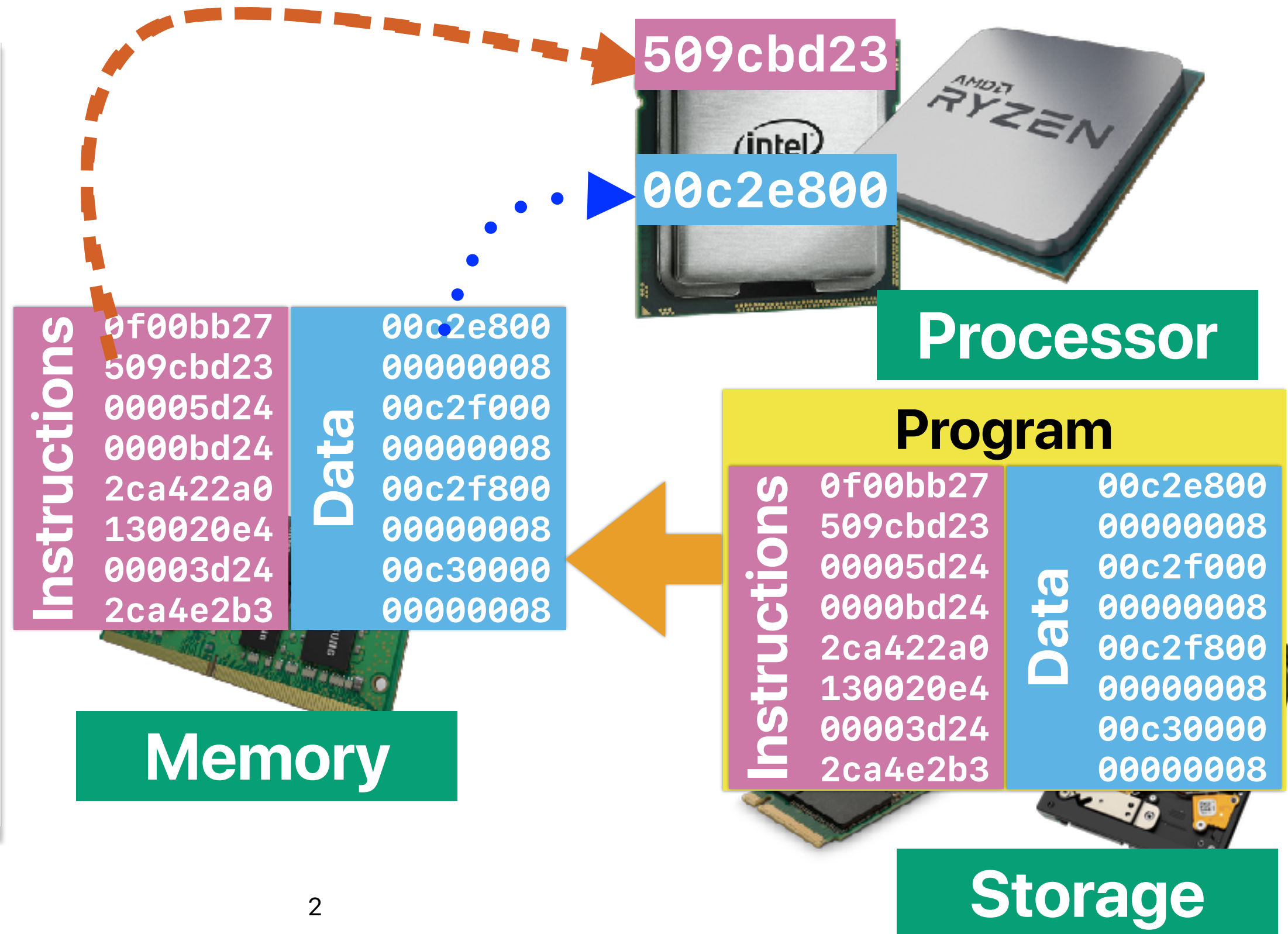


Performance (III) & Intro/ Memory Hierarchy

Hung-Wei Tseng

Recap: von Neumann Architecture



Recap: Definition of "Performance"

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

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

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

$$1GHz = 10^9 Hz = \frac{1}{10^9} sec\ per\ cycle = 1\ ns\ per\ cycle$$

Frequency(i.e., clock rate)

Recap: Definition of "Speedup"

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

$$n = \frac{\textit{Execution Time}_X}{\textit{Execution Time}_Y}$$

- The speedup of Y over X

$$\textit{Speedup} = \frac{\textit{Execution Time}_X}{\textit{Execution Time}_Y}$$

Recap: Amdahl's Law

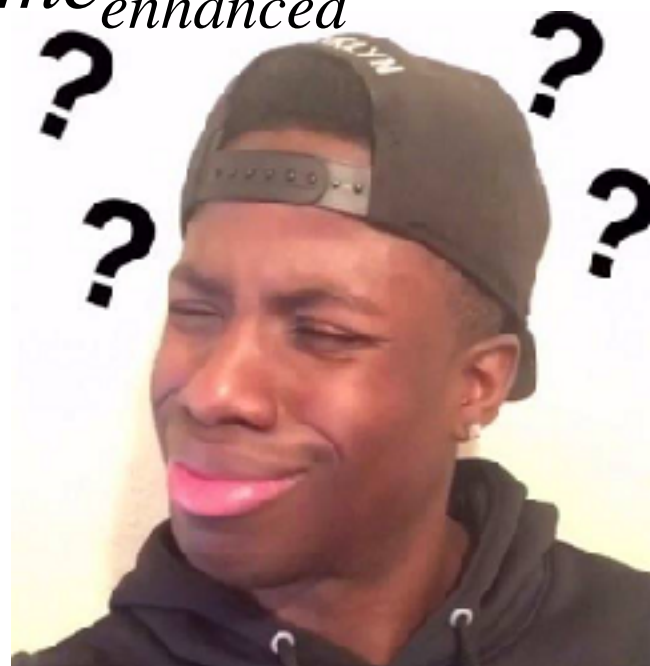


$$Speedup_{enhanced}(f, s) = \frac{1}{(1 - f) + \frac{f}{s}}$$

f — The fraction of time in the original program

s — The speedup we can achieve on f

$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}}$$



Amdahl's Law Corollary #1

- The maximum speedup is bounded by

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

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

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}!

