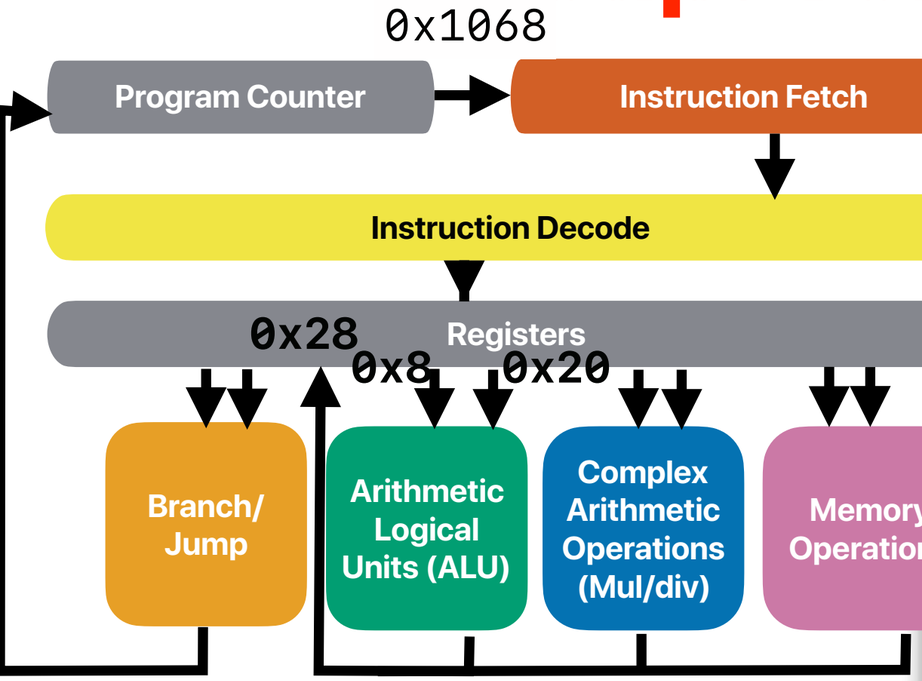
Performance

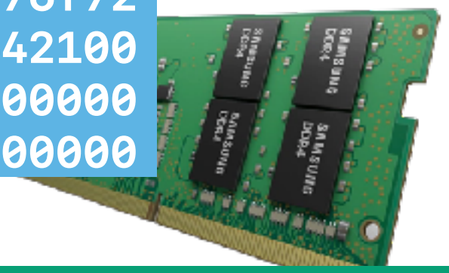
Hung-Wei Tseng

Recap: von Neumann architecture



Processor

Instructions	Data
f30f1efa	08400000
4883ec08	00000100
488d3d95	02004865
0f0000e8	6c6c6f2c
dcffffff	20776f72
31c04883	6c642100
c408c30f	00000000
1f440000	00000000



Memory

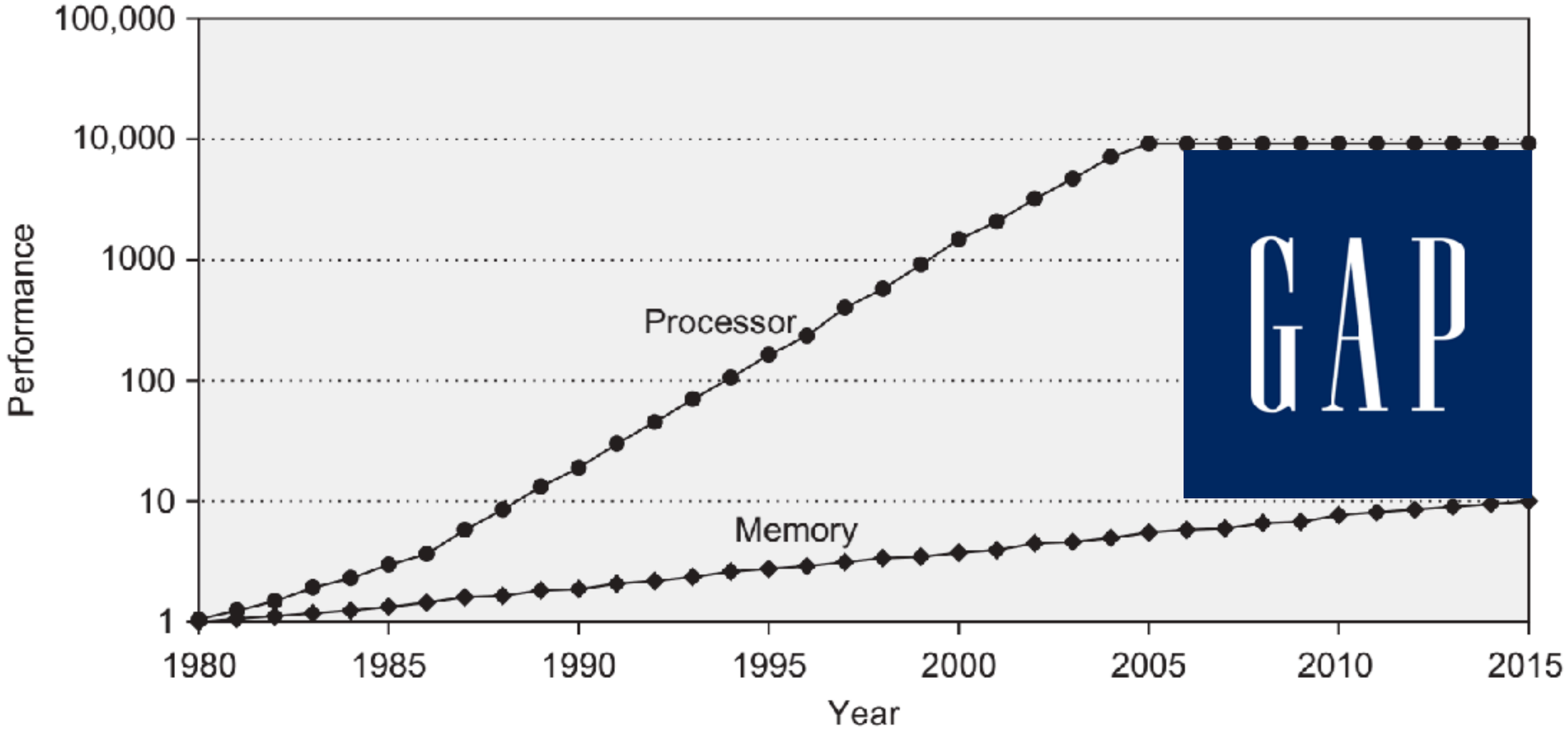
```
int main(){
    printf("Hello, world!\n");
}
```

Instructions	Data
f30f1efa	08400000
4883ec08	00000100
488d3d95	02004865
0f0000e8	6c6c6f2c
dcffffff	20776f72
31c04883	6c642100
c408c30f	00000000
1f440000	00000000

