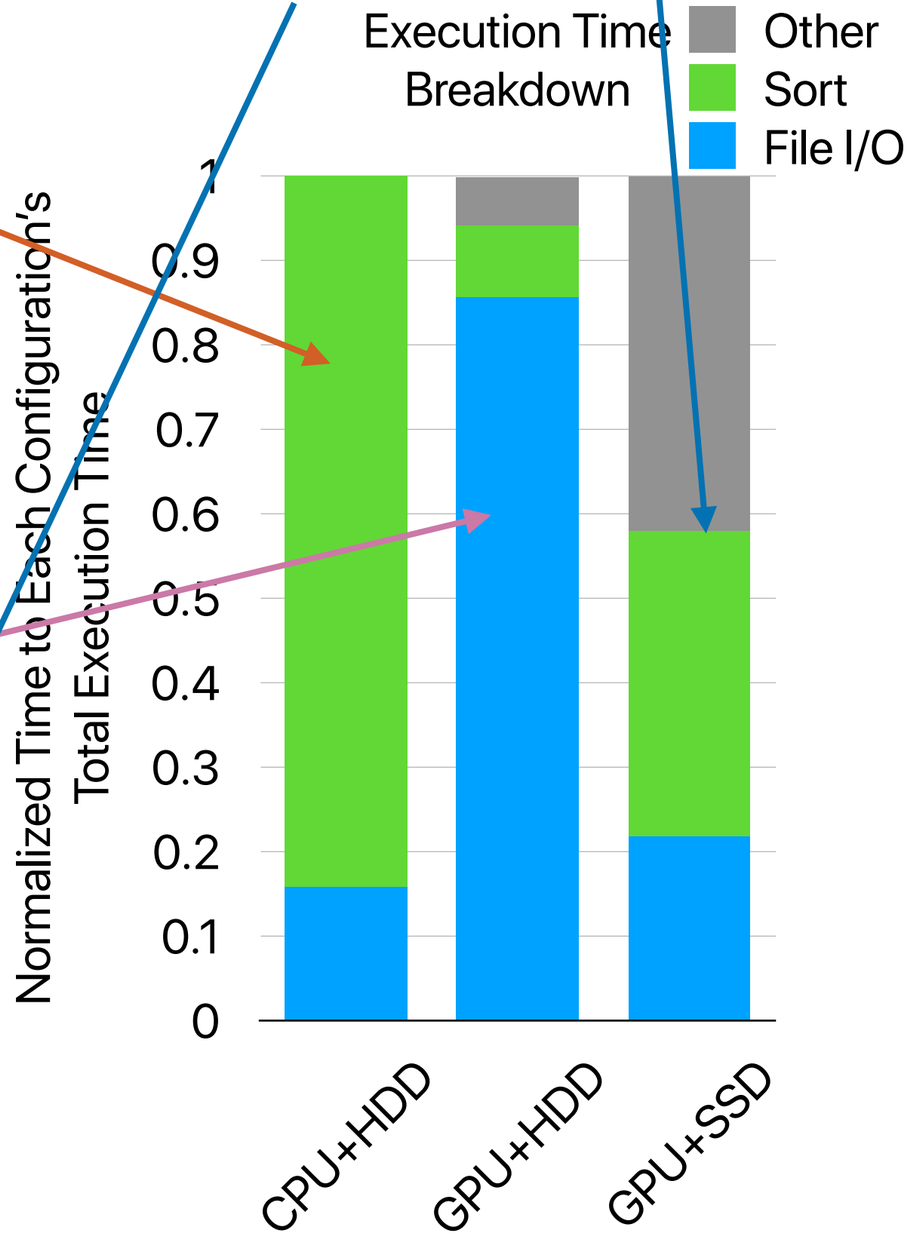




Sort was the most significant

File I/O is now more critical to performance

Something else (e.g., data movement) matters more



If we repeatedly optimizing our design based on Amdahl's law…

- With optimization, the common becomes uncommon.
- An uncommon case will (hopefully) become the new common case.
- Now you have a new target for optimization — You have to revisit "Amdahl's Law" every time you applied some optimization

Amdahl's Law on Multicore Architectures

- Symmetric multicore processor with n cores (if we assume the processor performance scales perfectly)

$$Speedup_{parallel}(f_{parallelizable}, n) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{n}}$$

Poll close in 1:30

Amdahl's Law on Multicore Architectures

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?
  - ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded
  - ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore
  - ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts
  - ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Poll close in 1:30

Amdahl's Law on Multicore Architectu

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?
  - ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded
  - ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore
  - ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts
  - ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Amdahl's Law on Multicore Architectures

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable \times Speedup(<1)}{\infty}}$$

  ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded

  ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

speedup is determined by 1-f

  ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts

  ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Corollary #3

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{\infty}}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

- Single-core performance still matters
  - It will eventually dominate the performance
  - If we cannot improve single-core performance further, finding more "parallelizable" parts is more important