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.

Outline

- Amdahl's Law (cont.)
- Fair Comparisons
- Right Metrics
- Introduction to Memory Hierarchy

Amdahl's Law (cont.)

Identify the most time consuming part

- Compile your program with -pg flag
- Run the program
 - It will generate a gmon.out
 - gprof your_program gmon.out > your_program.prof
- It will give you the profiled result in your_program.prof

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

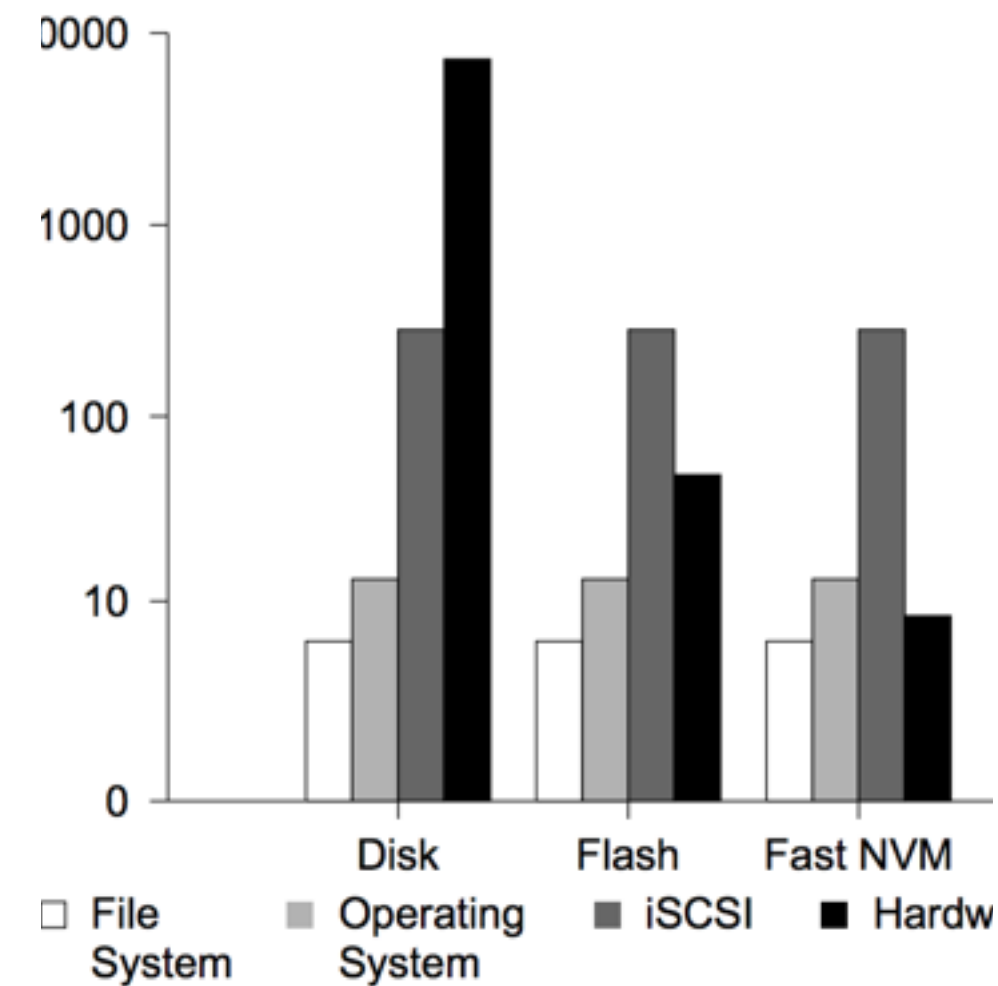
Storage Media

CPU

Storage
Media

CPU

- 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



Don't hurt non-common part too mach

- If the program spend 90% in A, 10% in B. Assume that an optimization can accelerate A by 9x, by hurts B by 10x...
- Assume the original execution time is T. The new execution

time $ET_{new} = \frac{ET_{old} \times 90\%}{9} + ET_{old} \times 10\% \times 10$

$$ET_{new} = 1.1 \times ET_{old}$$

$$Speedup = \frac{ET_{old}}{ET_{new}} = \frac{ET_{old}}{1.1 \times ET_{old}} = 0.91 \times \text{.....slowdown!}$$

You may not use Amdahl's Law for this case as Amdahl's Law does NOT
(1) consider overhead
(2) bound to slowdown

Amdahl's Law on Multicore Architectures

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

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

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

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

Amdahl's Law on Multicore Architectures

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

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

- 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
- 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

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})} \quad \text{speedup is determined by } 1-f$$