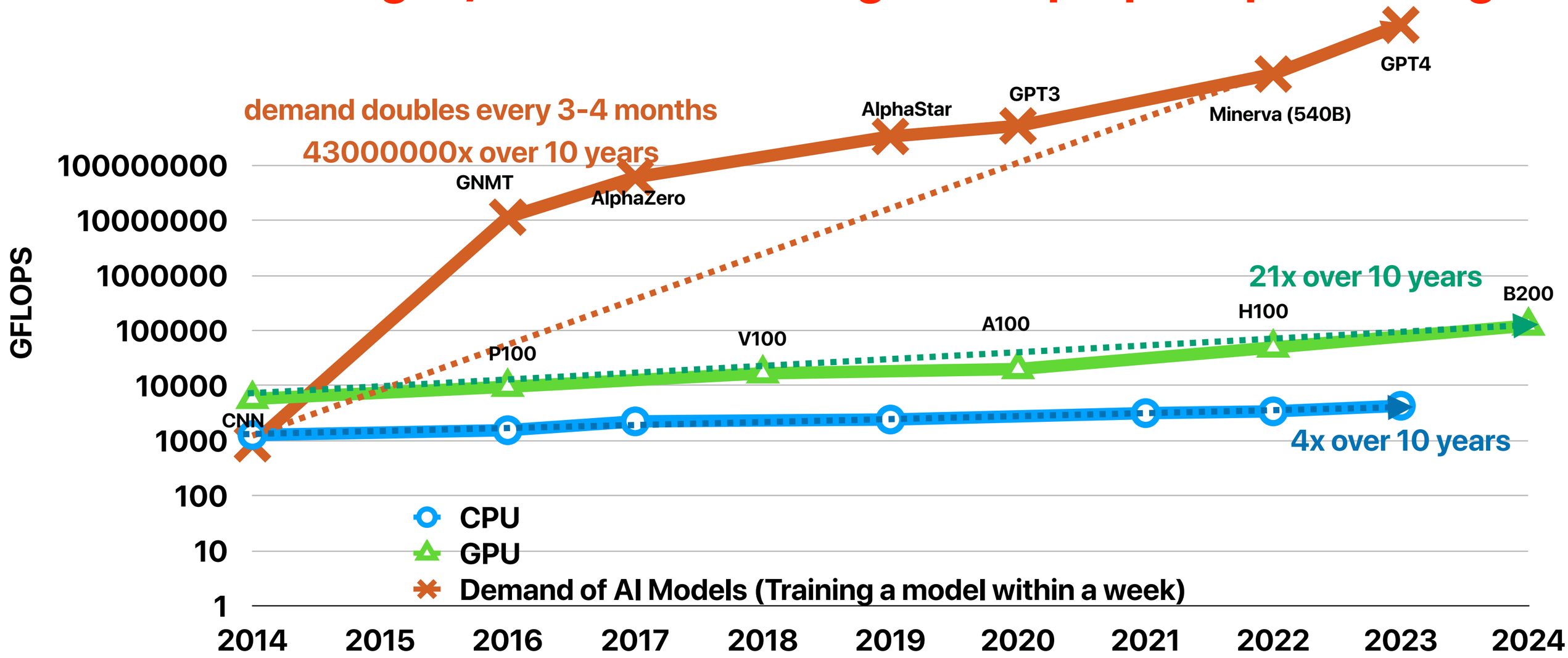
Storage

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

Recap: Performance gap between Processor/Memory



Mis-matching AI/ML demand and general-purpose processing



https://ourworldindata.org/grapher/artificial-intelligence-training-computation

Outline

- Definition of “Performance”
- The performance equation
- Other metrics and are they good?

Best Nation

Schools in the National Univers
a full range of undergraduate r
producing groundbreaking res

To unlock full rankings, SAT/A

SUMMARY

443

Scho

Loca

Rank

Nat

Best Compute

Best Computer Science Schools

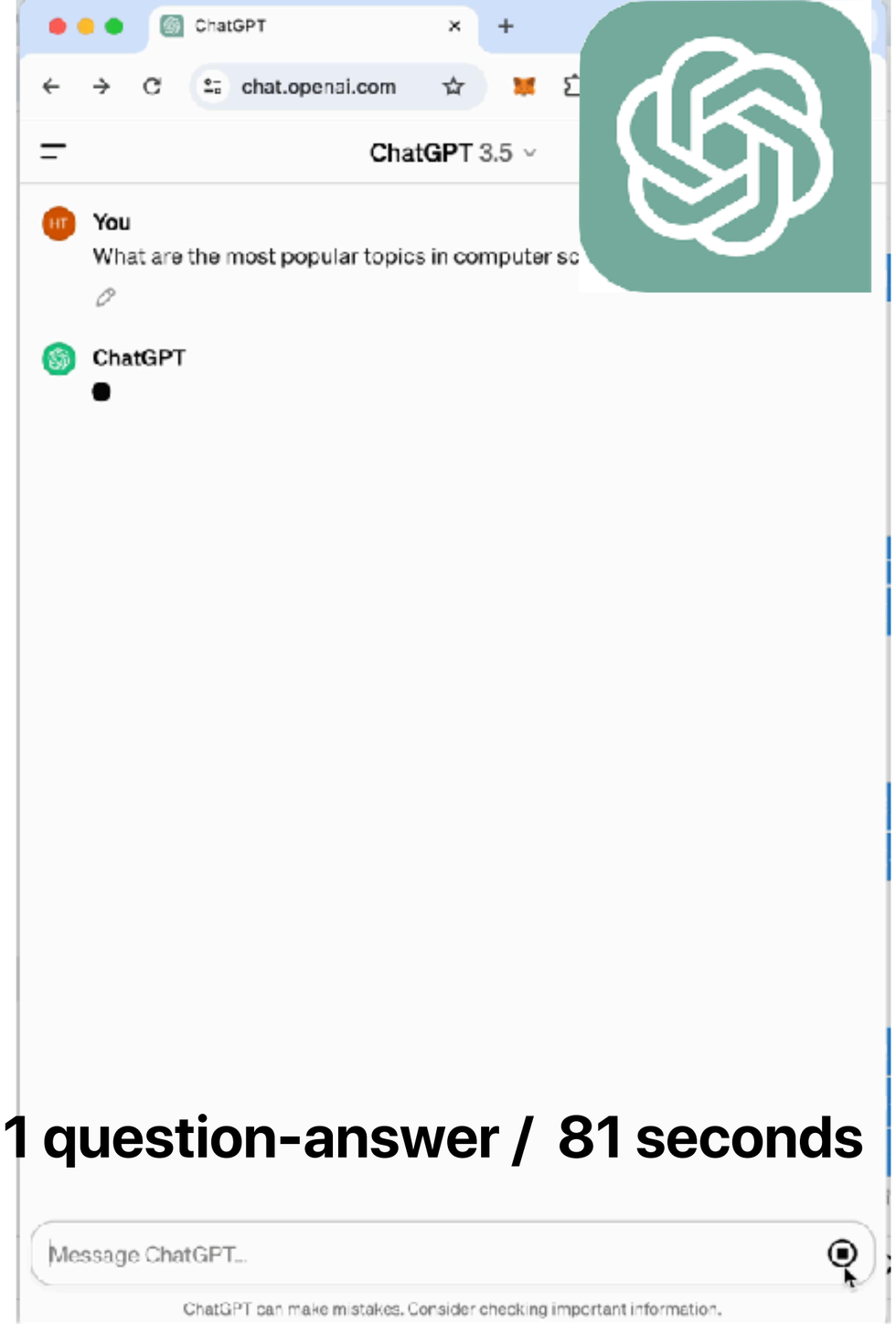
Ranked in 2022, part of Best Science S

Earning a graduate degree in computer
technology companies and colleges ar
reflects its average rating on a scale fr
institutions. Read the methodology »