Recap: Demo (2) — merge sort v.s. bitonic sort on GPUs

Merge Sort

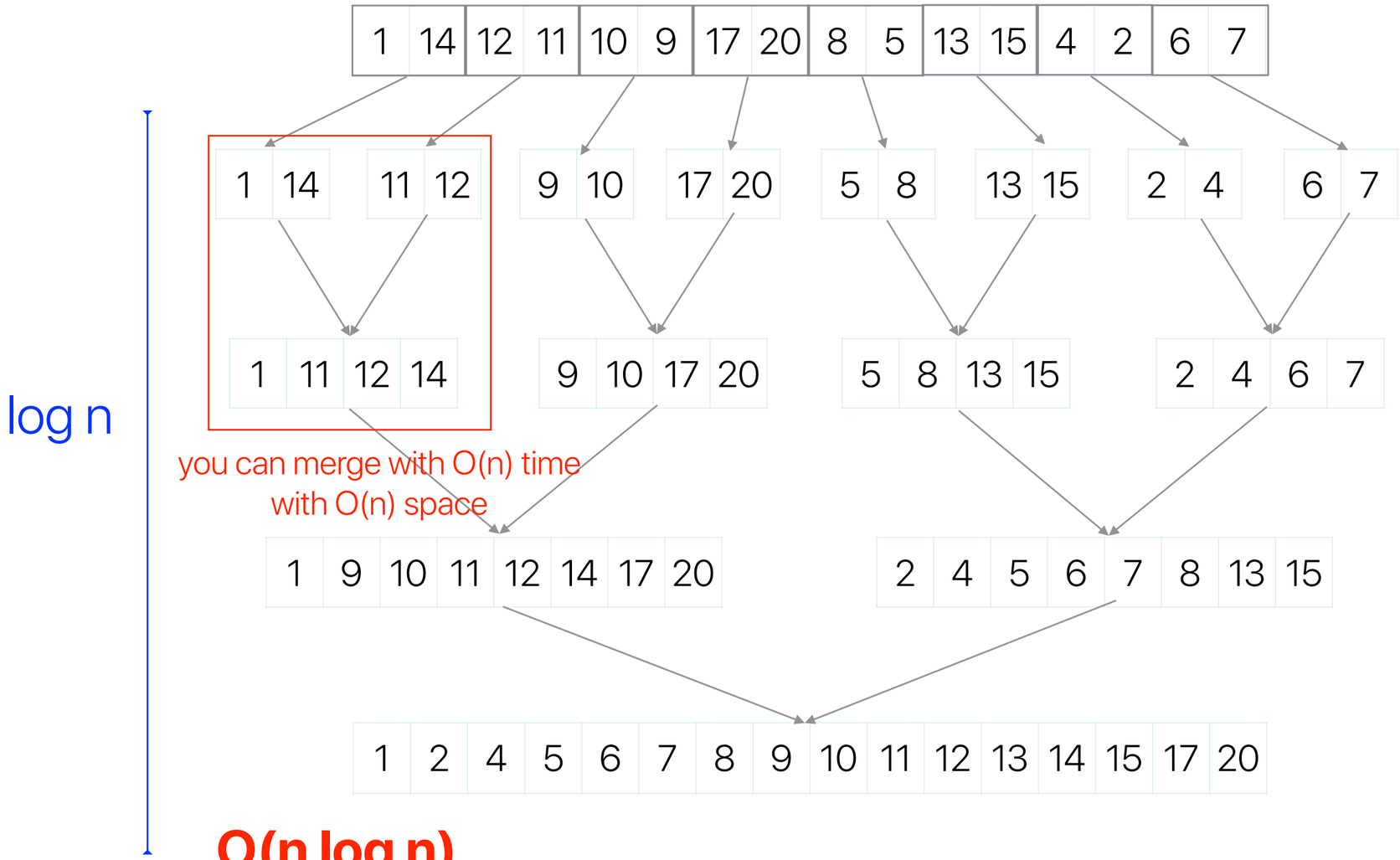
$O(nlog_2n)$

Bitonic Sort

$O(nlog_2^2n)$

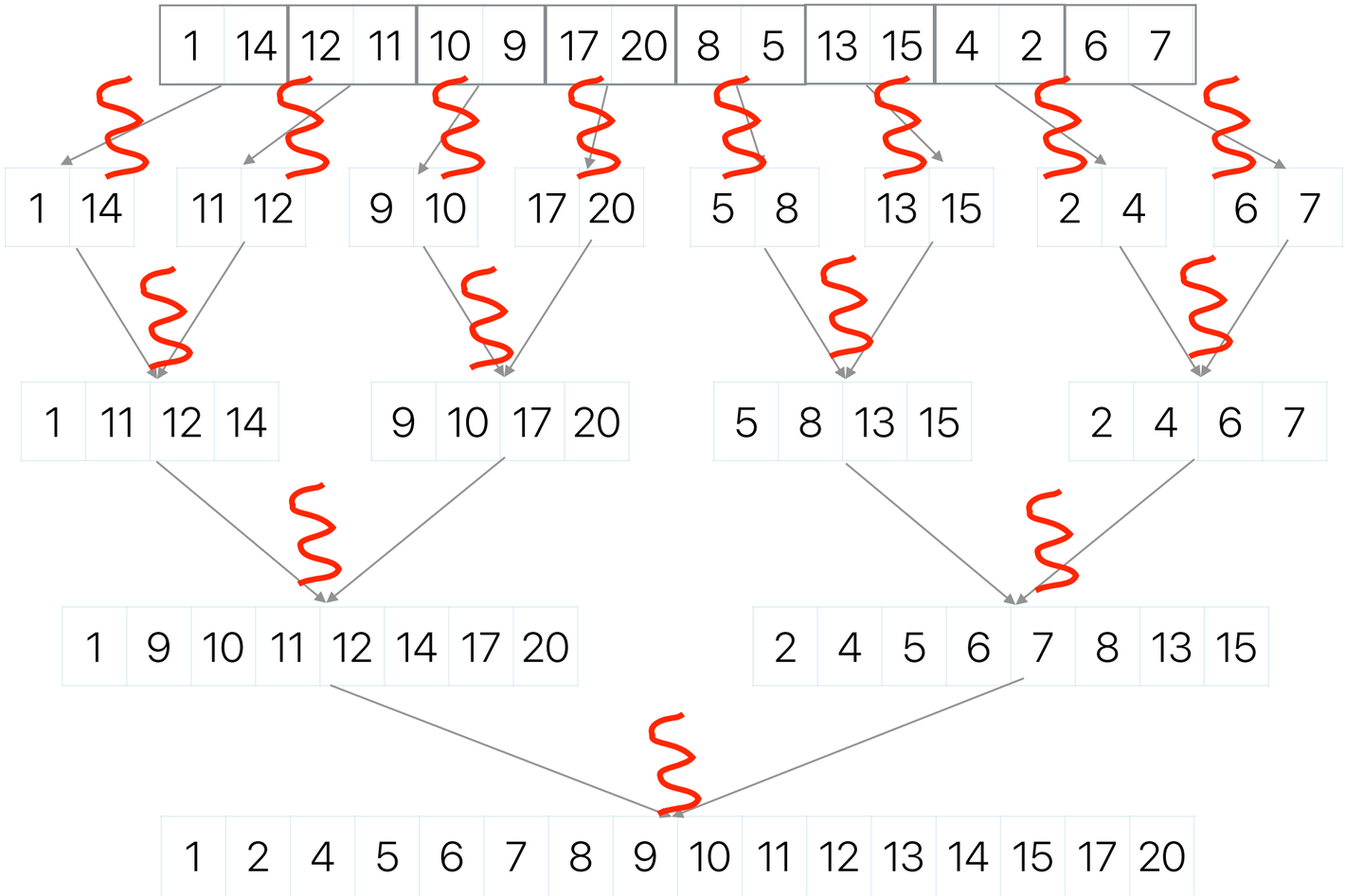
```
void BitonicSort() {

    int i,j,k;

    for (k=2; k<=N; k=2*k) {
        for (j=k>>1; j>0; j=j>>1) {
            for (i=0; i<N; i++) {
                int ij=i^j;
                if ((ij)>i) {
                    if ((i&k)==0 && a[i] > a[ij])
                        exchange(i,ij);
                    if ((i&k)!=0 && a[i] < a[ij])
                        exchange(i,ij);
                }
            }
        }
    }
}
```