A. 0

B. 1

C. 2

D. 3

E. 4

Corollary #3, Corollary #4 & Corollary #5

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

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

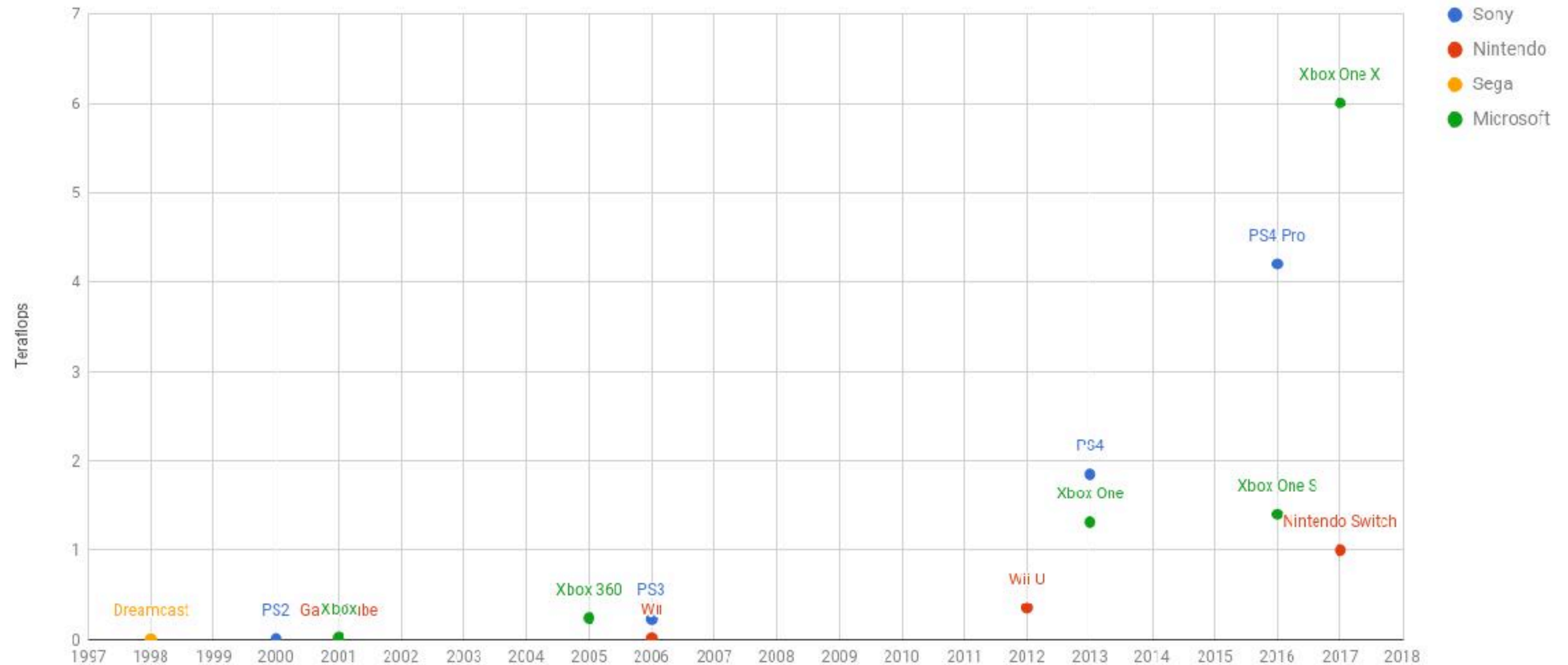
- Single-core performance still matters — it will eventually dominate the performance
- Finding more “parallelizable” parts is also important
- If we can build a processor with unlimited parallelism — the complexity doesn’t matter as long as the algorithm can utilize all parallelism — that’s why bitonic sort works!

“Fair” Comparisons

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

TFLOPS (Tera Floating-point Operations Per Second)

Console Teraflops



TFLOPS (Tera Floating-point Operations Per Second)

- TFLOPS does not include instruction count!
 - Cannot compare different ISA/compiler
 - Different CPI of applications, for example, I/O bound or computation bound
 - If new architecture has more IC but also lower CPI?

	TFLOPS	clock rate
XBOX One X	6	1.75 GHz
PS4 Pro	4	1.6 GHz
GeForce GTX 1080	8.228	3.5 GHz

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

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

- 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

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.

nvidia.com

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

8X Tesla V100

5.1 Hours

8X Tesla P100

15.5 Hours

0

4

8

12

16

Time to Solution in Hours—Lower Is Better

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

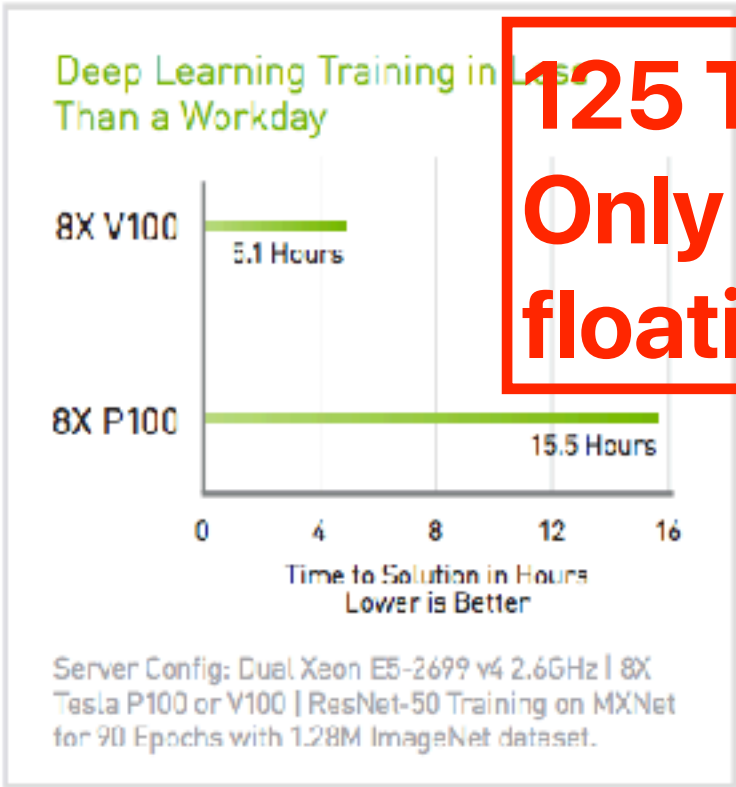
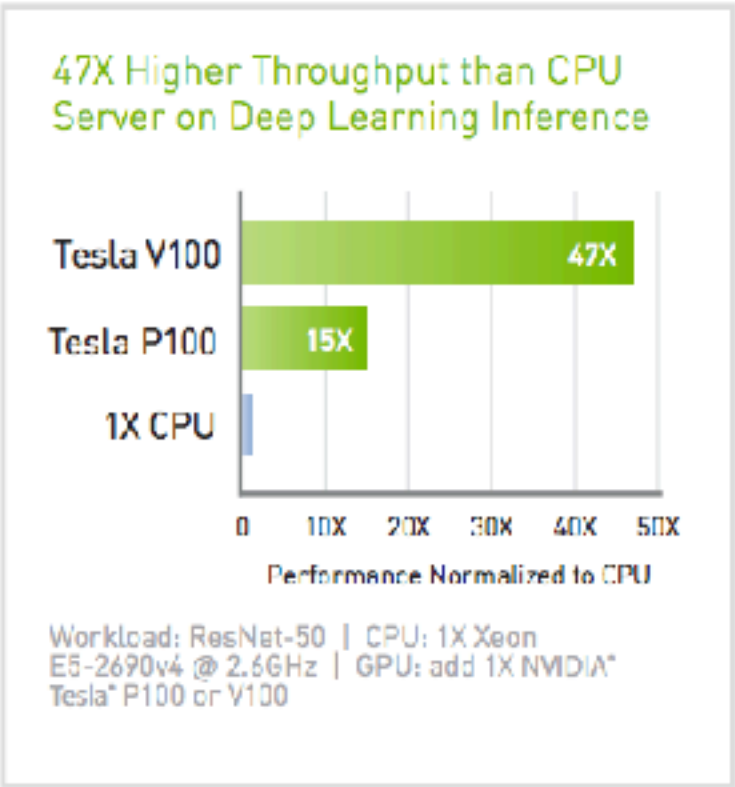
From recognizing speech to training virtual personal assistants and teaching autonomous cars to drive, data scientists are taking on increasingly complex challenges with AI. Solving these kinds of problems requires training deep learning models that are exponentially growing in complexity, in a practical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

24

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.