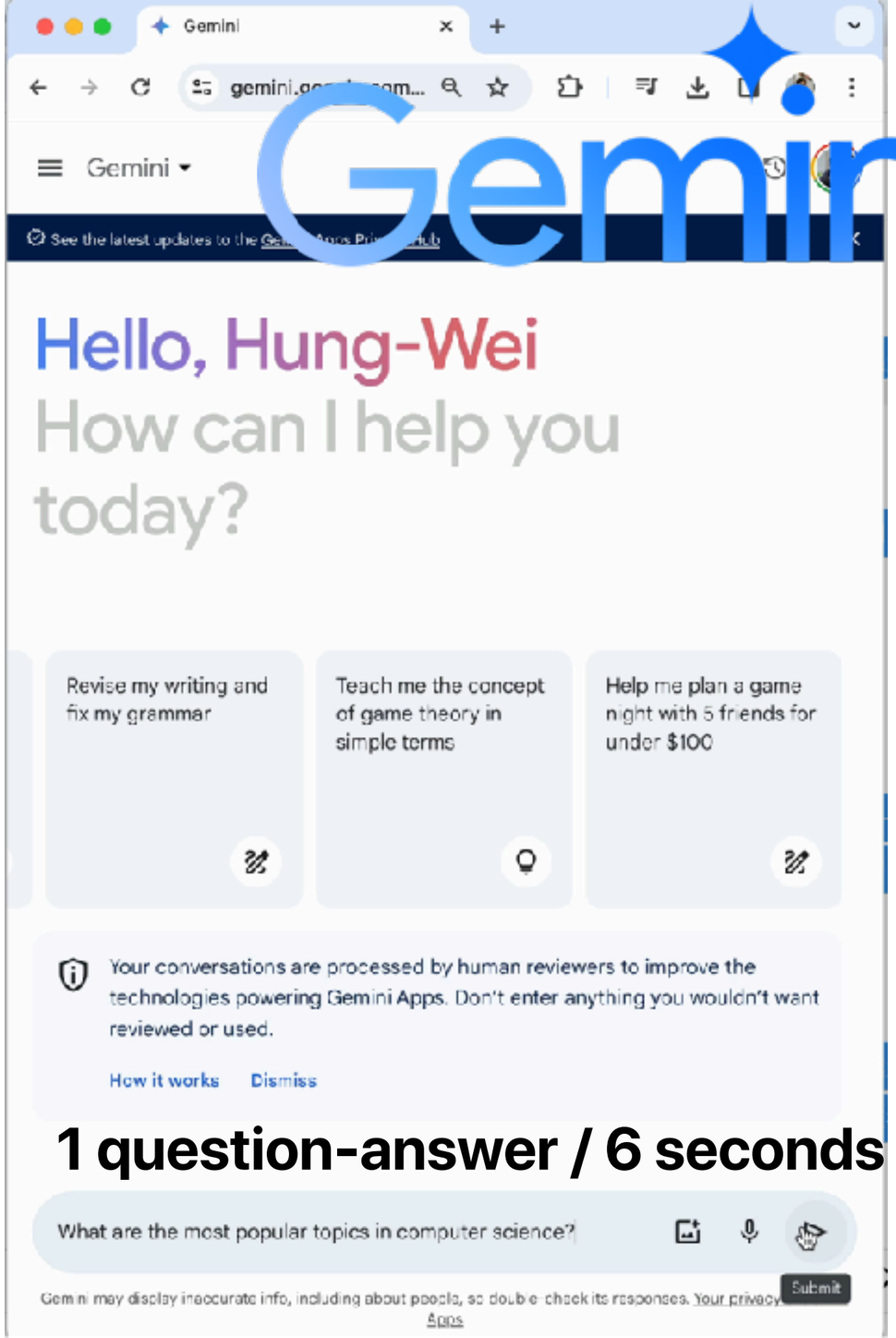
UC San Diego Ranked No. 1 Public University by Washington Monthly

Campus celebrated as a leader in social mobility, research and public

What does it really mean by “better” performance



1 question-answer / 81 seconds



1 question-answer / 6 seconds

Important performance metrics

- End-to-end latency — how much time the program/operation takes from the beginning to the end
- Response time — how much time the user starts to feel the program is running/finishing
- Throughput/bandwidth — the average amount of work/data can the program/system deliver within the execution time
- Energy consumption — the aggregated power during the execution time
- Cost of operation — the amount of money necessary for finishing an operation
- Quality of results — the human perception of the execution result
- Power consumption — the heat generation produced by the circuit