Merge sort

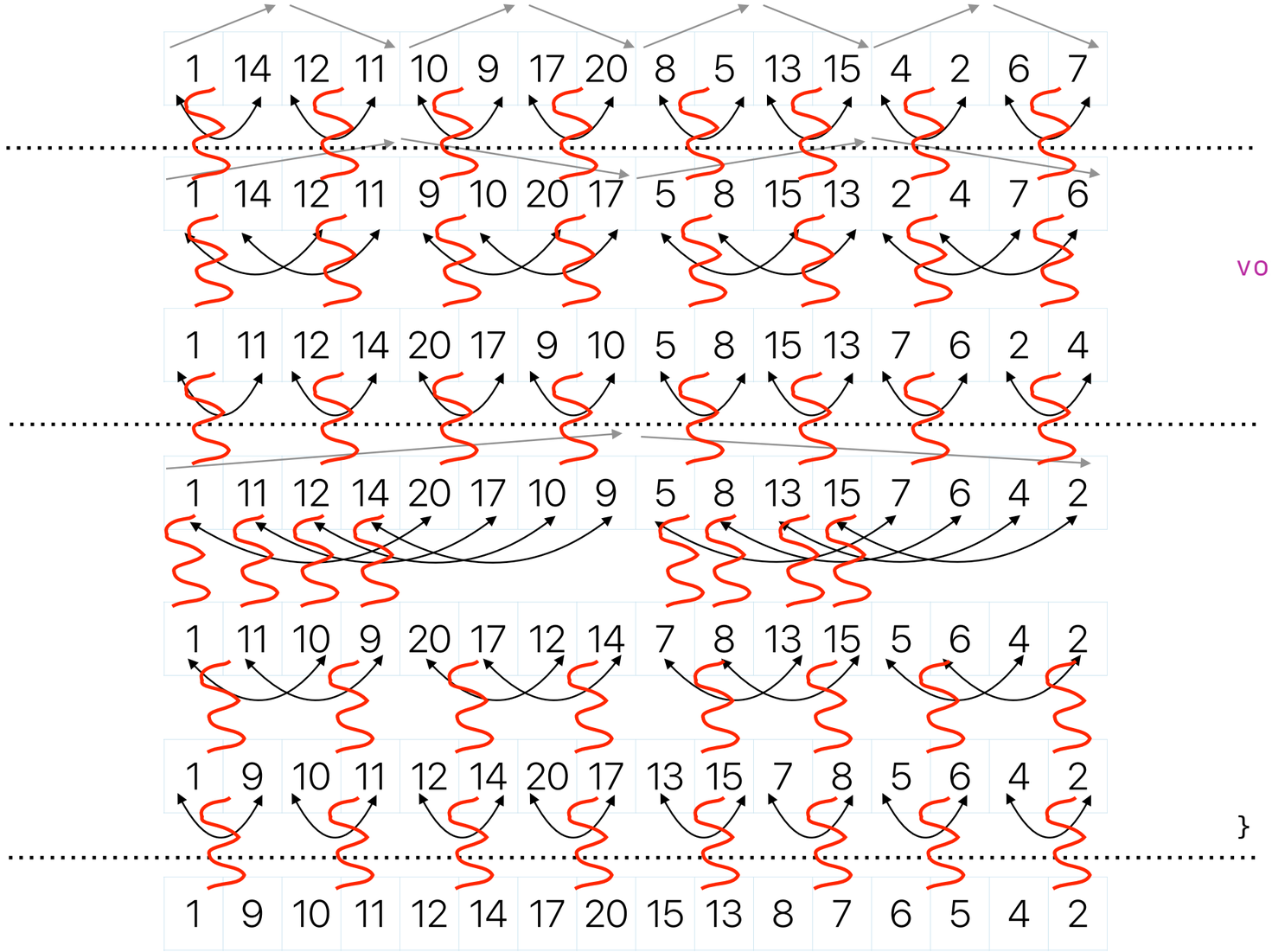


O(n log n)