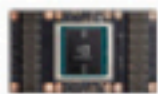


125 TFLOPS
Only @ 16-bit
floating point

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	300W	300W

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	$1/_{13}X$	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

What's wrong with inferences per second?

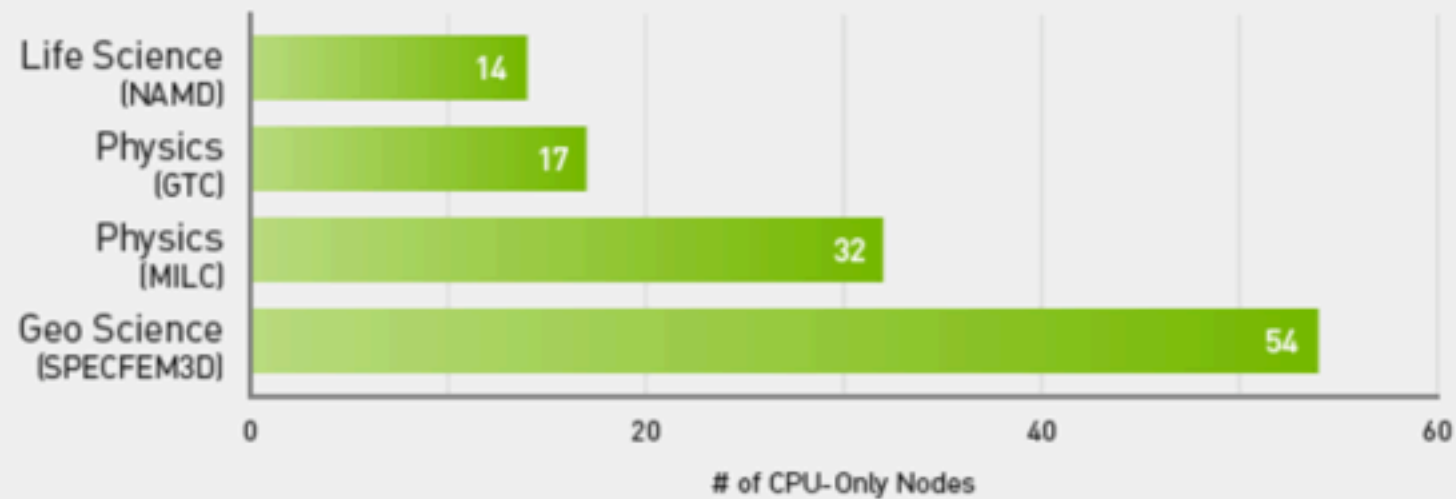
- There is no standard on how they inference
 - What model?
 - What dataset?
- That's why Facebook is trying to promote an AI benchmark — MLPerf

- *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].

1 GPU Node Replaces Up To 54 CPU Nodes

Node Replacement: HPC Mixed Workload



CPU Server: Dual Xeon Gold 6140@2.30GHz, GPU Servers: same CPU server w/ 4x V100 PCIe | CUDA Version: CUDA 9.x | Dataset: NAMD (STMV), GTC (mpi#proc.in), MILC (APEX Medium), SPECFEM3D (four_material_simple_model) | To arrive at CPU node equivalence, we use measured benchmark with up to 8 CPU nodes. Then we use linear scaling to scale beyond 8 nodes.

HIGH PERFORMANCE COMPUTING (HPC)

HPC is a fundamental pillar of modern science. From predicting weather to discovering drugs to finding new energy sources, researchers use large computing systems to simulate and predict our world. AI extends traditional HPC by allowing researchers to analyze large volumes of data for rapid insights where simulation alone cannot fully predict the real world.

Tesla V100 is engineered for the convergence of AI and HPC. It offers a platform for HPC systems to excel at both computational science for scientific simulation and data science for finding insights in data. By pairing NVIDIA CUDA[®] cores and **Tensor Cores** within a unified architecture, a single server with Tesla V100 GPUs can replace hundreds of commodity CPU-only servers for both traditional HPC and AI workloads. Every researcher and engineer can now afford an AI supercomputer to tackle their most challenging work.



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?