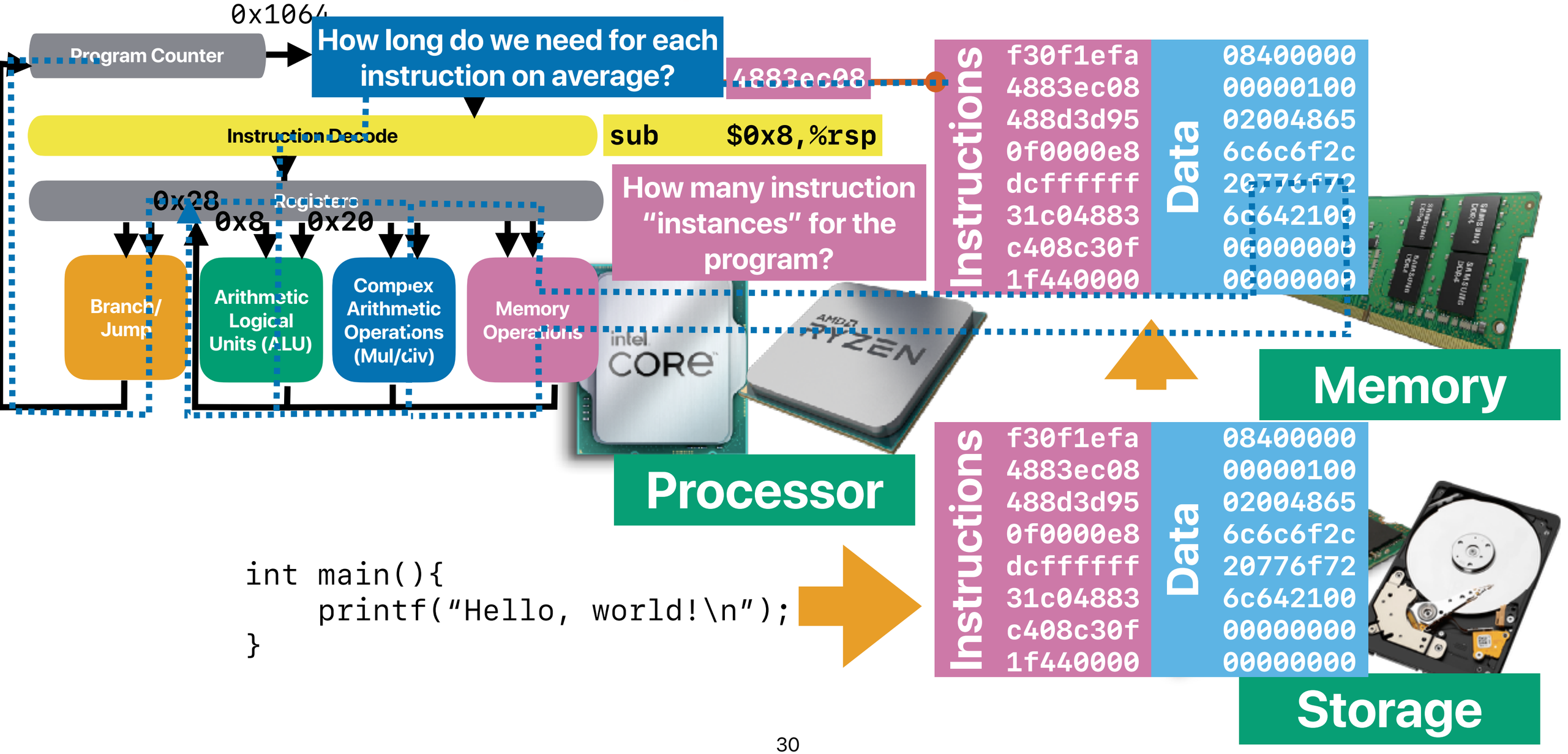
Important performance metrics

- End-to-end latency — how much **time** the program/operation takes from the beginning to the end
- Response time — how much **time** the user starts to feel the program is running/finishing
- Throughput/bandwidth — the average amount of work/data can the program/system deliver within the execution **time**
- Energy consumption — the aggregated power during the execution **time**
- Cost of operation — the amount of money necessary for finishing an operation (related to **time**)
- Quality of results — the human perception of the execution result
- Power consumption — the heat generation produced by the circuit

Takeaways: What does “perfect” mean?

- Latency is the most fundamental performance metric

Execution time of a program in the von Neumann model



Classic CPU Performance Equation (ET of a program)

How many instruction "instances" for the program? × **How long do we need for each instruction on average?**

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

C Code	x86 instructions

```
int init_data(int64_t *data, int
data_size) {
    register unsigned int i = 0;
    for(i = 0; i < data_size; i++)
{
       s+=data[i];
    }
    return s;
}

int main(int argc, char **argv) {
    int *data = malloc(8000000000);
    init_data(data, 1000000000)
    return 0;
}
```

```
init_data:
.LFB16:
   endbr64
   testl  %esi, %esi
   jle .L2
   leal-1(%rsi), %ecx
   xorl%eax, %eax
.L3:
   movslq   (%rdi), %rdx
   addq     $4, %rdi
   addq     %rdx, %tax
   cmpq     %rcx, %rdi
   jne       .L3
.L2:
   xorl%eax, %eax
   ret
```

If data memory access instructions takes 4 cycles, others take only 1 cycle, CPU freq. = 4 GHz

$$CPI_{average} = 20\% \times 4 + 20\% \times 3 + 60\% \times 1) = 2$$

$$ET = (5 \times 10^9) \times 2 \times \frac{1}{4 \times 10^9}\ sec = 2.5\ sec$$

1000000000x

CPU Performance Equation

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

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

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

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

$$\frac{1}{Frequency(i.e., clock\ rate)}$$

Performance equation

- Consider the following c code snippet and x86 instructions implement the code snippet

C	x86

```
for(i = 0; i < count; i++) {
    s += a[i];
}
```

```
.L3:
movslq  (%rdi), %rdx
addq    $4, %rdi
addq    %rdx, %tax
cmpq    %rcx, %rdi
jne     .L3
```

If (1) count is set to 1,000,000,000, (2) a memory instruction takes 4 cycles, (3) a branch/jump instruction takes 3 cycles, (4) other instructions takes 1 cycle on average, and (5) the processor runs at 4 GHz, how much time is it take to finish executing the code snippet?

A. 0.5 sec
B. 1 sec
C. 2.5 sec
D. 3.75 sec
E. 4 sec

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

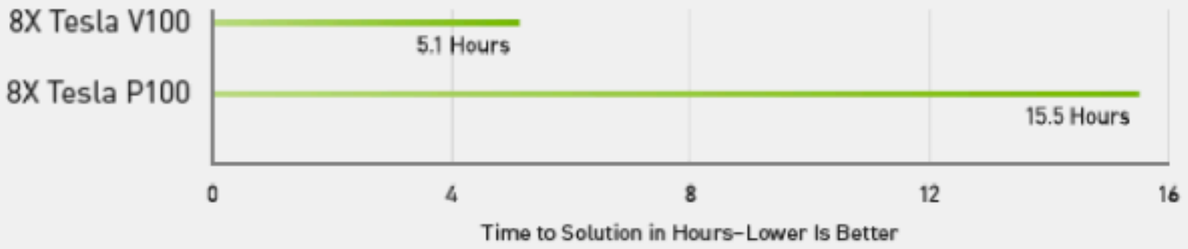
$$ET = \underbrace{(5 \times 10^9)}_{\text{total \# of dynamic instructions}} \times \underbrace{(20\% \times 4 + 20\% \times 3 + 60\% \times 1)}_{\text{average CPI}} \times \frac{1}{4 \times 10^9}\ sec = 2.5\ sec$$

Takeaways: What does “perfect” mean?

- Latency is the most fundamental performance metric
- Instruction count, cycles per instruction, cycle time define the latency of execution on CPUs

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

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.