Parallel merge sort

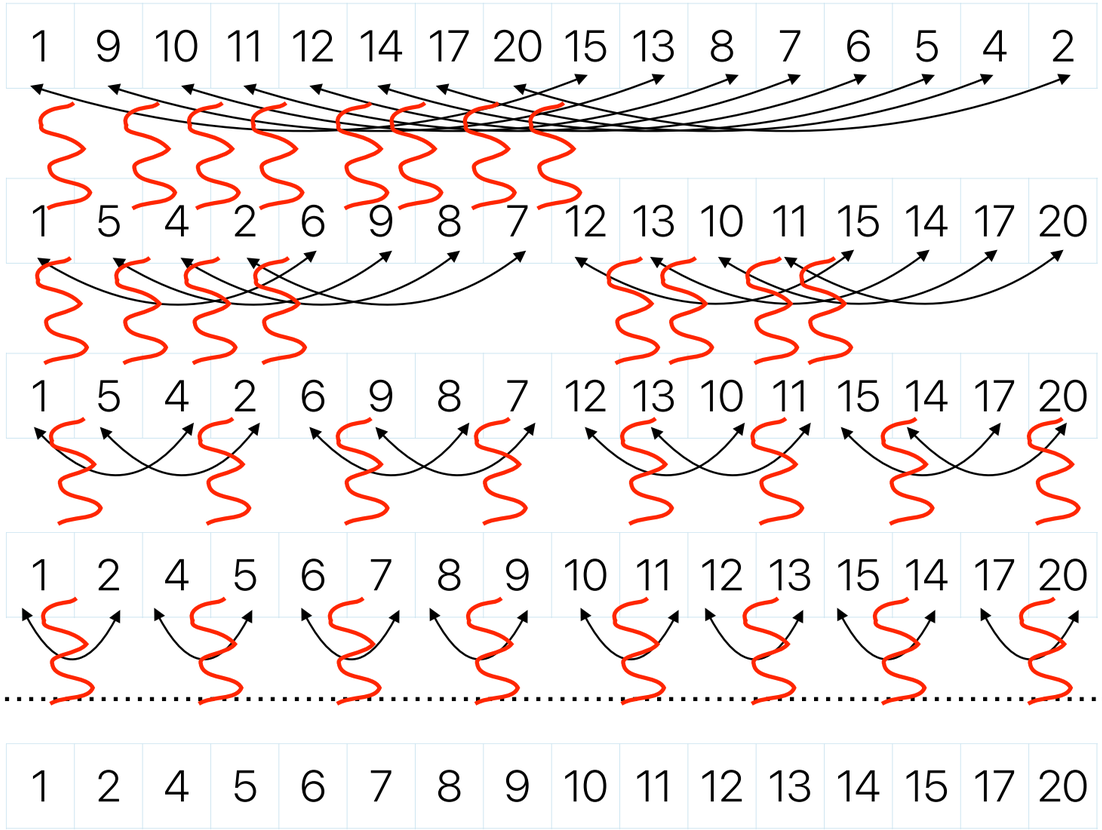


Bitonic sort




```
void BitonicSort() {

    int i,j,k;

    for (k=2; k<=N; k=2*k) {
        for (j=k>>1; j>0; j=j>>1) {
            for (i=0; i<N; i++) {
                int ij=i^j;
                if ((ij)>i) {
                    if ((i&k)==0 && a[i] > a[ij])
                        exchange(i,ij);
                    if ((i&k)!=0 && a[i] < a[ij])
                        exchange(i,ij);
                }
            }
        }
    }
}
```

Bitonic sort (cont.)




```
void BitonicSort() {

    int i,j,k;

    for (k=2; k<=N; k=2*k) {
        for (j=k>>1; j>0; j=j>>1) {
            for (i=0; i<N; i++) {
                int ij=i^j;
                if ((ij)>i) {
                    if ((i&k)==0 && a[i] > a[ij])
                        exchange(i,ij);
                    if ((i&k)!=0 && a[i] < a[ij])
                        exchange(i,ij);
                }
            }
        }
    }
}
```

benefits — in-place merge (no additional space is necessary), very stable comparison patterns

$O(n \log^2 n)$ — hard to beat $n(\log n)$ if you can't parallelize this a lot!

Corollary #4

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{\infty}}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

- If we can build a processor with unlimited parallelism
  - The complexity doesn't matter as long as the algorithm can utilize all parallelism
  - That's why bitonic sort or MapReduce works!
- **The future trend of software/application design is seeking for more parallelism rather than lower the computational complexity**

