

Performance (3): (Don't Be A) Great Pretender

Hung-Wei Tseng

N SERIES

GREAT PRETENDER

Great Pretender

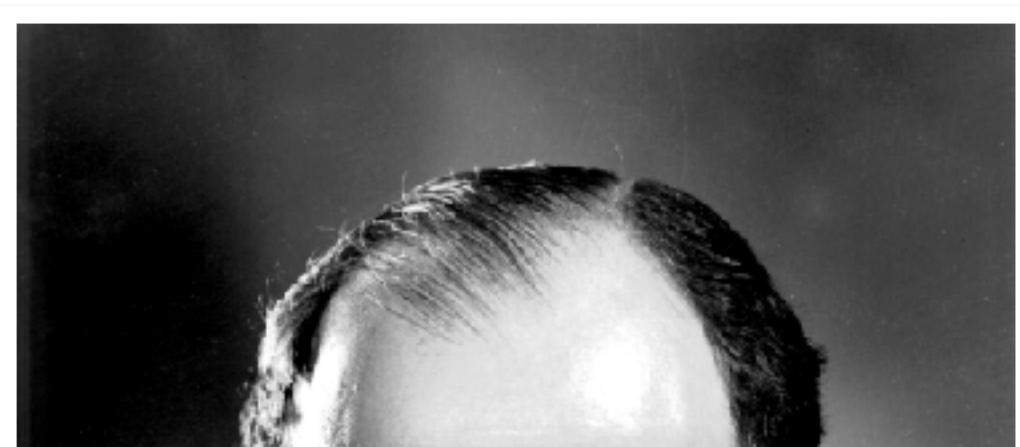
2020 | **TV-MA** | 1 Season | Drama Anime

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.

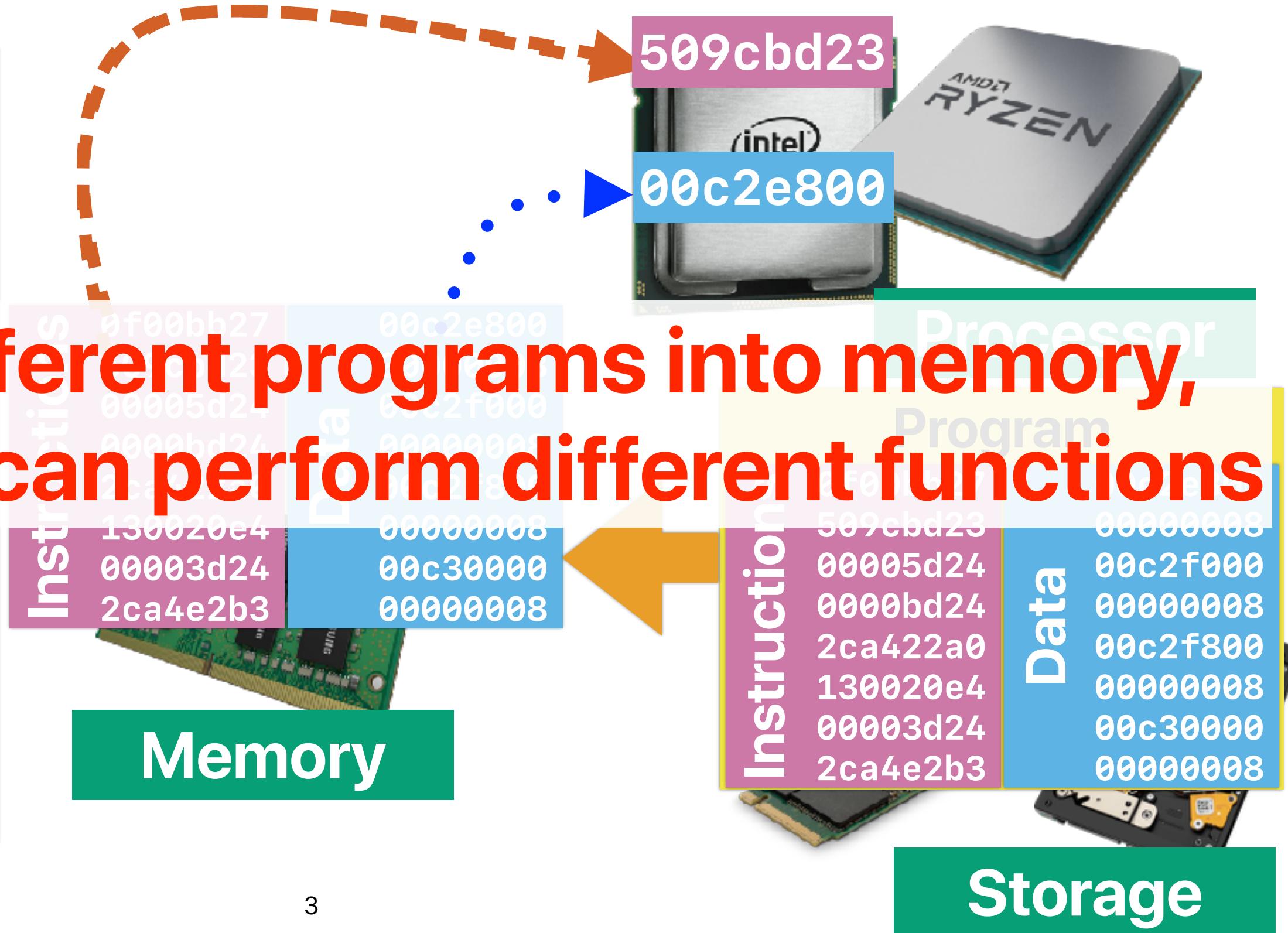
Starring: Chiaki Kobayashi, Junichi Suwabe, Natsumi Fujiwara



Recap: von Neumann Architecture



By loading different programs into memory,
your computer can perform different functions



Summary of CPU Performance Equation

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

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

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

- 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**

Lessons learned from Amdahl's Law

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

- Corollary #1: Maximum speedup
- Corollary #2: Make the common case fast
 - Common case changes all the time

$$\begin{aligned} Speedup_{max}(f, \infty) &= \frac{1}{(1 - f)} \\ Speedup_{max}(f_1, \infty) &= \frac{1}{(1 - f_1)} \\ Speedup_{max}(f_2, \infty) &= \frac{1}{(1 - f_2)} \\ Speedup_{max}(f_3, \infty) &= \frac{1}{(1 - f_3)} \\ Speedup_{max}(f_4, \infty) &= \frac{1}{(1 - f_4)} \end{aligned}$$

Have you ever been fooled by commercials? How and why?