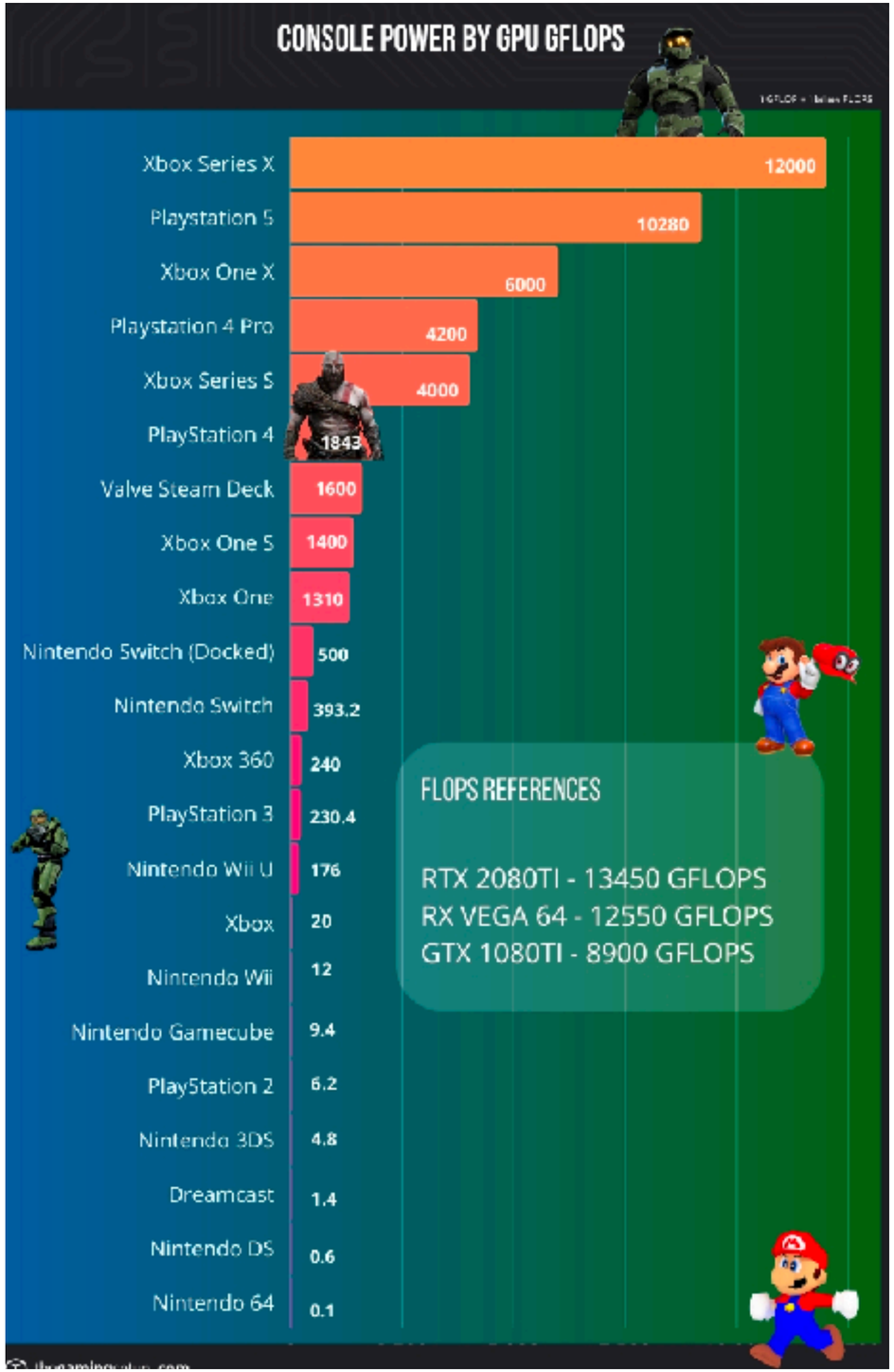
AI TRAINING

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



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"):

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

Floating point operations (FLOP"s"):

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

Given $i = j = k = 2048$

$2^{3\times11} \times 2 = 2^{34}$ **FLOPs in total**

TFLOPS (Tera FLoating-point Operations Per Second)

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

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!

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

What's wrong with inferences per second?

- There is no standard on how they inference — but these affect!
  - What model?
  - What dataset?
  - Quality?
- 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].

Inference per second

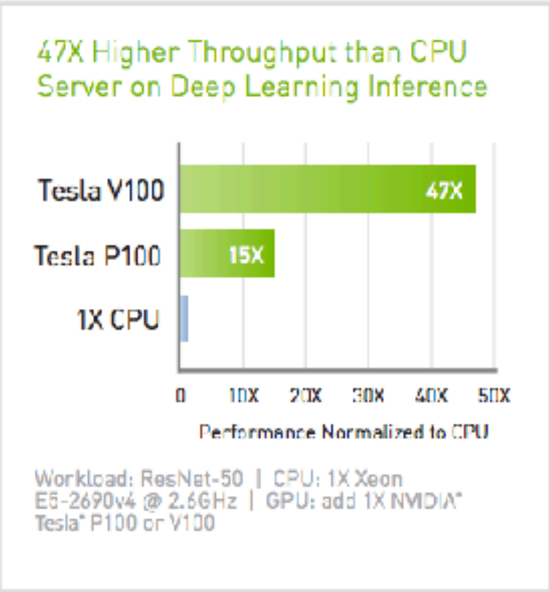
$$\frac{Inferences}{Second} = \frac{Inferences}{Operation} \times \frac{Operations}{Second}$$

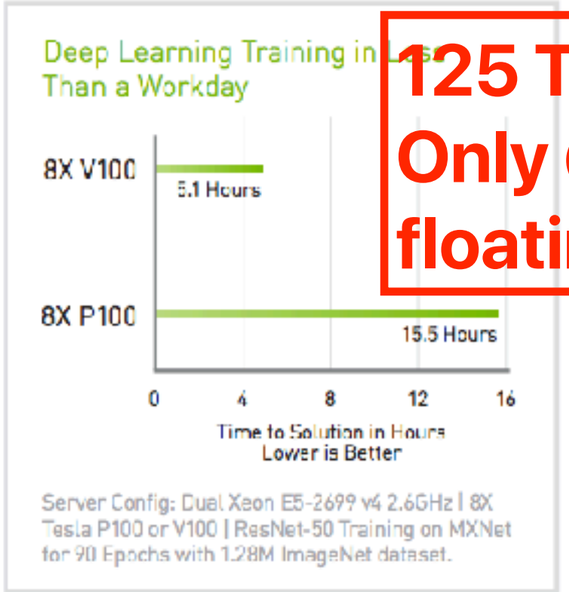
$$= \frac{Inferences}{Operation} \times [\frac{operations}{cycle} \times \frac{cycles}{second} \times \#\_of\_PEs \times Utilization\_of\_PEs]$$

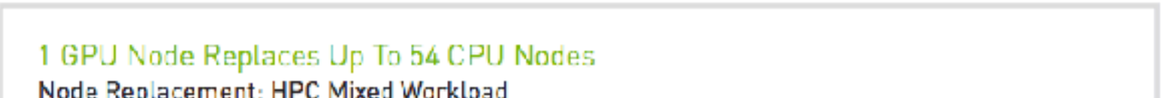
	Hardware	Model	Input Data
Operations per inference		v	
Operations per cycle	v		
Cycles per second	v		
Number of PEs	v		
Utilization of PEs	v	v	
Effectual operations out of (total) operations		v	v
Effectual operations plus unexploited ineffectual operations per cycle	v		

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.