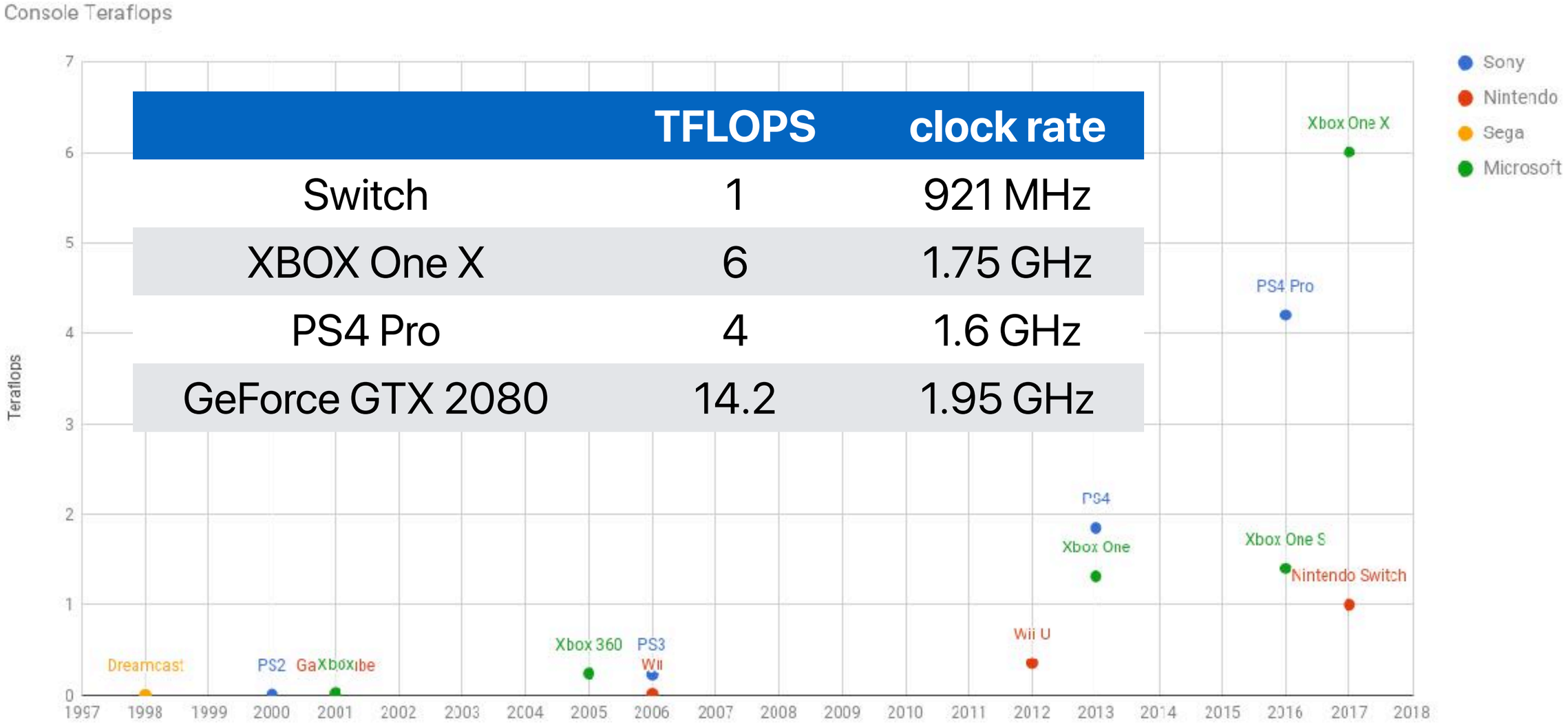
"Fair" Comparisons

Andrew Davison. Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers. In Humour the Computer, MITP, 1995

V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer. How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful. In IEEE Solid-State Circuits Magazine, vol. 12, no. 3, pp. 28-41, Summer 2020.

TFLOPS (Tera FLoating-point Operations Per Second)

	TFLOPS	clock rate
Switch	1	921 MHz
XBOX One X	6	1.75 GHz
PS4 Pro	4	1.6 GHz
GeForce GTX 2080	14.2	1.95 GHz



Is TFLOPS (Tera FLoating-point Operations Per Second) a good metric?

$$TFLOPS = \frac{\#\ of\ floating\ point\ instructions \times 10^{-12}}{Exection\ Time}$$

$$= \frac{IC \times \%\ of\ floating\ point\ instructions \times 10^{-12}}{IC \times CPI \times CT}$$

$$= \frac{\%\ of\ floating\ point\ instructions \times 10^{-12}}{CPI \times CT}$$

IC is gone!

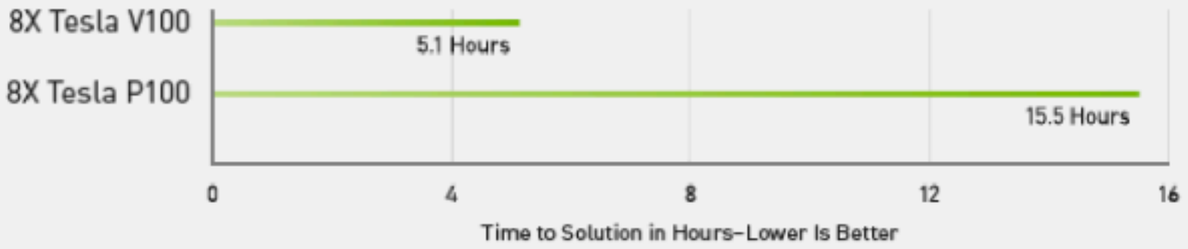
- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

TFLOPS (Tera FLoating-point Operations Per Second)

- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

	TFLOPS	clock rate
Switch	1	921 MHz
XBOX One X	6	1.75 GHz
PS4 Pro	4	1.6 GHz
GeForce GTX 2080	14.2	1.95 GHz

Deep Learning Training in Less Than a Workday



Server Config: Dual Xeon E5-2699 v4 2.6 GHz | 8X NVIDIA® Tesla® P100 or V100 | ResNet-50 Training on MXNet for 90 Epochs with 1.28M ImageNet Dataset.