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

Latency v.s. Bandwidth/Throughput

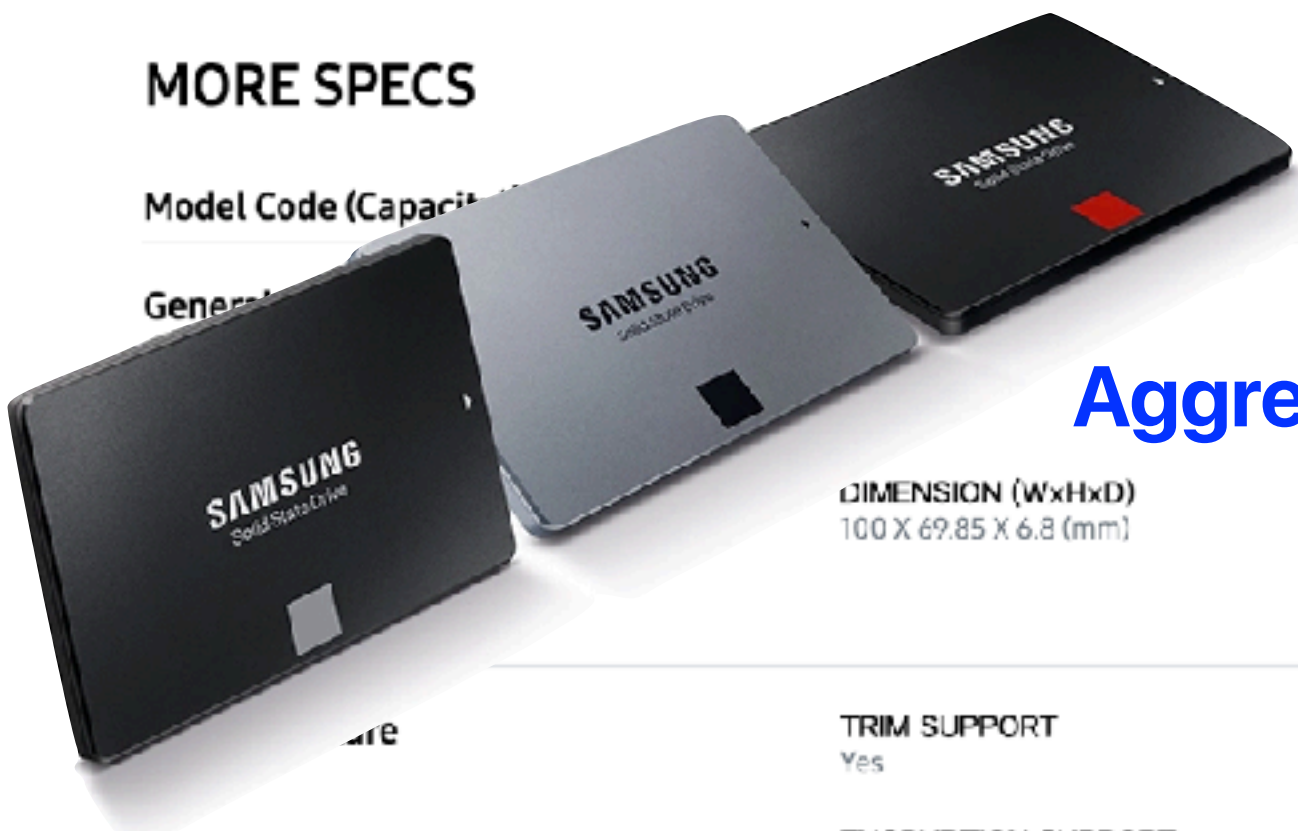
- Latency — the amount of time to finish an operation
 - access time
 - response time
- Throughput — the amount of work can be done within a given period of time
 - bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
 - IOPs
 - MFLOPs

RAID — Improving throughput

MORE SPECS

Model Code (Capacity)

Generation



Aggregated Bandwidth: 500 MB/sec

DIMENSION (WxHxD)
100 X 69.85 X 6.8 (mm)

TRIM SUPPORT
Yes

ENCRYPTION SUPPORT
AES 256-bit Encryption (Class 0) TCG/Opal
IEEE1667 (Encrypted drive)

Performance²⁾

SEQUENTIAL READ
Up to 550 MB/s

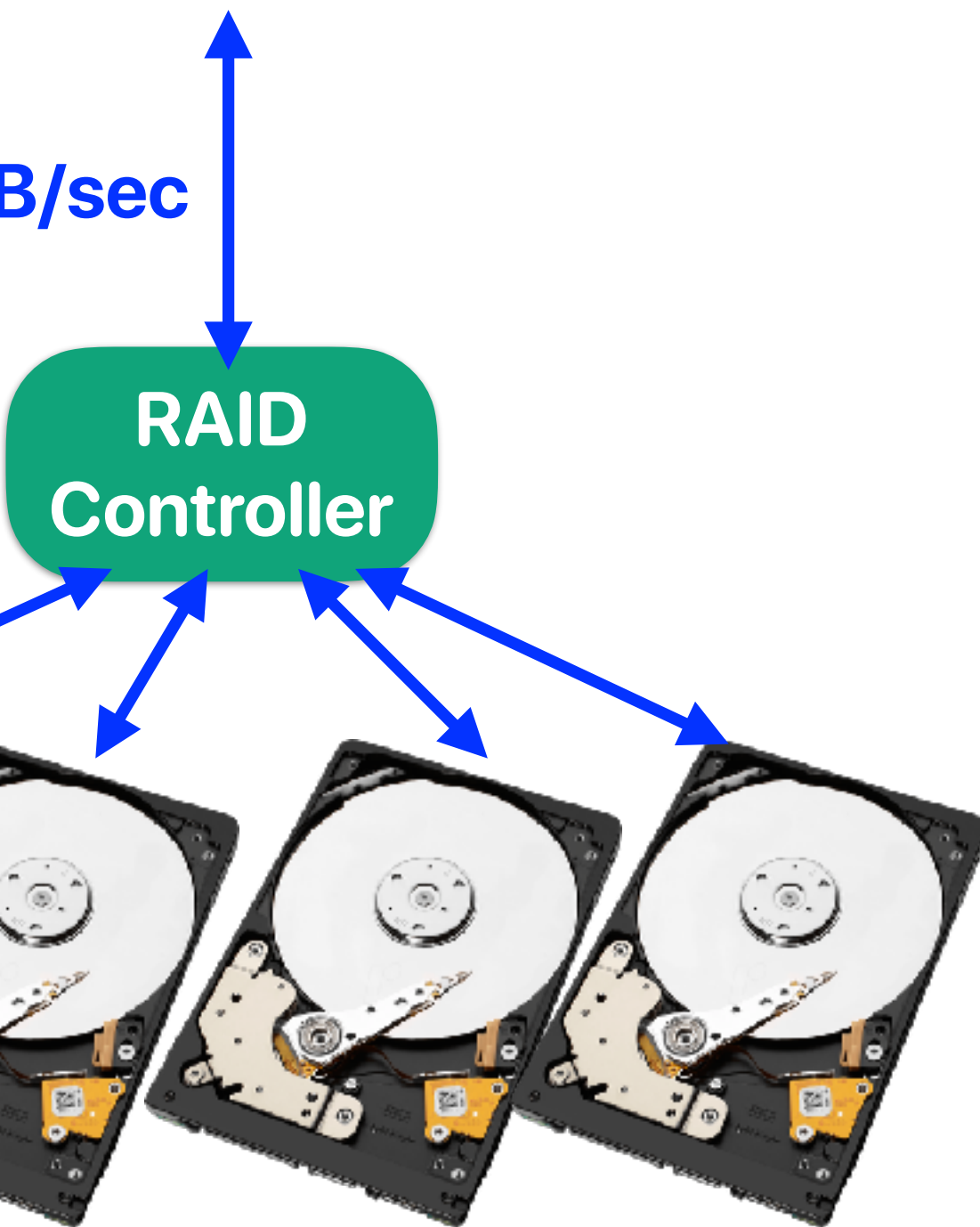
RANDOM WRITE (4KB, QD32)
Up to 89,000 IOPS