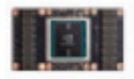




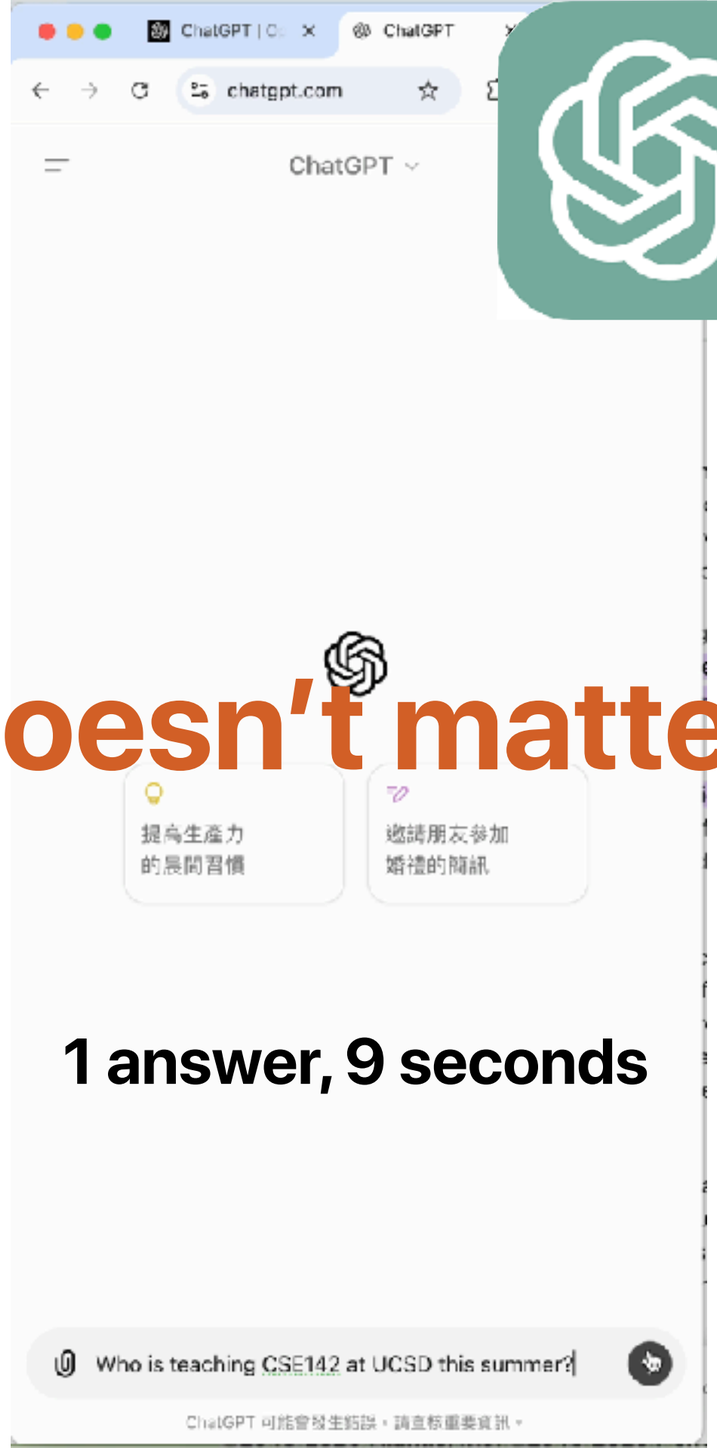


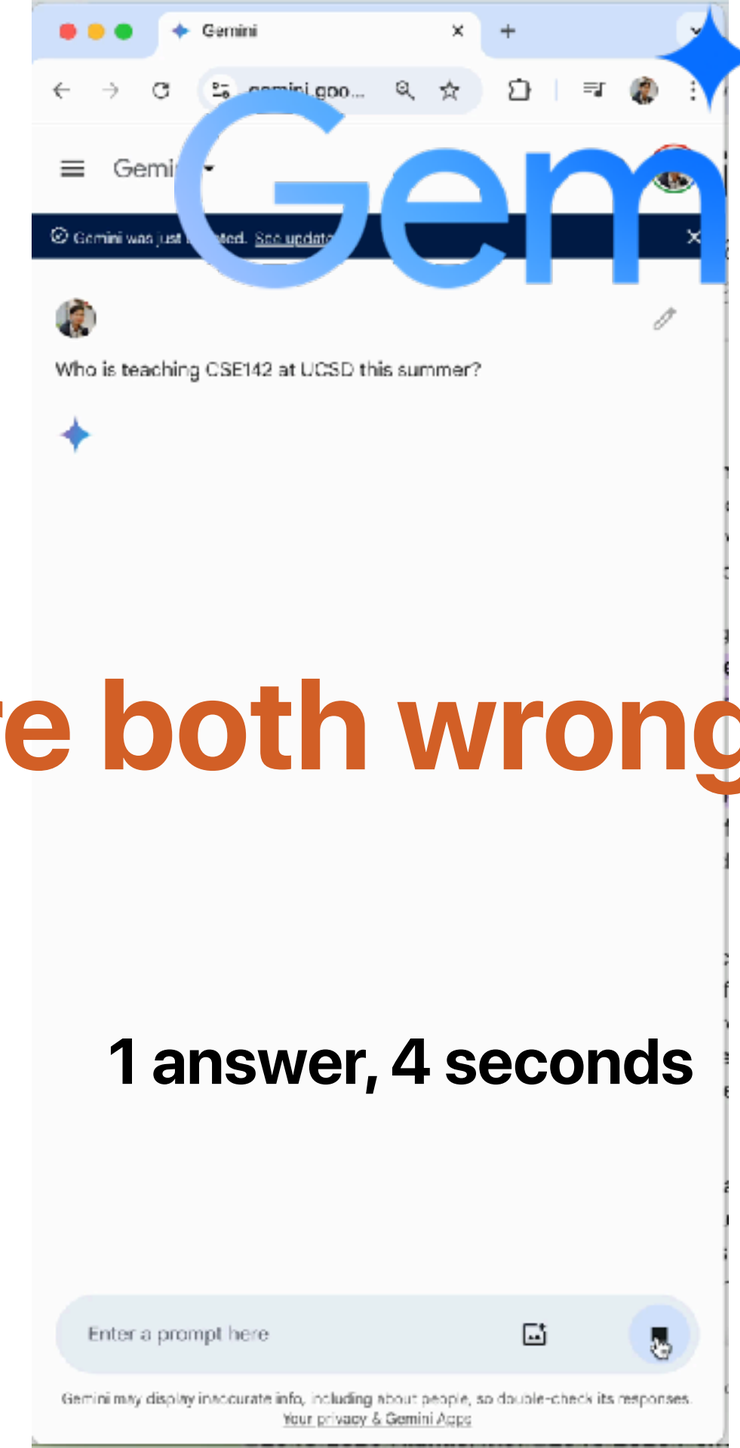
125 TFLOPS Only @ 16-bit floating point

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		





Doesn’t matter — they’re both wrong :)

1 answer, 9 seconds

1 answer, 4 seconds

gamespot.com/articles/nvidia-accused-of-cheating-in-3dmark-03/1100-6028694/#:~:text="We%20have%20now%20established%20that,dr

Nvidia accused of cheating in 3DMar

Futuremark, the maker of 3DMark 03, releases a patch for the g to correct the way Nvidia drivers manipulate results in the popu

By David Becker on May 23, 2003 at 4:27PM PDT

Saratoga, California-based Futuremark on Friday said in a statement that Nvidia tweaked the software needed to run its new GeForce FX 5900 processor to distort performance in Futuremark's 3DMark 03 testing application. The company said drivers for the new Nvidia chip were altered to detect activity characteristic of a benchmark and adjust performance accordingly.

Recently, there have been questions and some confusion regarding 3DMark 03 results obtained with certain Nvidia products, Futuremark said in the statement. "We have now established that Nvidia's Detonator FX drivers contain certain detection mechanisms that cause an artificially high score when using 3DMark 03," the statement read.

Futuremark has released the version 330 patch for 3DMark 03, which prevents the Nvidia drivers from identifying the benchmark. By Futuremark's measure, the performance of the Nvidia GeForce FX 5900 Ultra drops 24 percent with the patch, compared with a drop of less than 2 percent with ATI's Radeon 9800 Pro with the latest ATI drivers.

A representative at Nvidia questioned the validity of Futuremark's conclusions. "Since Nvidia is not part of the Futuremark beta program (a program which costs of hundreds of thousands of dollars to participate in), we do not get a chance to work with Futuremark on writing the shaders like we would with a real applications developer," the representative said. "We don't know what they did, but it looks like they have intentionally tried to create a scenario that makes our products look bad."

Nvidia is amid a hard-fought battle with rival ATI Technologies to claim the performance lead in PC graphics processors. After years of Nvidia dominating both in market share and performance, ATI

https://www.gamespot.com/articles/nvidia-accused-of-cheating-in-3dmark-03/1100-6028894/#:~:text=%22We%20have%20now%20established%20that,drivers%20from%20identifying%20the%20benchmark.