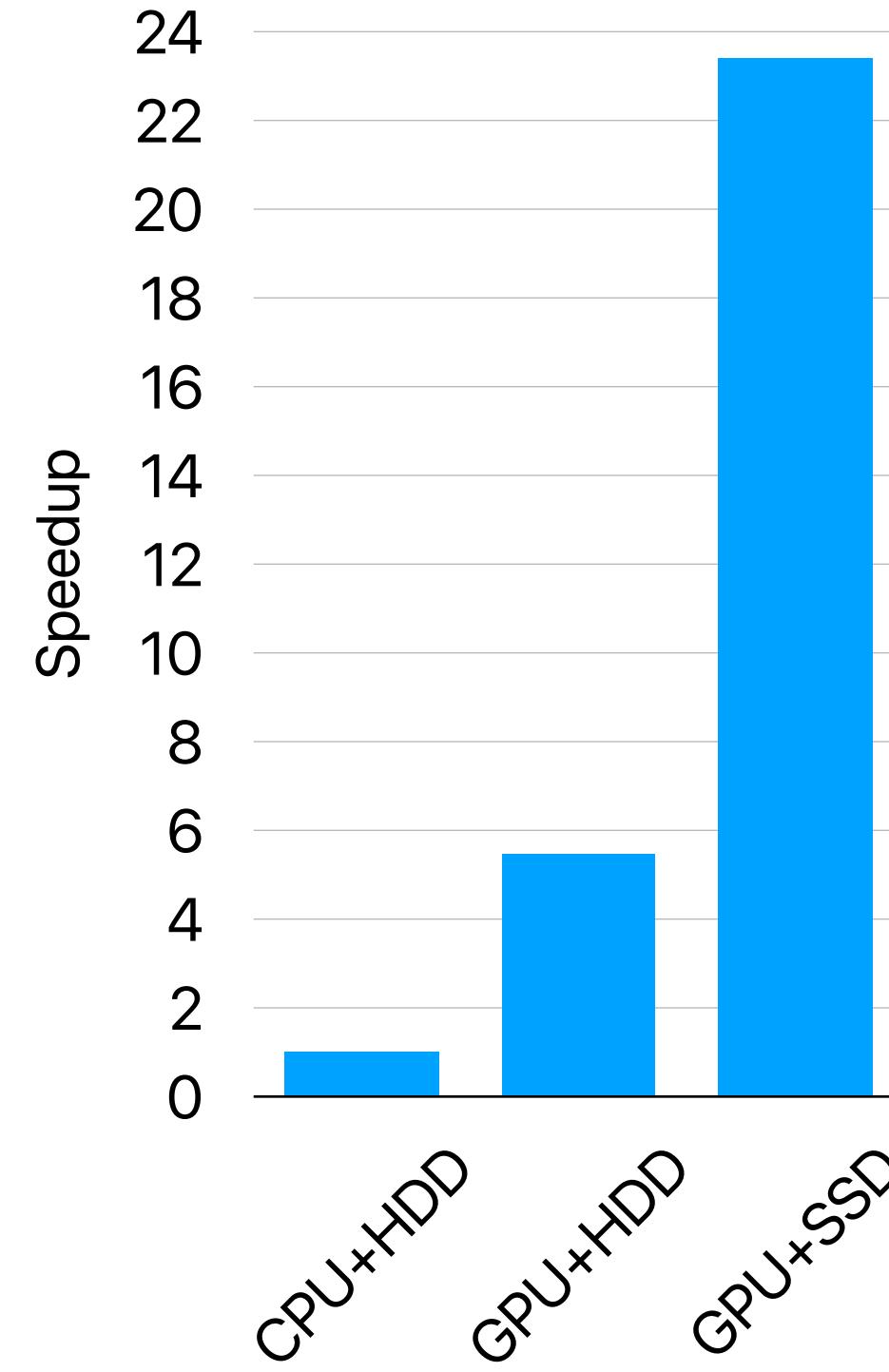
Outline

- More about Amdahl's Law
- Fair evaluations

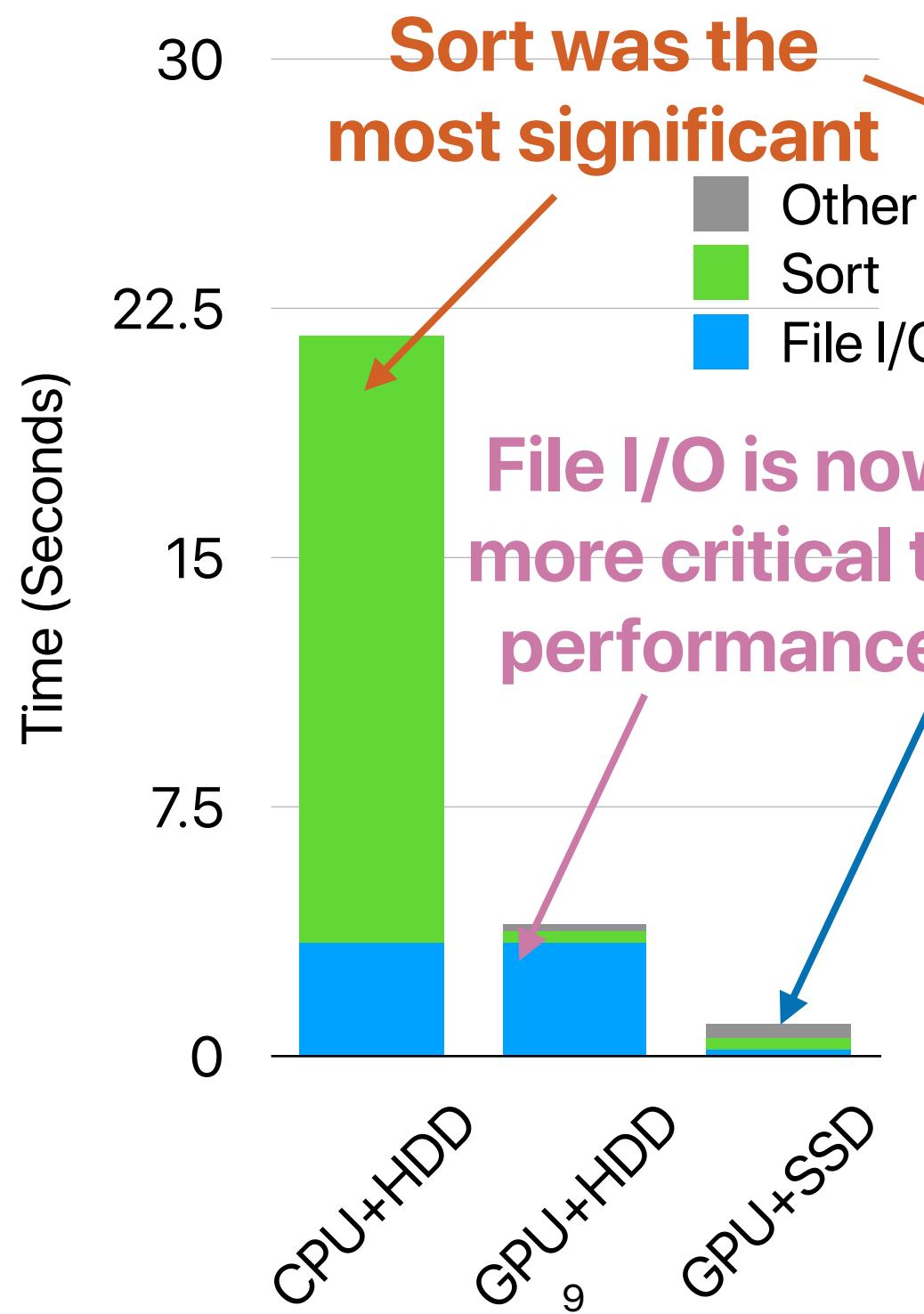
Demo — sort

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

Speedup



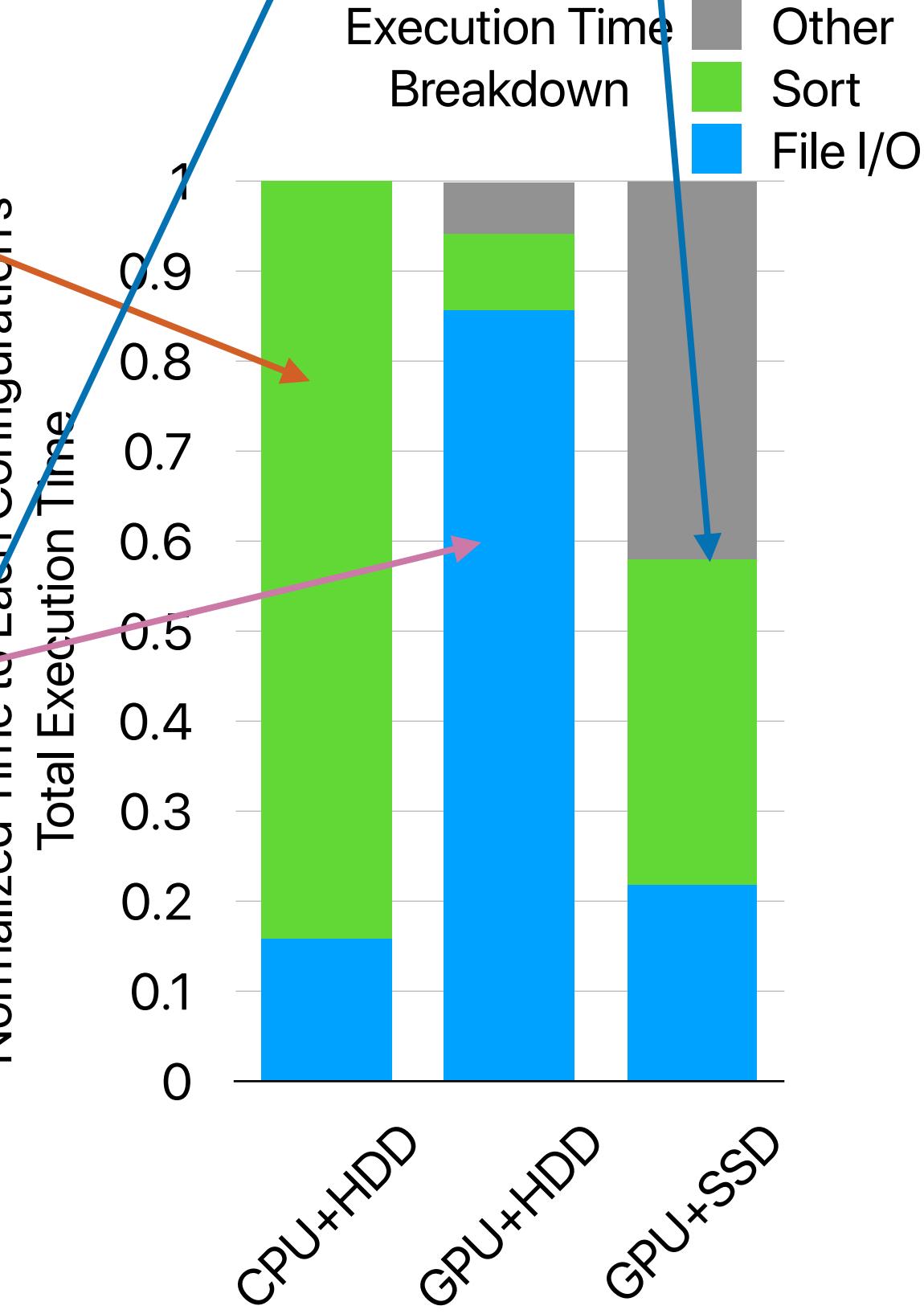
Cumulative Execution Time



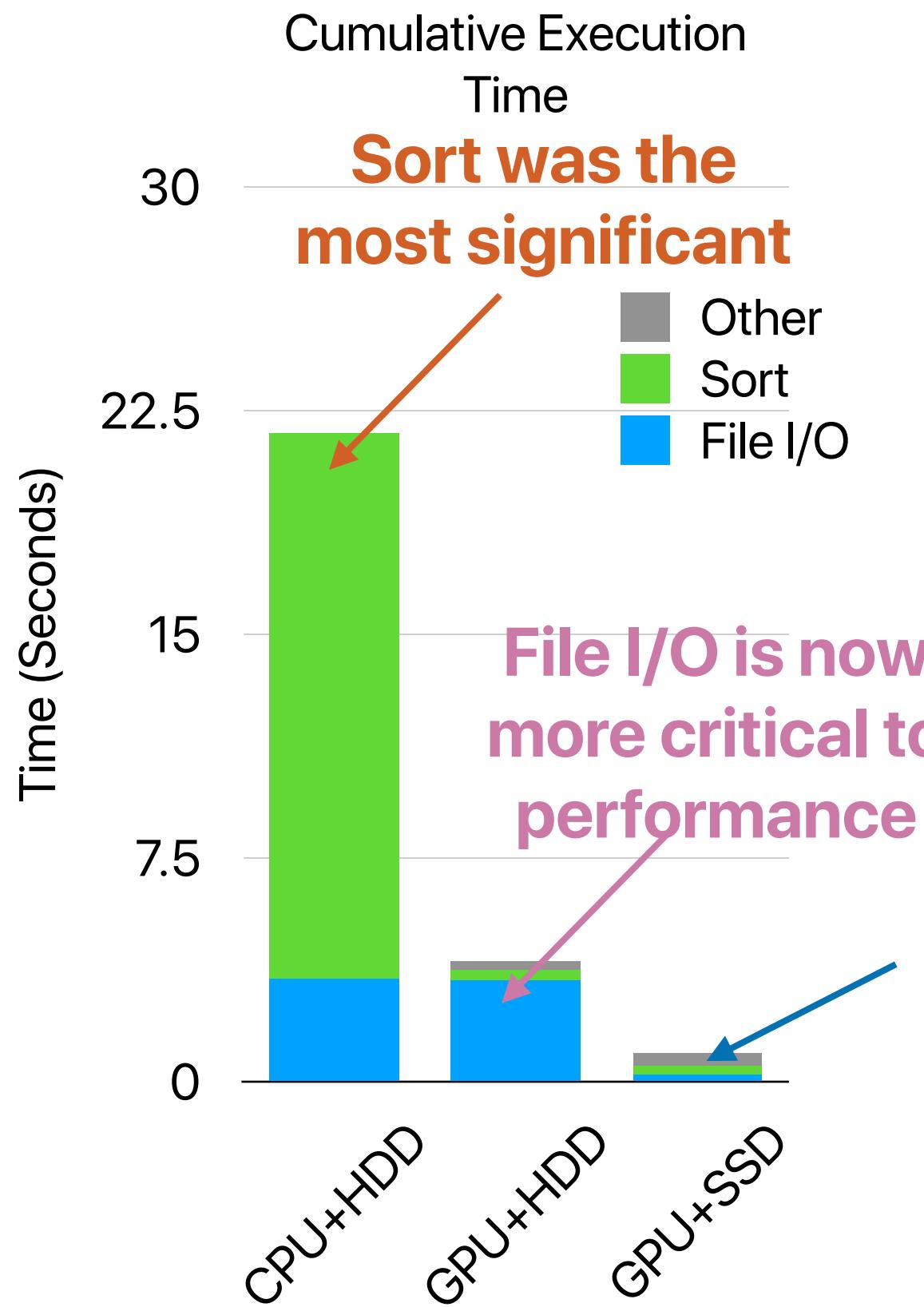
Sort was the most significant

File I/O is now more critical to performance

Normalized Time to Each Configuration's Total Execution Time



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

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

Amdahl's Law on Multicore Architectures

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

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, n) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{n}}$$



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 executing each parallel partition 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?

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}} \times \text{Speedup} < 1}{\infty}}$$

- If we have unlimited parallelism, the performance of executing each parallel partition 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
 - 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