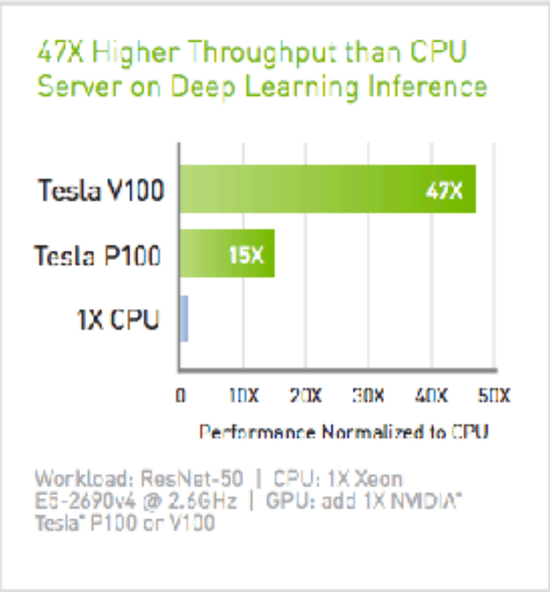
AI TRAINING

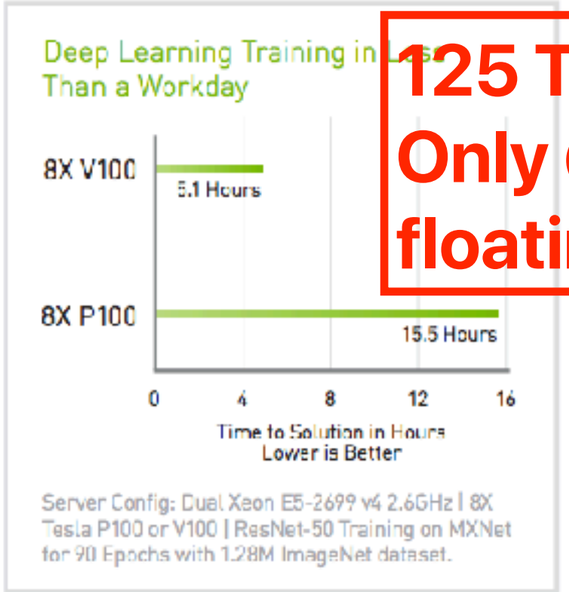
From recognizing speech to training virtual personal assistants and teaching autonomous cars to drive, data scientists are taking on increasingly complex challenges with AI. Solving these kinds of problems requires training deep learning models that are exponentially growing in complexity, in a practical amount of time.

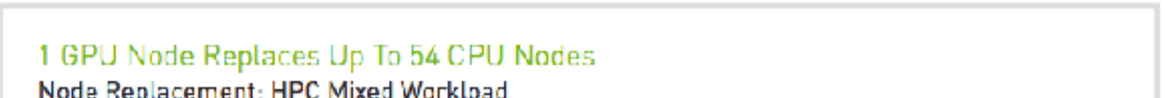
With 640 Tensor Cores, Tesla V100 is the world's first GPU to break the 100 teraFLOPS (TFLOPS) barrier of deep learning performance. The next generation of NVIDIA NVLink™ connects multiple V100 GPUs at up to 300 GB/s to create the world's most powerful computing servers. AI models that would consume weeks of computing resources on previous systems can now be trained in a few days. With this dramatic reduction in training time, a whole new world of problems will now be solvable with AI.

The Most Advanced Data Center GPU Ever Built.

NVIDIA® Tesla® V100 is the world's most advanced data center GPU ever built to accelerate AI, HPC, and graphics. Powered by NVIDIA Volta, the latest GPU architecture, Tesla V100 offers the performance of up to 100 CPUs in a single GPU—enabling data scientists, researchers, and engineers to tackle challenges that were once thought impossible.

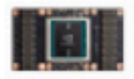






125 TFLOPS Only @ 16-bit floating point

SPECIFICATIONS



	Tesla V100 PCIe	Tesla V100 SXM2
GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS
GPU Memory	32GB /16GB HBM2	
Memory Bandwidth	900GB/sec	
ECC	Yes	
Interconnect Bandwidth	32GB/sec	300GB/sec
System Interface	PCIe Gen3	NVIDIA NVLink
Form Factor	PCIe Full Height/Length	SXM2
Max Power		

They try to tell it's the better AI hardware

https://blogs.nvidia.com/blog/2017/04/10/ai-drives-rise-accelerated-computing-datacenter/

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	1/13X	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16 MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

Inference per second

$$\frac{Inferences}{Second} = \frac{Inferences}{Operation} \times \frac{Operations}{Second}$$

$$= \frac{Inferences}{Operation} \times [\frac{operations}{cycle} \times \frac{cycles}{second} \times \#\_of\_PEs \times Utilization\_of\_PEs]$$

	Hardware	Model	Input Data
Operations per inference		v	
Operations per cycle	v		
Cycles per second	v		
Number of PEs	v		
Utilization of PEs	v	v	
Effectual operations out of (total) operations		v	v
Effectual operations plus unexploited ineffectual operations per cycle	v		

What's wrong with inferences per second?

- There is no standard on how they inference — but these affect!
  - What model?
  - What dataset?
- That's why Facebook is trying to promote an AI benchmark — MLPerf

- *Pitfall: For NN hardware, Inferences Per Second (IPS) is an inaccurate summary performance metric.*

Our results show that IPS is a poor overall performance summary for NN hardware, as it's simply the inverse of the complexity of the typical inference in the application (e.g., the number, size, and type of NN layers). For example, the TPU runs the 4-layer MLP1 at 360,000 IPS but the 89-layer CNN1 at only 4,700 IPS, so TPU IPS vary by 75X! Thus, using IPS as the single-speed summary is *even more misleading* for NN accelerators than MIPS or FLOPS are for regular processors [23], so IPS should be even more disparaged. To compare NN machines better, we need a benchmark suite written at a high-level to port it to the wide variety of NN architectures. Fathom is a promising new attempt at such a benchmark suite [3].

Poll close in 1:30

With MLPerf, are we good with inferences/second?

- The following table shows the inference/second using ImageNet dataset and ResNet-50 v1.5 model as well as the number of maximum concurrent "inferences" each machine can support. If we are targeting as making decisions for autonomous cars — requires a decision to be made within 25ms, which of the following architecture would work?