Environment

AVERAGE POWER CONSUMPTION
(SYSTEM LEVEL)³⁾

1,000 GB: Average 2.2 W Maximum 4.0 W
2,000 GB: Average 3.1 W Maximum 4.2 W
4,000 GB: Average 3.1 W Maximum 5.4 W
(Burst mode)

Access time: 10 ms
Bandwidth: 125 MB/sec



The performance between RAID and SSD

- Compare (X) RAID consists of 4x H.D.D. where each has 10 ms access time and 125 MB/sec bandwidth — aggregated bandwidth at 500 MB/Sec (Y) a single SSD with 100 us access time and 550MB/Sec bandwidth. Both accept 4KB data as the smallest request size. If we want to load a program with 100KB code size, how much faster is Y over X at least?
 - A. 1x — no speedup
 - B. 1.1x
 - C. 4x
 - D. 4.4x
 - E. 100x

The performance between RAID and SSD

- Compare (X) RAID consists of 4x H.D.D. where each has 10 ms access time and 125 MB/sec bandwidth — aggregated bandwidth at 500 MB/Sec (Y) a single SSD with 100 us access time and 550MB/Sec bandwidth. Both accept 4KB data as the smallest request size. If we want to load a program with 100KB code size, how much faster is Y over X at least?
 - A. 1x — no speedup
 - B. 1.1x
 - C. 4x
 - D. 4.4x
 - E. 100x

The performance between RAID and SSD

- Compare (X) RAID consists of 4x H.D.D. where each has 10 ms access time and 125 MB/sec bandwidth — aggregated bandwidth at 500 MB/Sec (Y) a single SSD with 100 us access time and 550MB/Sec bandwidth. Both accept 4KB data as the smallest request size. If we want to load a program with 100KB code size, how much faster is Y over X at least?

A. 1x — no speedup

B. 1.1x

C. 4x

D. 4.4x

E. 100x

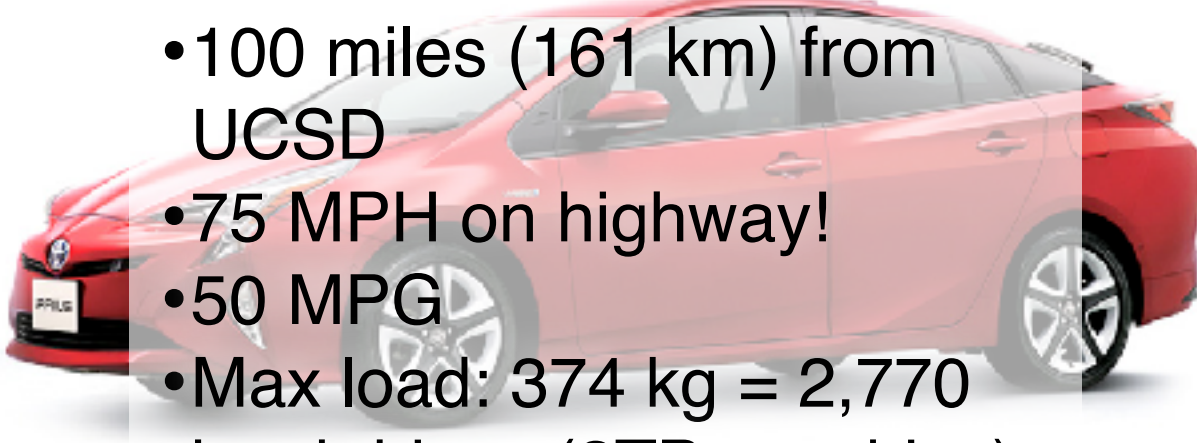

$$ET_{HDD_{BestCase}} = 10 \text{ ms}$$

$$ET_{SSD_{worst}} = \frac{100KB}{4K} \times 100 \text{ us} = 2.5 \text{ ms}$$

Latency and Bandwidth trade-off

- Increase bandwidth can hurt the response time of a single task
- If you want to transfer a 2 Peta-Byte video from UCSD
 - 100 miles (161 km) from UCR
 - Assume that you have a 100Gbps ethernet
 - 2 Peta-byte over 167772 seconds = 1.94 Days
 - 22.5TB in 30 minutes
 - Bandwidth: 100 Gbps