Demo — merge sort v.s. bitonic sort on GPUs

Merge Sort

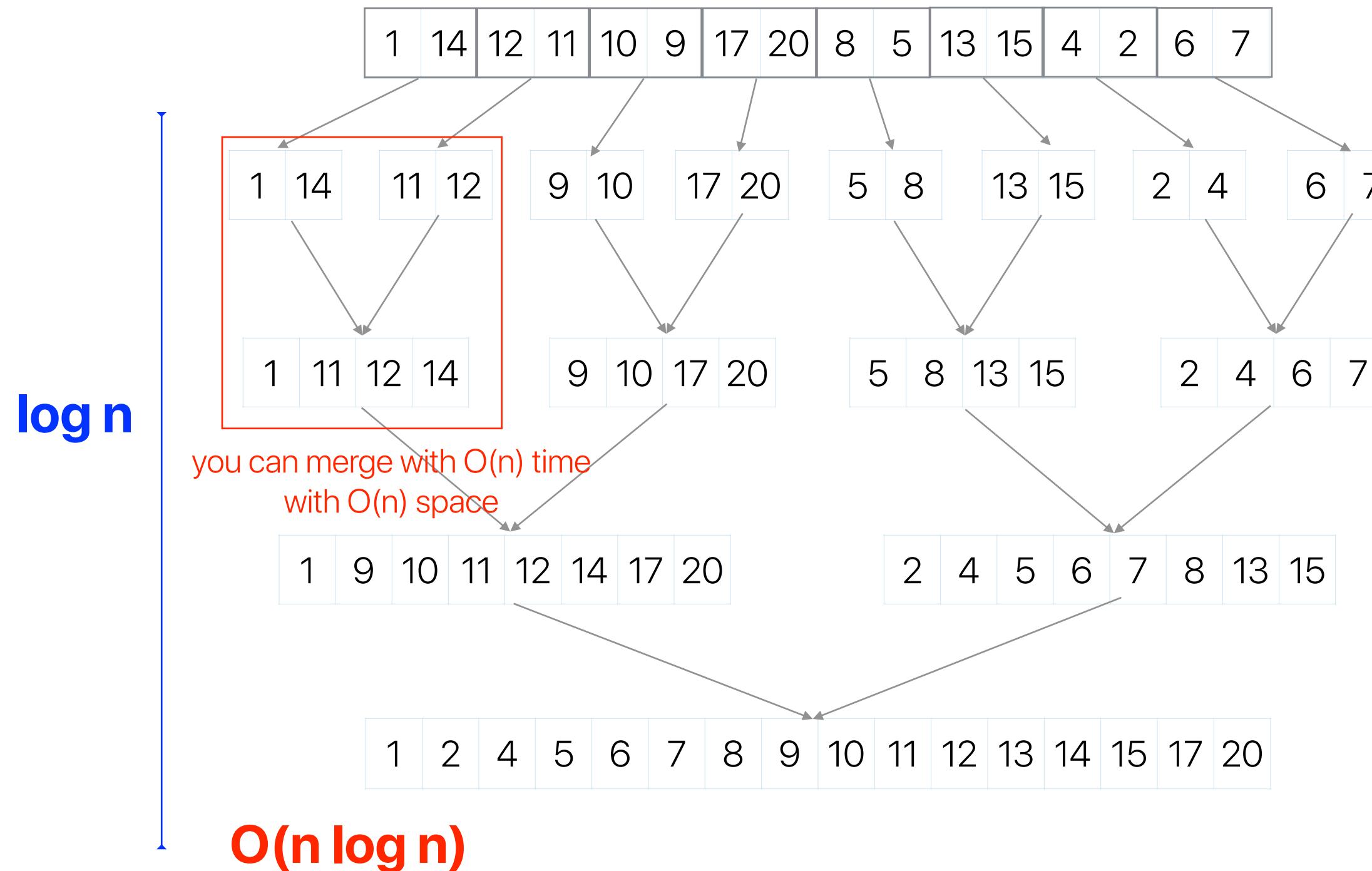
$$O(n \log_2 n)$$

Bitonic Sort

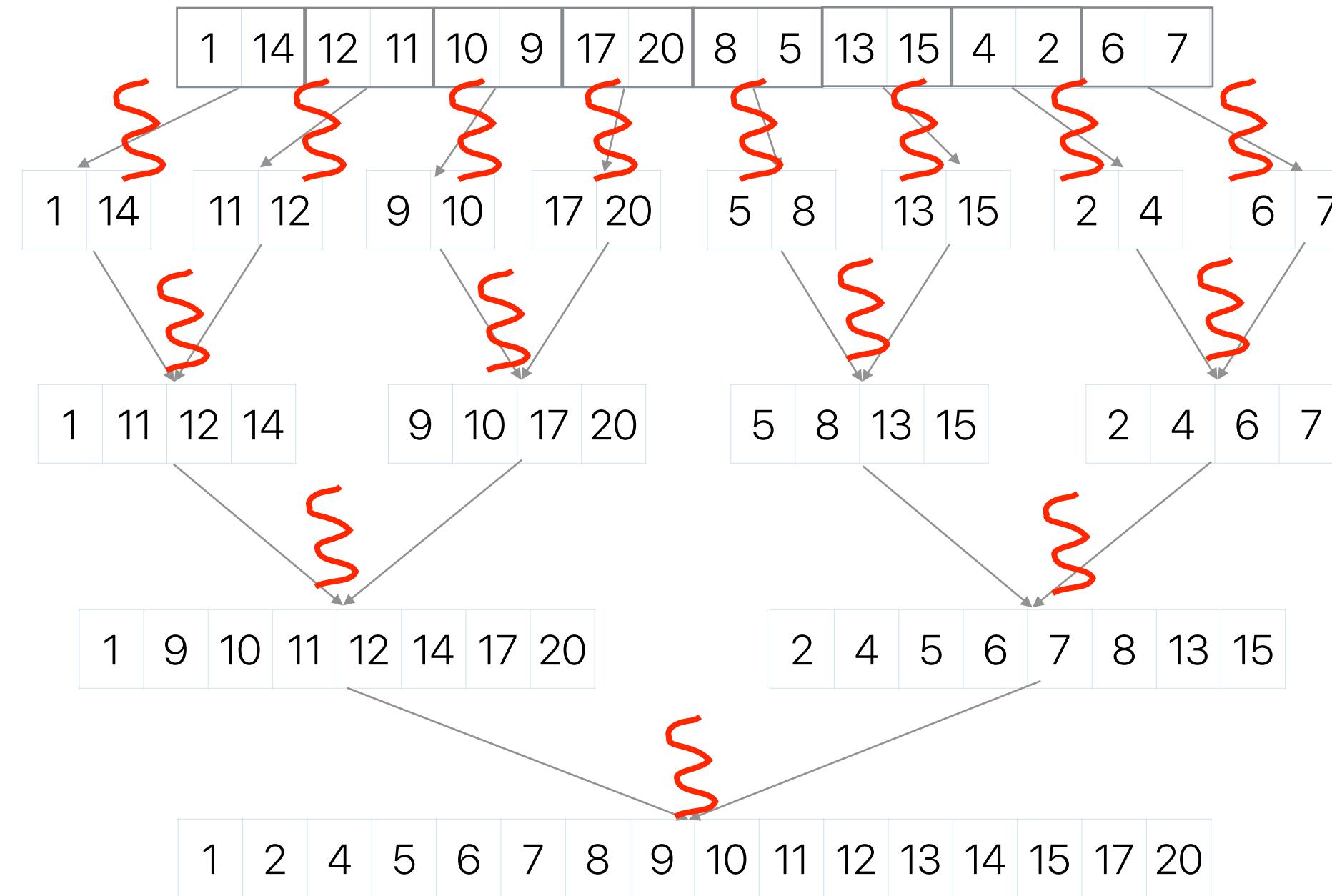
$$O(n \log_2^2 n)$$

```
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);  
                }  
            }  
        }  
    }  
}
```