Latency v.s. Bandwidth/Throughput

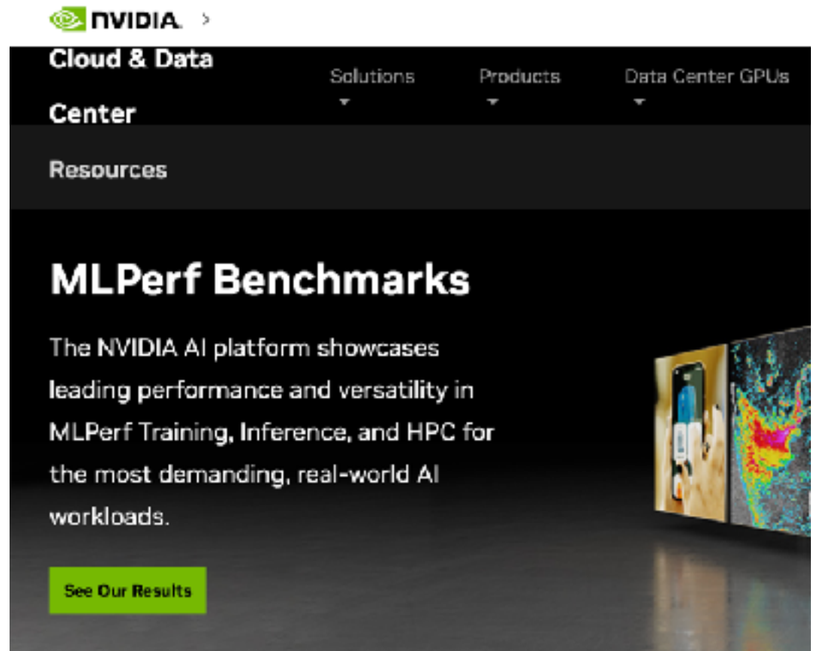
- Latency — the amount of time to finish an operation
  - End-to-end execution time of “something”
  - Access time
  - Response time
- Throughput — the amount of work can be done within a given period of time (typically “something” per “timeframe” or the other way around)
  - Bandwidth (MB/Sec, GB/Sec,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

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

The reason of “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



Takeaways: What does “perfect” mean?

- Latency is the most fundamental performance metric
- Classic CPU performance equation —Instruction count (IC), cycles per instruction (CPI), cycle time (CT) define the latency of execution on CPUs
- Performance metrics without considering all three factors in the classic performance equation can mislead — anything throughput typically miss one of them

An increasingly important performance metric

un.org/en/climatechange/net-zero-coalition

ome to the United Nations العربية 中文 English Françe

For a livable climate: Net-zero commitments must be

What is net zero?

Put simply, net zero means cutting carbon emissions to a small amoun
nature and other carbon dioxide removal measures, leaving zero in the

Why is net zero important?

The science shows clearly that in order to avert the worst impacts of cl
increase needs to be limited to 1.5°C above pre-industrial levels. Curren
1800s, and emissions continue to rise. To keep global warming to no r
need to be reduced by 45% by 2030 and reach net zero by 2050.

How can net zero be achieved?

Transitioning to a net-zero world is one of the greatest challenges hum
transformation of how we produce, consume, and move about. The en
emissions today and holds the key to averting the worst effects of clim
energy from renewable sources, such as wind or solar, would dramatic

Is there a global effort to reach net zero?

Yes, a growing coalition of countries, cities, businesses and other instit
countries, including the biggest polluters – China, the United States, In
about 88% of global emissions. More than 9,000 companies, over 100
financial institutions have joined the Race to Zero, pledging to take rigo

sustainability.google/operating-sustainably/net-zero-carbon/

Net-zero carbon

DOI: 10.1145/3637867

BY AMIN VAHDAT, XIAOYU MA, AND DAVID PATTERSON

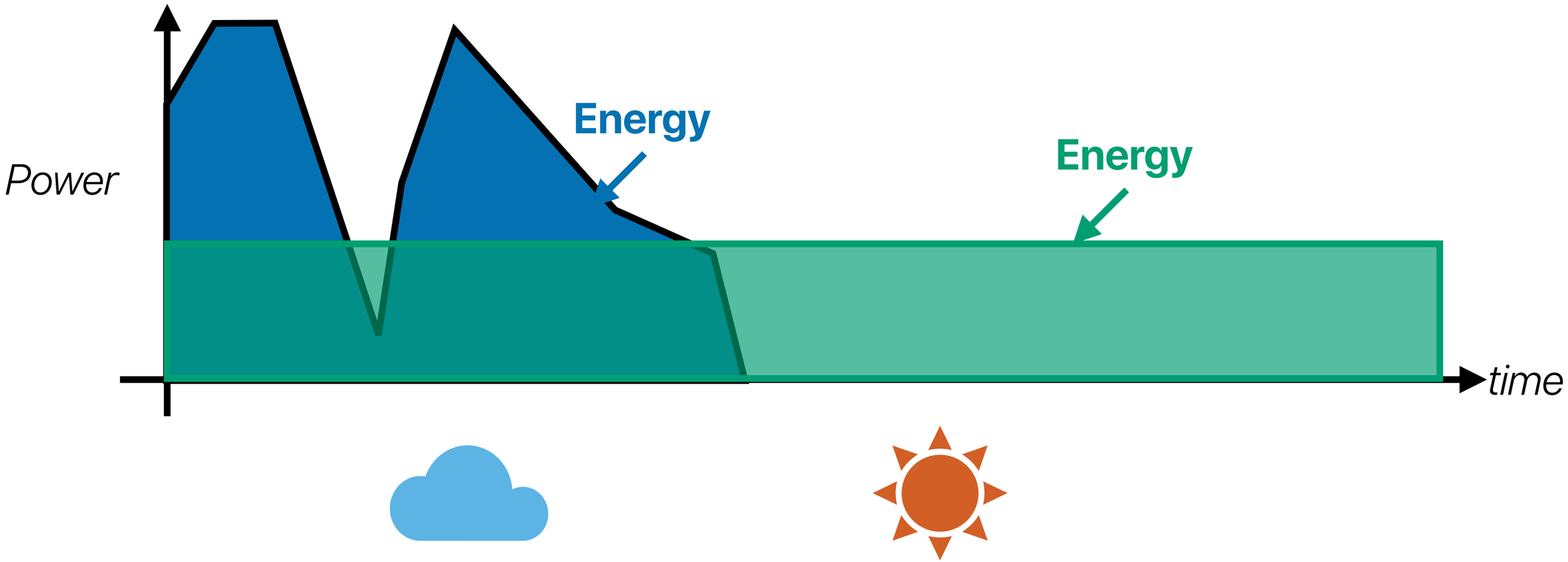
New Computer Evaluation Metrics for a Changing World

New Computer Evaluation Metrics for a Changing World

Amin Vahdat, Xiaoyu Ma, David Patterson
Google
Communications of the ACM, 2024

Power/Energy/Carbon footprint

The Green can be more if power is not low enough



If we run the task when there is no green energy— more carbon footprint!

$$Energy = Power \times Execution\_Time$$

Demo — changing the max frequency and performance

- Change the maximum frequency of the intel processor — you learned how to do this when we discuss programmer's impact on performance
- LIKWID a profiling tool providing power/energy information
  - likwid-perfctr -g ENERGY [command_line]
  - Let's try blockmm and popcount and see what's happening!