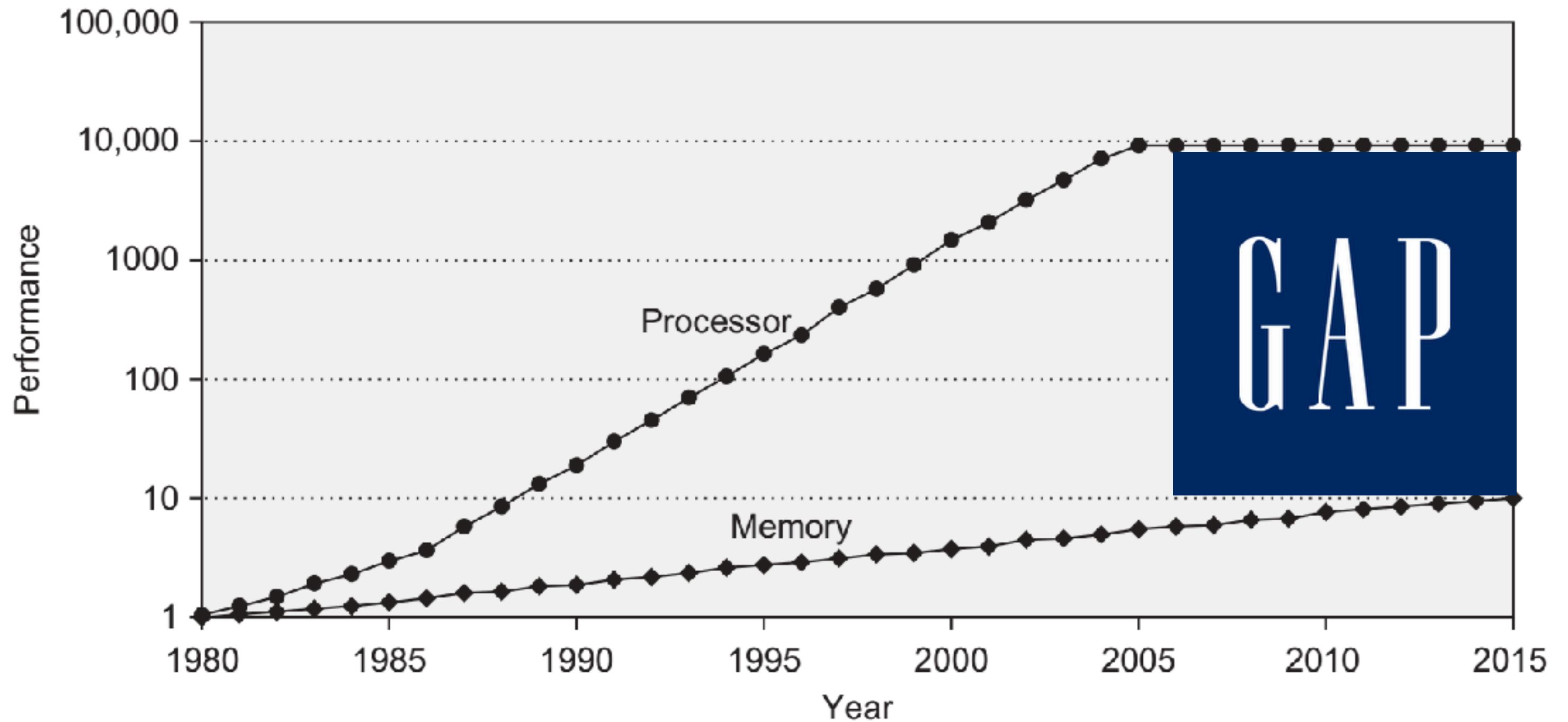
Or ...

	<div>Toyota Prius</div> <div><ul style="list-style-type: none">• 100 miles (161 km) from UCSD• 75 MPH on highway!• 50 MPG• Max load: 374 kg = 2,770 hard drives (2TB per drive)</div> <div></div>	<div>10Gb Ethernet</div> <div></div>
bandwidth	290GB/sec	100 Gb/s or 12.5GB/sec
latency	3.5 hours	2 Peta-byte over 167772 seconds = 1.94 Days
response time	You see nothing in the first 3.5 hours	You can start watching the movie as soon as you get a frame!

Memory Hierarchy

Hung-Wei Tseng

Performance gap between Processor/Memory



Performance of modern DRAM

Production year	Chip size	DRAM type	Best case access time (no precharge)			Precharge needed
			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.