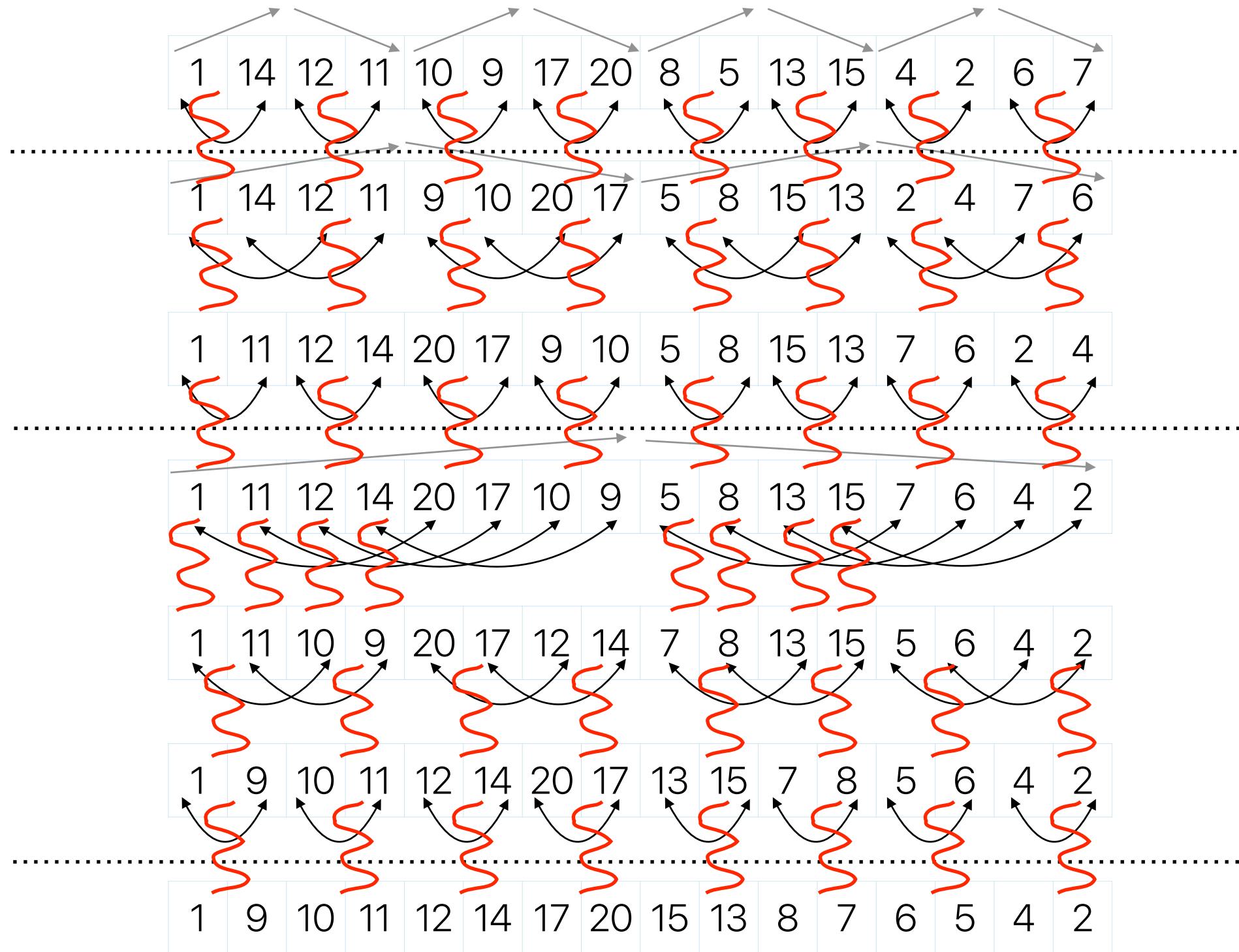
Merge sort



Parallel merge sort

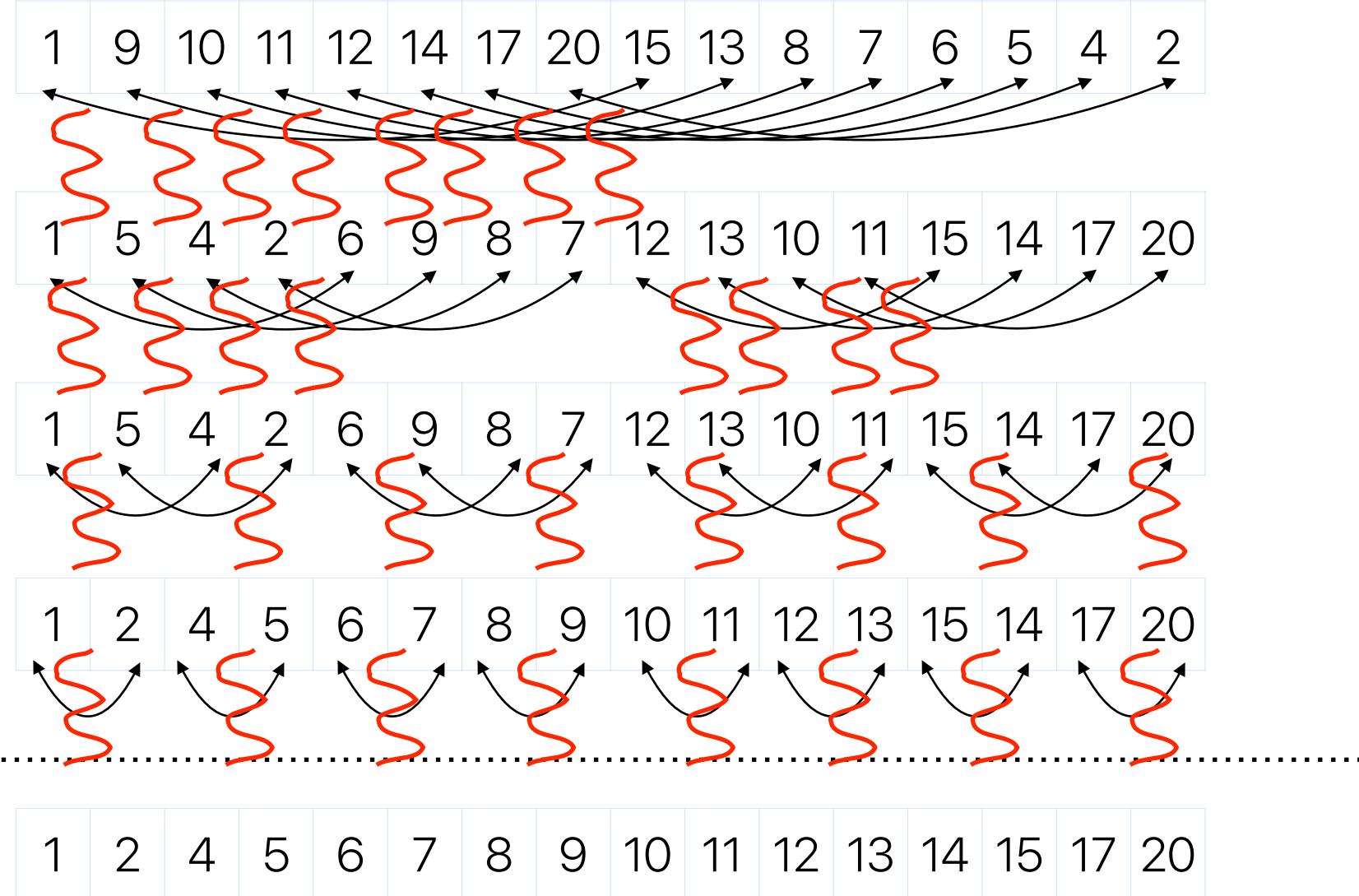


Bitonic sort



```
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

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{\infty}}$$

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}})}$$

- 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**

**Is it the end of computational
complexity?**

Corollary #5

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}}) + \frac{f_{\text{parallelizable}}}{\infty}}$$

$$\text{Speedup}_{\text{parallel}}(f_{\text{parallelizable}}, \infty) = \frac{1}{(1 - f_{\text{parallelizable}})}$$

- 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
 - Algorithm complexity still gives some “insights” regarding the growth of execution time in the same algorithm, though still not accurate

However, parallelism is not “tax-free”

However, parallelism is not “tax-free”

- Synchronization
- Preparing data
- Addition function calls
- Data exchange if the parallel hardware has its own memory hierarchy



Amdahl's Law considering overhead

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

- r is some other parameter that affects the overhead
 - input size?
 - degree of parallelism?
- The overhead may scale differently than the original problem
 - that's why we introduce "perf()" function

Corollary #6: Don't hurt non-common part too much

- If the program spend 90% in A, 10% in B. Assume that an optimization can accelerate A by 9x, by hurts B by 10x (i.e., an overhead that is 9x longer than the original execution time)...

$$\text{Speedup} = \frac{1}{(1-f) + \text{perf}(r) + \frac{f}{s}} = \frac{1}{(1-0.9) + (1-0.9) \times 9 + \frac{0.9}{9}} = 0.91 \times$$

Lessons learned from Amdahl's Law

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

- Corollary #1: Maximum speedup
- Corollary #2: Make the common case fast
 - Common case changes all the time
- Corollary #3: Optimization is a moving target
- Corollary #4: Exploiting more parallelism from a program is the key to performance gain in modern architectures