	Intel® Xeon® Platinum 9200 processors (CPU)	Google Cloud TPU v3 (TPU)	NVIDIA/Supermicro 4029GP-TRT-OTO-28 8xT4 (GPU)
Inferences per second	5,965.62	32,716.00	44,977.80
Cores	112 processors * 2-way SMT	2 MXU	320*8 MXU
Number of Maximum Parallel Inferencing Instances	224	2	320*8 = 2560

https://mlperf.org/inference-results/

A. CPU and TPU
B. TPU and GPU
C. Only GPU
D. Only TPU
E. All would work well

> operation. Ideally, the camera-to-recognition latency per frame should not substantially exceed the inter-frame time of the input images (e.g., 25 milliseconds for a 40 FPS camera).

https://www.cs.unc.edu/~anderson/papers/rtas19.pdf

Poll close in 1:30

With MLPerf, are we good with inferences/sec

- The following table shows the inference/second using ImageNet dataset and ResNet-50 v1.5 model as well as the number of maximum concurrent "inferences" each machine can support. If we are targeting as making decisions for autonomous cars — requires a decision to be made within 25ms, which of the following architecture would work?

	Intel® Xeon® Platinum 9200 processors (CPU)	Google Cloud TPU v3 (TPU)	NVIDIA/Supermicro 4029GP-TRT-OTO-28 8xT4 (GPU)
Inferences per second	5,965.62	32,716.00	44,977.80
Cores	112 processors * 2-way SMT	2 MXU	320*8 MXU
Number of Maximum Parallel Inferencing Instances	224	2	320*8 = 2560

https://mlperf.org/inference-results/

A. CPU and TPU
B. TPU and GPU
C. Only GPU
D. Only TPU
E. All would work well

> operation. Ideally, the camera-to-recognition latency per frame should not substantially exceed the inter-frame time of the input images (e.g., 25 milliseconds for a 40 FPS camera).

https://www.cs.unc.edu/~anderson/papers/rtas19.pdf

With MLPerf, are we good with inferences/second?

- The following table shows the inference/second using ImageNet dataset and ResNet-50 v1.5 model as well as the number of maximum concurrent "inferences" each machine can support. If we are targeting as making decisions for autonomous cars — requires a decision to be made within 25ms, which of the following architecture would work?

	Intel® Xeon® Platinum 9200 processors (CPU)	Google Cloud TPU v3 (TPU)	NVIDIA/Supermicro 4029GP-TRT-OTO-28 8xT4 (GPU)
Inferences per second	5,965.62	32,716.00	44,977.80
MXU	112 processors * 2-way SMT	2 MXU	320*8 MXU
Number of Maximum Parallel Inferencing Instances	224	2	320*8 = 2560

https://mlperf.org/inference-results/

A. CPU and TPU
B. TPU and GPU
C. Only GPU
D. Only TPU
E. All would work well

Batches/Sec $\frac{5965.62}{224} = 26.63$ $\frac{32716}{2} = 16358$ $\frac{44977.8}{2560} = 17.5694531$

Seconds/Batch $\frac{1}{26.63} = 37.55ms$ $\frac{1}{16358} = 611us$ $\frac{1}{17.5694531} = 56.91ms$

Choose the right metric — Latency v.s. Throughput/Bandwidth

Latency v.s. Bandwidth/Throughput

- Latency — the amount of time to finish an operation
  - Access time
  - Response time
- Throughput — the amount of work can be done within a given period of time
  - Bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
  - IOPs (I/O operations per second)
  - FLOPs (Floating-point operations per second)
  - IPS (Inferences per second)

With MLPerf, are we good with inferences/second?

- The following table shows the inference/second using ImageNet dataset and ResNet-50 v1.5 model as well as the number of maximum concurrent "inferences" each machine can support. If we are targeting as making decisions for autonomous cars — requires a decision to be made within 100ms, which of the following architecture would work?

Latency sensitive

	Intel® Xeon® Platinum 9200 processors (CPU)	Google Cloud TPU v3 (TPU)	NVIDIA/Supermicro 4029GP-TRT-OTO-28 8xT4 (GPU)
Inferences per second	5,965.62	32,716.00	44,977.80
MXU		128*128*2	4*4*320*8
Number of Maximum Parallel Inferencing Instances	224	128*2 = 256	4*320*8 = 10240

Bandwidth

https://mlperf.org/inference-results/

A. CPU and TPU
B. TPU and GPU
C. Only GPU
D. Only TPU
E. All would work well

Batches/Sec

$$\frac{5965.62}{224} = 26.63 \qquad \frac{32716}{256} = 128 \qquad \frac{44977.8}{10240} = 4.39$$

Seconds/Batch

$$\frac{1}{26.63} = 37.55ms \qquad \frac{1}{128} = 7.81ms \qquad \frac{1}{128} = 227.79ms$$

Latency/Delay v.s. Throughput

	Toyota Prius	100 Gb Network
	•100 miles (161 km) from UCSD •75 MPH on highway! •Max load: 374 kg = 2,770 hard drives (2TB per drive)	•100 miles (161 km) from UCSD •Lightspeed! — $3*10^8$m/sec •Max load:4 lanes operating at 25GHz
bandwidth	290GB/sec	100 Gb/s or 12.5GB/sec
total latency	3.5 hours	2 Peta-byte over 167772 seconds = 1.94 Days
latency in getting the first moivie	You see nothing in the first 3.5 hours	100GB/100Gb = 8 secs! You can start watching the first movie in 8 secs!