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

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

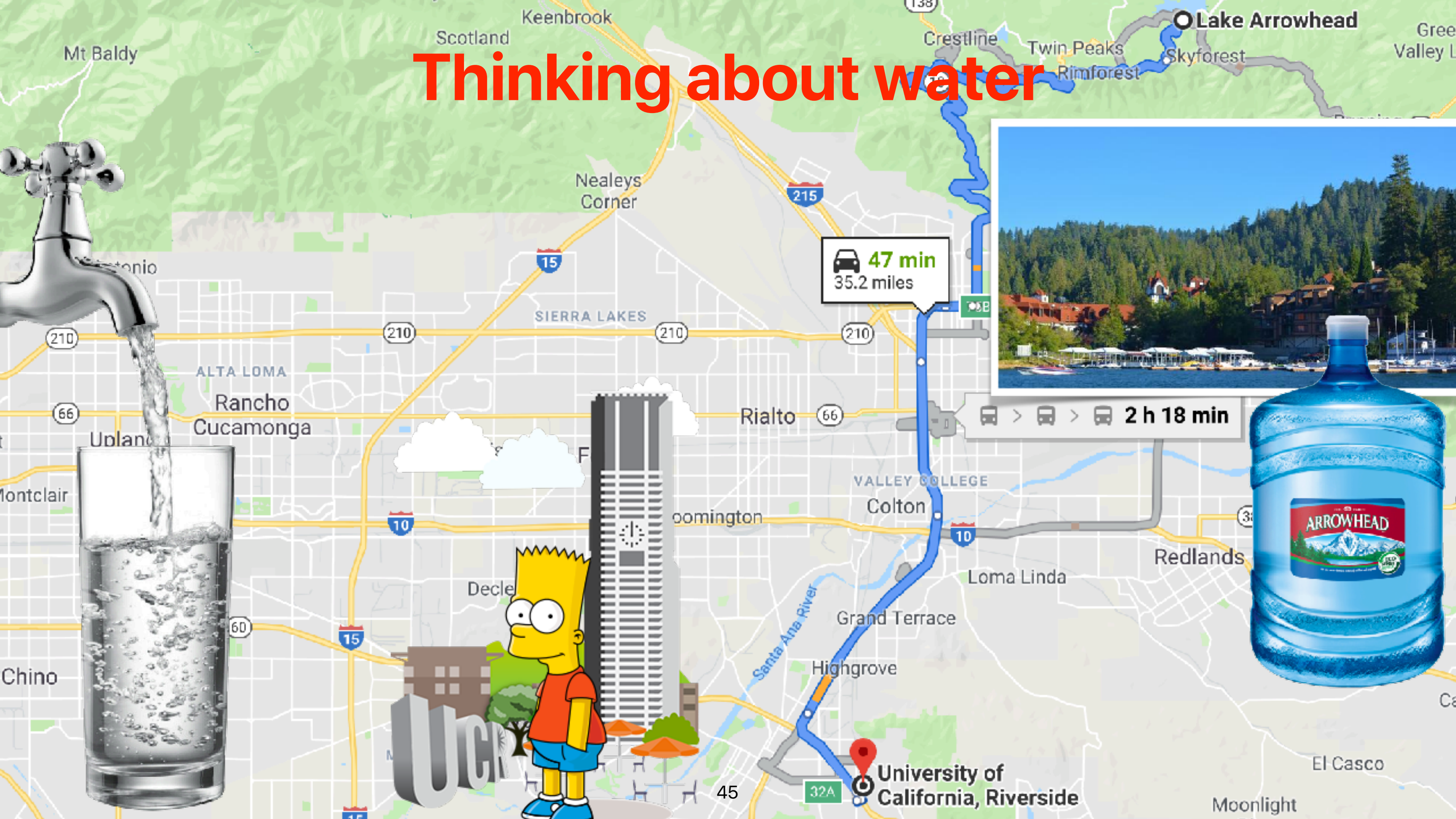
C. 27

D. 35

E. 69

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

Thinking about water



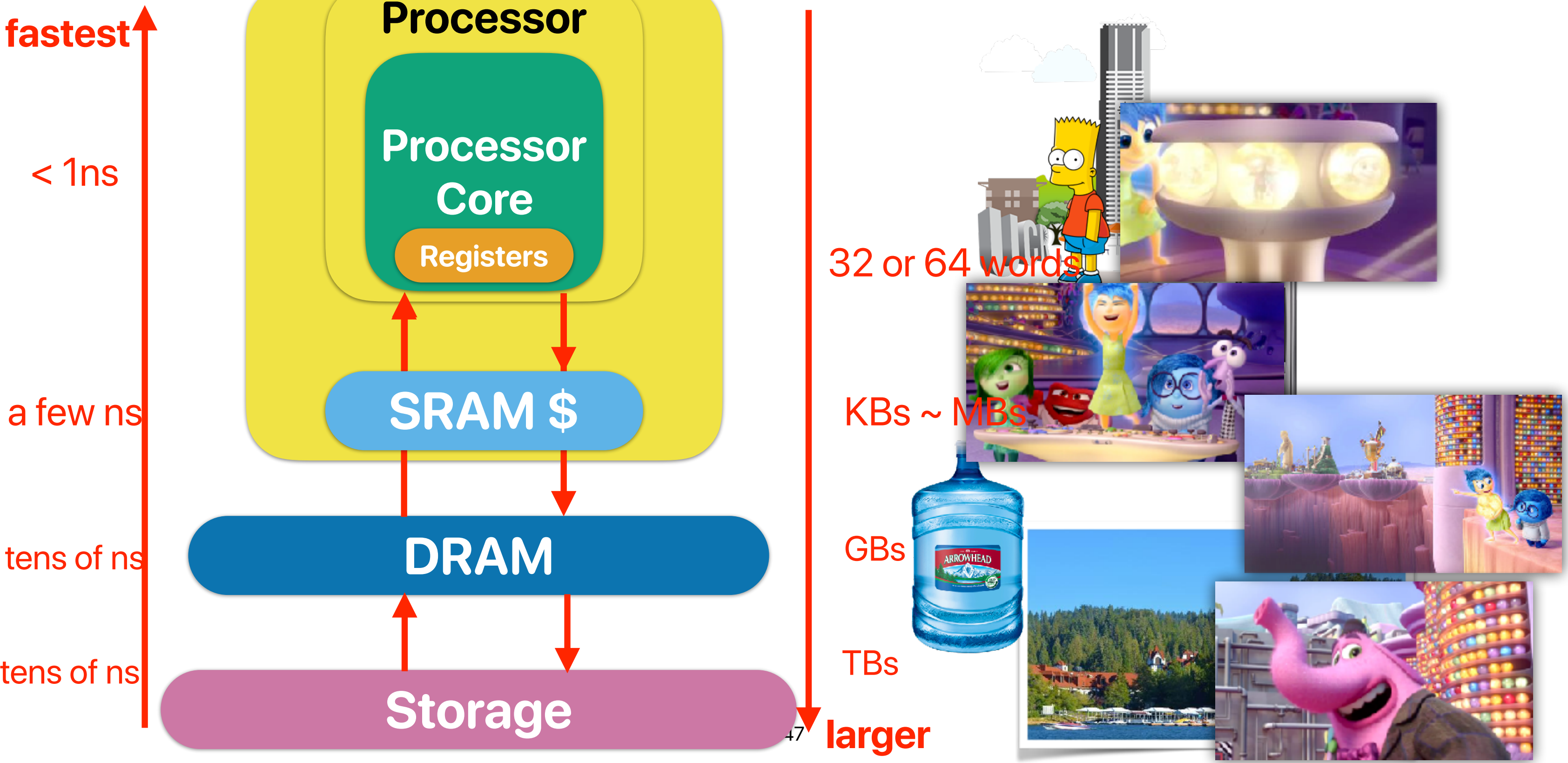
Alternatives?

Memory technology	Typical access time	\$ per GiB in 2012
SRAM semiconductor memory	0.5–2.5 ns	\$500–\$1000
DRAM semiconductor memory	50–70 ns	\$10–\$20
Flash semiconductor memory	5,000–50,000 ns	\$0.75–\$1.00
Magnetic disk	5,000,000–20,000,000 ns	\$0.05–\$0.10



Fast, but expensive \$\$\$

Memory Hierarchy



How can memory hierarchy help in performance?

- 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, in addition to DDR4, whose latency 26 ns, we also got an SRAM cache with latency of just at 0.5ns and can capture 90% of the desired data/instructions. what's the average CPI (pick the most close one)?

- A. 2
- B. 4
- C. 8
- D. 16
- E. 32

How can memory hierarchy help in performance?

- 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, in addition to DDR4, whose latency 26 ns, we also got an SRAM cache with latency of just at 0.5ns and can capture 90% of the desired data/instructions. what's the average CPI (pick the most close one)?

- A. 2
- B. 4
- C. 8
- D. 16
- E. 32

How can memory hierarchy help in performance?

- 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, in addition to DDR4, whose latency 26 ns, we also got an SRAM cache with latency of just at 0.5ns and can capture 90% of the desired data/instructions. what’s the average CPI (pick the most close one)?
 - A. 2
 - B. 4
 - C. 8

 $1 + (1 - 90\%) \times [100\% \times (52) + 30\% \times 52] = 7.76 \text{ cycles}$
 - D. 16
 - E. 32

Announcement

- Office hour of Hung-Wei Tseng changes
— **MF 1p-2p @ WCH 406**
- Check
 - website for slides/schedules
 - iLearn for quizzes/assignments/podcasts
 - piazza for discussions
- No lectures next week
- Assignment #1 due 10/16
- Reading quiz due 10/21