- Corollary #5: Single-core performance still matters
- Corollary #6: Don't hurt the non-common case too much

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

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

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

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

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

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

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

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

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

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.

Latency v.s. Bandwidth/Throughput

- Latency — the amount of time to finish an operation
 - End-to-end execution time of “something”
 - Access time
 - Response time
- Throughput — the amount of work can be done within a given period of time (typically “something” per “timeframe” or the other way around)
 - Bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
 - IOPs (I/O operations per second)
 - FLOPs (Floating-point operations per second)
 - IPS (Inferences per second)

Demo: matmul on GPU

Size	Latency	Relative Latency	Throughput (Output Numbers Per Second)	Relative Throughput
16x16x16	~ 0.09ms	1	0.09ms/256	1
32x32x32	~ 0.09ms	1	0.09ms/1024	4
64x64x64	~ 0.09ms	1	0.09ms/4096	16

Larger throughput doesn't mean shorter latency!



Artificial Intelligence Computing Leadership from NVIDIA

CLOUD & DATA CENTER

PRODUCTS ▾

SOLUTIONS ▾

APPS ▾

FOR DEVELOPERS

TECHNOLOGIES ▾

Tesla V100

AI TRAINING

AI INFERENCE

HPC

DATA CENTER GPUs

SPECIFICATIONS

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.

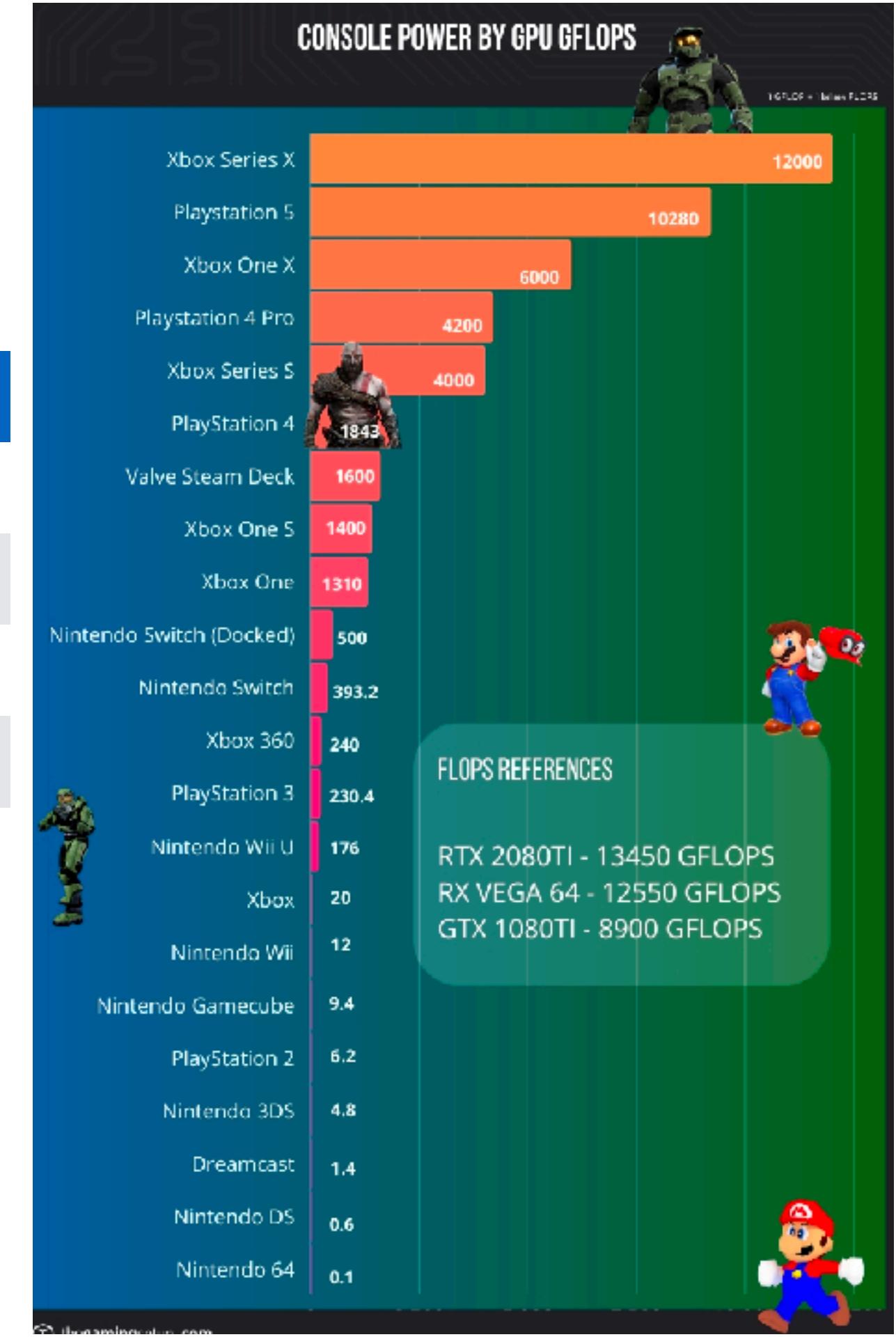
AI TRAINING

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.

TFLOPS (Tera FLoating-point Operations Per Second)

	TFLOPS	clock rate
Switch	1	921 MHz
PS5	10.28	2.23 GHz
XBox Series X	12	1.825 GHz
GeForce RTX 3090	40	1.395 GHz



TFLOPS (Tera FLoating-point Operations Per Second)

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

Let's measure the FLOPS of matrix multiplications