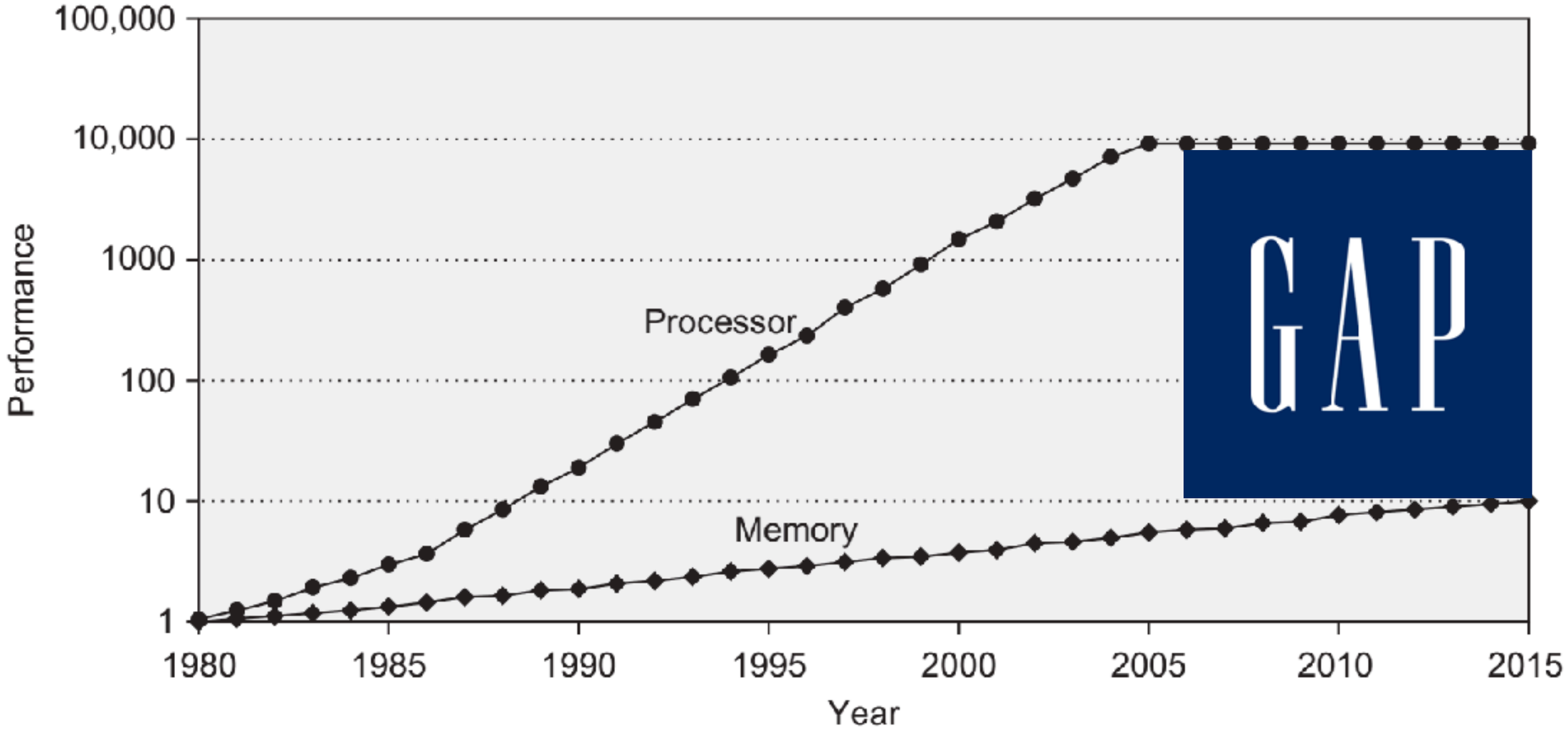
12 ways to Fool the Masses When Giving Performance Results on Parallel Computers

- Quote only 32-bit performance results, not 64-bit results.
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
- Quietly employ assembly code and other low-level language constructs.
- Scale up the problem size with the number of processors, but omit any mention of this fact.
- Quote performance results projected to a full system.
- Compare your results against scalar, unoptimized code on Crays.
- When direct run time comparisons are required, compare with an old code on an obsolete system.
- If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation.
- Quote performance in terms of processor utilization, parallel speedups or MFLOPS per dollar.
- Mutilate the algorithm used in the parallel implementation to match the architecture.
- Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.
- If all else fails, show pretty pictures and animated videos, and don't talk about performance.

Memory Hierarchy

Hung-Wei Tseng

Performance gap between Processor/Memory



Performance of modern DRAM

			Best case access time (no precharge)			Precharge needed
Production year	Chip size	DRAM type	RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G bit	DDR4	13	13	26	39

Figure 2.4 Capacity and access times for DDR SDRAMs by year of production. Access time is for a random memory word and assumes a new row must be opened. If the row is in a different bank, we assume the bank is precharged; if the row is not open, then a precharge is required, and the access time is longer. As the number of banks has increased, the ability to hide the precharge time has also increased. DDR4 SDRAMs were initially expected in 2014, but did not begin production until early 2016.

Poll close in 1:30

The impact of “slow” memory

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has “perfect” memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What’s the average CPI (pick the most close one)?
  A. 9
  B. 17
  C. 27
  D. 35
  E. 69

Poll close in 1:30

The impact of “slow” memory

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has “perfect” memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What’s the average CPI (pick the most close one)?
  A. 9
  B. 17
  C. 27
  D. 35
  E. 69

The impact of "slow" memory

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What's the average CPI (pick the most close one)?

  A. 9
  B. 17
  C. 27
  D. 35
  E. 69

$$1 + 100\% \times (52) + 30\% \times 52 = 68.6\ cycles$$


Q & A

Announcement

- Assignment #1 due 10/19
  - Assignments SHOULD BE done individually — but if you discussed with others, make sure you put their names on your submission
  - We will drop your least performing assignment as well
  - Attendance counts as one assignment
- Reading quiz due next Wednesday before the lecture
  - We will drop two of your least performing reading quizzes
  - You have two shots, both unlimited time
- Office Hours on Zoom (the office hour link, not the lecture one)
  - Walk-in, no appointment is necessary
  - Hung-Wei/Prof. Usagi: M 8p-9p, W 2p-3p
  - Quan Fan: F 1p-3p

Computer
Science &
Engineering

203

つづく