```
for(i = 0; i < ARRAY_SIZE; i++) {  
    for(j = 0; j < ARRAY_SIZE; j++) {  
        for(k = 0; k < ARRAY_SIZE; k++) {  
            c[i][j] += a[i][k]*b[k][j];  
        }  
    }  
}
```

Floating point operations per second (FLOP"S"):

Floating point operations (FLOP"s"):

$$i \times j \times k \times 2$$

Given $i = j = k = 2048$

$$2^{3 \times 11} \times 2 = 2^{34} \text{ FLOPs in total}$$

$$FLOPS = \frac{2^{34}}{ET_{seconds}}$$



How reflective is FLOPS?

- Given the FLOPS number measured, how many of the followings are true?
 - The FLOPS number remains the same on each architecture if we change the data size
 - The FLOPS number remains the same on each architecture if we change the data type to double
 - The FLOPS number remains the same on each architecture if we change the algorithm implementation
 - The FLOPS number reflects the performance ratio of different architectures when executing floating point applications
- A. 0
B. 1
C. 2
D. 3
E. 4

How reflective is FLOPS?

- Given the FLOPS number measured, how many of the followings are true?
 - The FLOPS number remains the same on each architecture if we change the data size
 - The FLOPS number remains the same on each architecture if we change the data type to double
 - The FLOPS number remains the same on each architecture if we change the algorithm implementation
 - The FLOPS number reflects the performance ratio of different architectures when executing floating point applications
- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

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

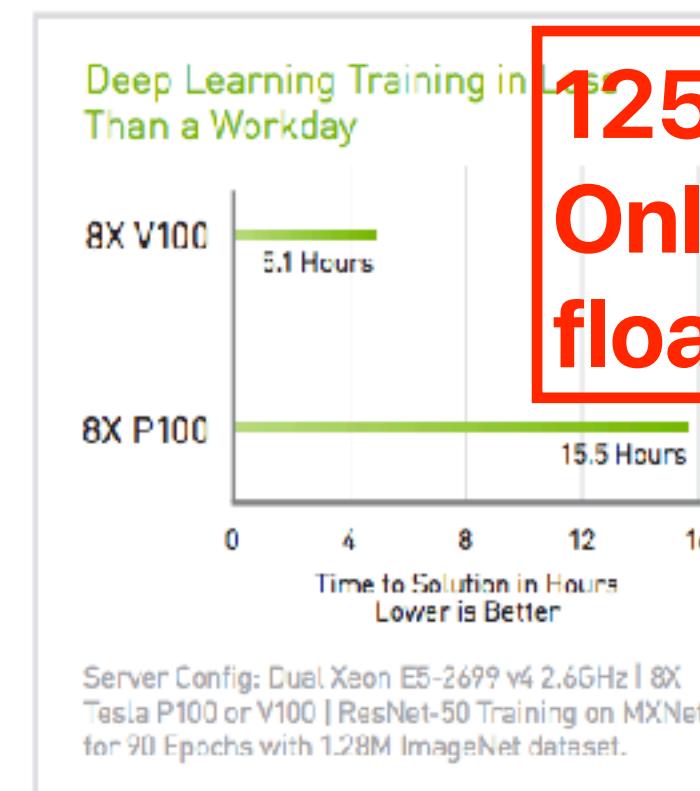
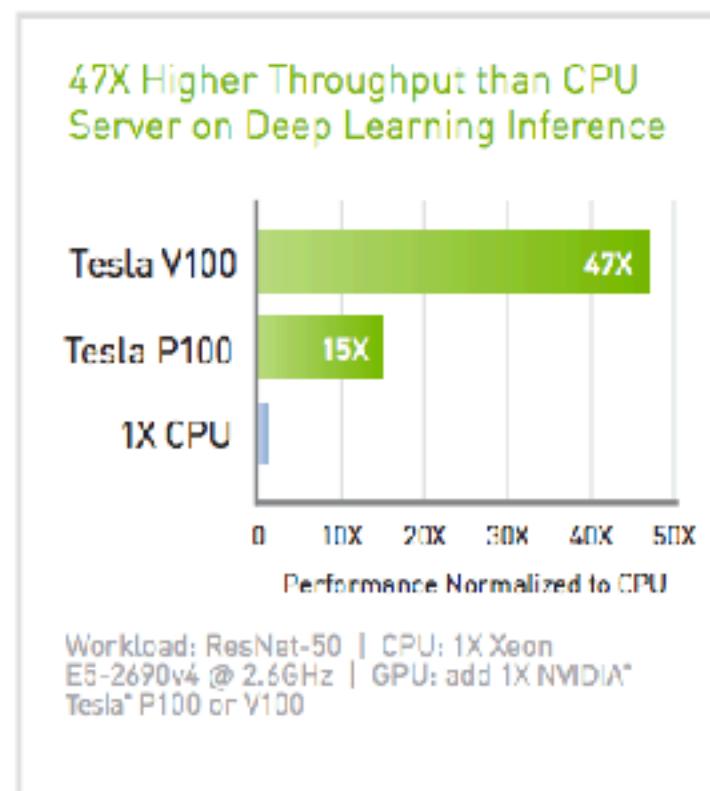
$$\begin{aligned}TFLOPS &= \frac{\# \text{ of floating point instructions} \times 10^{-12}}{\text{Execution Time}} \\&= \frac{IC \times \% \text{ of floating point instructions} \times 10^{-12}}{IC \times CPI \times CT} \\&= \frac{\% \text{ of floating point instructions} \times 10^{-12}}{CPI \times CT}\end{aligned}$$

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

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



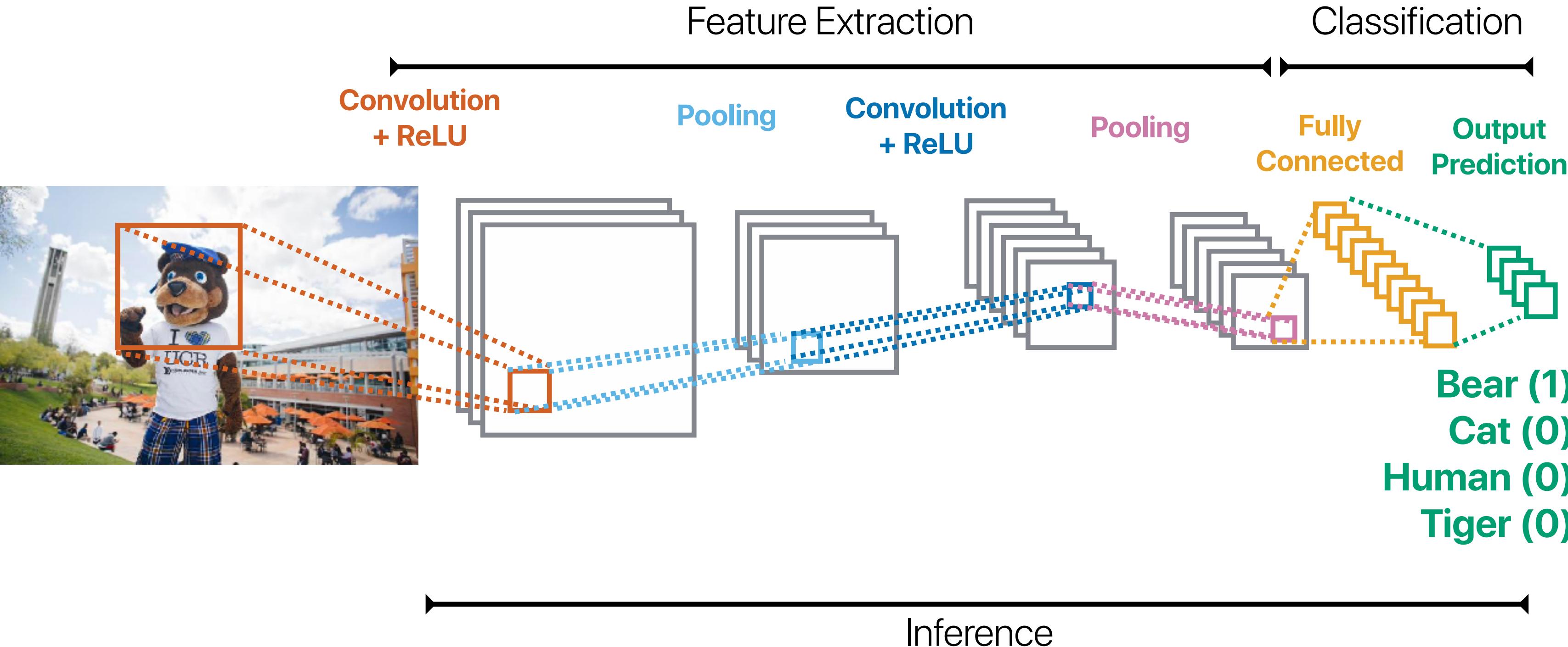
Tesla V100
SXM2

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	375W	300W

1 GPU Node Replaces Up To 54 CPU Nodes

Node Replacement: HPC Mixed Workload

The Machine Learning Inference Pipeline



Bard v.s. ChatGPT Round #1

Bard

Who is Hung-Wei Tseng?

Hung-Wei Tseng is an associate professor in the Department of Electrical and Computer Engineering and a cooperating faculty of the Department of Computer Science and Engineering at University of California, Riverside. He is now leading the Extreme Storage & Computer Architecture Laboratory. He is interested in diverse research topics that allow applications or programmers to more efficiently use modern heterogeneous hardware components. Together with his students, they have demonstrated how intelligent storage devices can help improve performance, power and energy for data-intensive systems.

Their work has been recognized by IEEE Micro "Top Picks from the 2019 Computer Architecture Conferences" (IEEE MICRO Top Picks 2020) and Facebook Research Award, 2018. Prior to joining UCR, he served as an assistant professor for the Department of Computer Science and the Department of Electrical and Computer Engineering at NC State University. He was a PostDoc of the Non-volatile Systems Laboratory and a lecturer of the Department of Computer Science and Engineering at University of California, San Diego with Professor Steven Swanson.

Sources - Learn more
1. ictra.ugr.es/~htseng/
2. www.nvsl.io/portfolio/hung-wei-tseng/

Like Share Google It

How important is Computer Architecture in Computer Science? ▶

1 answer(inference)/6 secs



I am unable to determine who Hung-Wei Tseng is. Can you please provide me with more context or details about the individual in question?

HT Who is Hung-Wei Tseng at UCR?

Hung-Wei Tseng is an Associate Professor of Computer Science and Engineering at the University of California, Riverside (UCR). His research interests are in the area of computer networks and distributed systems, with a focus on improving network performance and security. Some of his recent research work has focused on topics such as network measurement and monitoring, network optimization, and network security. Hung-Wei Tseng has been a faculty member at UCR since 2013, and he has received several awards and recognitions for his research contributions.

Regenerate response

How important is Computer Architecture in Computer Science? ▶

1 answer(inference)/18 secs

Inference per second is intensively used

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

	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

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

Bard v.s. ChatGPT Round #2

Bard



The screenshot shows the Bard interface with a sidebar on the left containing links like 'Reset chat', 'Bard Activity', 'FAQ', and 'Help & support'. The main area displays a card for 'Who is teaching CS203 at UCR?'. It includes a profile picture of Hung-Wei Tseng, a brief bio stating he is an assistant professor teaching CS203 at UCR, and contact information: Room 406, Winston Cheng Hall, 900 University Ave., Riverside, CA 92521; phone (951) 827-4222 ext 28047; and email htseng@ucr.edu. Below the card are social sharing icons for LinkedIn, Facebook, Twitter, and Google+.

1 answer/6 secs

The screenshot shows the ChatGPT interface with a sidebar on the left containing links like '+ New chat', 'UCR CS203 Instructor', 'Hung-Wei Tseng's Achievements', 'HW Memory Adoption Best', 'Clear conversations', 'Upgrade to Plus', 'Dark mode', 'Updates & FAQ', and 'Logout'. The main area displays a card for 'Who is teaching CS203 at UCR?'. It includes a profile picture of Hung-Wei Tseng, a brief bio stating he is an AI language model and does not have access to real-time information about who is teaching CS203 at UCR, and a note to check UCR's official website or contact the university's department of computer science directly for the most up-to-date information on course instructors.

1 answer/6 secs

What's (not) considered by Inference per second

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

$$= \frac{\text{Inferences}}{\text{Operation}} \times \left[\frac{\text{operations}}{\text{cycle}} \times \frac{\text{cycles}}{\text{second}} \times \#_{\text{of_PEs}} \times \text{Utilization_of_PEs} \right]$$

	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?
 - Quality?

“Fair” Comparisons

Andrew Davison. Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers.
In Humour the Computer, MITP, 1995
Norman P. Jouppi et. al.. In-Datacenter Performance Analysis of a Tensor Processing Unit
<https://doi.org/10.1145/3079856.3080246>



Extreme Multitasking Performance

- Dual 4K external monitors
- 1080p device display
- 7 applications

What's missing in this video clip?

- The ISA of the “competitor”
- Clock rate, CPU architecture, cache size, how many cores
- How big the RAM?
- How fast the disk?

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.

Fair comparison in computer architectures

- Metrics: you must consider the fact that performance is composed of IC, CPI, and CT. — any metric that misses one of them is misleading
- Only one variation in each comparison
 - Only change the processor, but not ISA (related to IC) and others
 - Only change the algorithm, but not others
 - The same dataset, must be the same outcome

Why we need benchmark suites

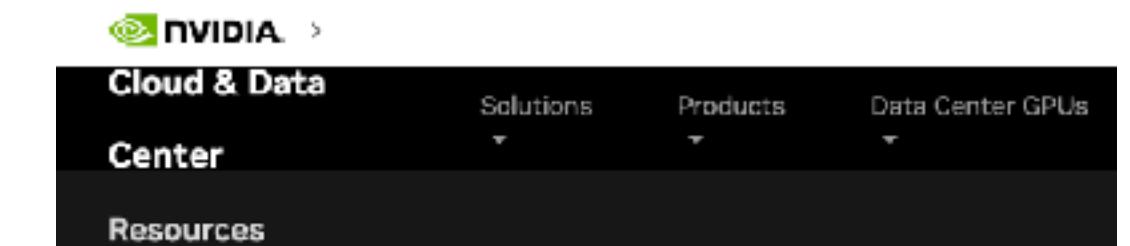
- Allowing people evaluate systems with exactly the same program and the same inputs and validate results from different machines
- Popular benchmark suites
 - SPEC — CPU benchmark
 - MLPerf — ML systems



The screenshot shows the homepage of the SPEC Standard Performance Evaluation Corporation. The header features the SPEC logo and the text "Standard Performance Evaluation Corporation". Below the header is a navigation bar with links for Home, Benchmarks, Tools, Results, Contact, Blog, Join Us, Search, and Help. A sidebar on the left contains links for Results (Published Results, Results Search, Fair Use Policy), Information (CPU2017 Documentation, Systems Requirements, Plus II Reporting Rules, Using SPEC-CPU2017, Definitions, Technical Report, Nujni CPU), and About Amdahl. The main content area is titled "SPEC CPU® 2017" and describes the benchmark package.

- *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].

In-Datacenter Performance Analysis of a Tensor Processing Unit
<https://doi.org/10.1145/3079856.3080246>



The screenshot shows the NVIDIA Cloud & Data Center website. The top navigation bar includes the NVIDIA logo, a search bar, and links for Cloud & Data Center, Solutions, Products, and Data Center GPUs. The "Cloud & Data Center" menu is expanded, showing sub-links for Center and Resources. The main content area is titled "MLPerf Benchmarks" and discusses the NVIDIA AI platform's performance and versatility in MLPerf Training, Inference, and HPC for demanding real-world AI workloads.

The NVIDIA AI platform showcases leading performance and versatility in MLPerf Training, Inference, and HPC for the most demanding, real-world AI workloads.

Feedback for Post-Assignment Survey

- Why everything on server?
 - It's the new norm of industry
 - The new Google infrastructure only allow access through the web-based coding infrastructure similar to the codesever of escalab.org/datahub/
- Assignment is long and wordy
 - Assignment #1 is even longer
 - We want you to learn through doing it and spending time on every detail
 - Some of you actually want more explanation...
- Issues with Git/Linux/Terminal
 - It's a basic CS student skill that you should know how to address those
- No deadline extensions
 - You cannot expect deadline extension when you're at work, right?
 - Time management is the most important "soft" skill in your life

The use of Piazza

- Post answerable questions.
 - Examples of unanswerable questions:
 - Is this output correct?
 - My answer is like this, why is it considered incorrect in gradescope?
 - Don't do screenshot
 - It's taking us more time in find solutions for you as we cannot search text directly through our logs
 - It's also preventing you from Googling the message directly
 - It's not encouraged in Google's internal trouble shooting as well

Announcement

- Assignment #2 is up and due **this Thursday**
 - Please act quickly — it's going to be 2x longer than assignment #1
 - Only gradescope autograder result counts
 - I have seen only 1 submission on gradescope for now
 - escalab.org/datahub is only providing a platform for you to develop and maintain codebase
 - Hints: review our slides/demos would be helpful
 - Make sure your GitHub repo is up-to-date before your submission
- Check our website for slides, gradescope for quizzes/assignments, piazza for discussions
- Youtube channel for lecture recordings:
<https://www.youtube.com/c/ProfUsagi/playlists>

Computer Science & Engineering

203

つづく