Takeaways: What does "perfect" mean?

- Latency is the most fundamental performance metric
- Classic CPU performance equation —Instruction count (IC), cycles per instruction (CPI), cycle time (CT) define the latency of execution on CPUs
- Performance metrics without considering all three factors in the classic performance equation can mislead — anything throughput typically miss one of them
- CO_2e, energy efficiency and power become increasingly important — but latency/execution time still matters as

$$Energy = Power \times Execution\_Time$$

What Affects Each Factor in Performance Equation

Demo — programmer & performance

A

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

B

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

$O(n^2)$ **Complexity** $O(n^2)$

Instruction Count?

Clock Rate

CPI

Demo — programmer & performance

A

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

B

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

$O(n^2)$	Complexity	$O(n^2)$
Same	Instruction Count?	Same
Same	Clock Rate	Same
???	CPI	???

Use “performance counters” to figure out!

- Modern processors provides performance counters
  - instruction counts
  - cache accesses/misses
  - branch instructions/mis-predictions
- How to get their values?
  - You may use “perf stat” in linux
  - You may use Instruments —> Time Profiler on a Mac
  - Intel’s vtune — only works on Windows w/ intel processors
  - You can also create your own functions to obtain counter values

Demo — programmer & performance

A

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

B

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

$O(n^2)$	Complexity	$O(n^2)$
Same	Instruction Count?	Same
Same	Clock Rate	Same
Better	CPI	Worse

Programmers can also set the cycle time

https://software.intel.com/sites/default/files/comment/1716807/how-to-change-frequency-on-linux-pub.txt

```
==========================================
Subject: setting CPU speed on running linux system


If the OS is Linux, you can manually control the CPU speed by reading and writing some virtual files in the "/proc"

1.) Is the system capable of software CPU speed control?
If the "directory" /sys/devices/system/cpu/cpu0/cpufreq exists, speed is controllable.
-- If it does not exist, you may need to go to the BIOS and turn on EIST and any other C and P state control and vi



2.) What speed is the box set to now?
Do the following:
$ cd /sys/devices/system/cpu
$ cat ./cpu0/cpufreq/cpuinfo_max_freq
3193000
$ cat ./cpu0/cpufreq/cpuinfo_min_freq
1596000

3.) What speeds can I set to?
Do
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
It will list highest settable to lowest; example from my NHM "Smackover" DX58SO HEDT board, I see:
3193000 3192000 3059000 2926000 2793000 2660000 2527000 2394000 2261000 2128000 1995000 1862000 1729000 159600
You can choose from among those numbers to set the "high water" mark and "low water" mark for speed.  If you set "h


4.) Show me how to set all to highest settable speed!
Use the following little sh/ksh/bash script:
$ cd /sys/devices/system/cpu  # a virtual directory made visible by device drivers
$ newSpeedTop=`awk '{print $1}' ./cpu0/cpufreq/scaling_available_frequencies`
$ newSpeedLow=$newSpeedTop  # make them the same in this example
$ for c in ./cpu[0-9]* ; do
>   echo $newSpeedTop >${c}/cpufreq/scaling_max_freq
>   echo $newSpeedLow >${c}/cpufreq/scaling_min_freq
> done
$

5.) How do I return to the default - i.e. allow machine to vary from highest to lowest?
Edit line # 3 of the script above, and re-run it.  Change the line:
$ newSpeedLow=$newSpeedTop  # make them the same in this example
```

Takeaways: What matters?

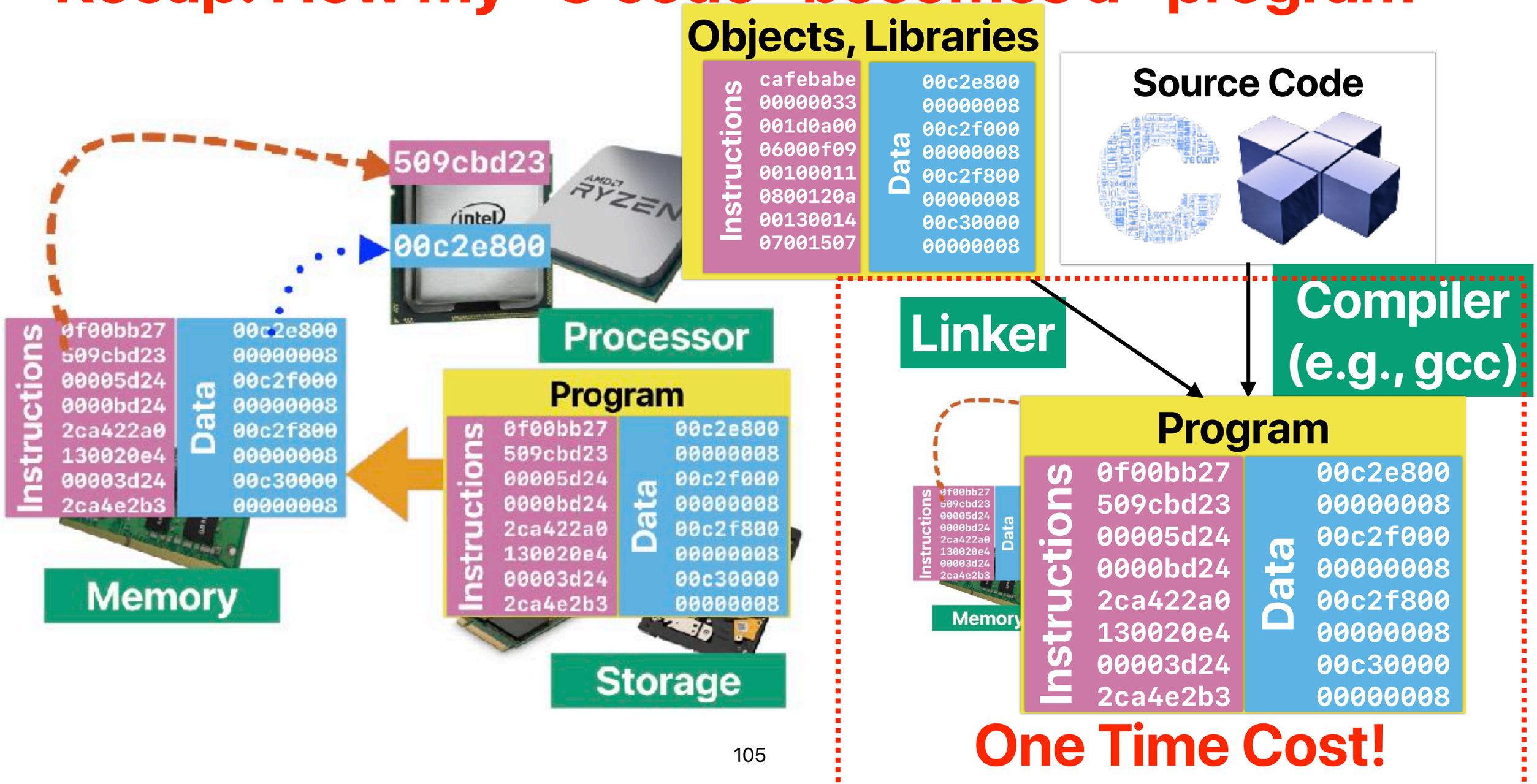
- Programmers can control all three factors in the classic performance equation

Programming languages

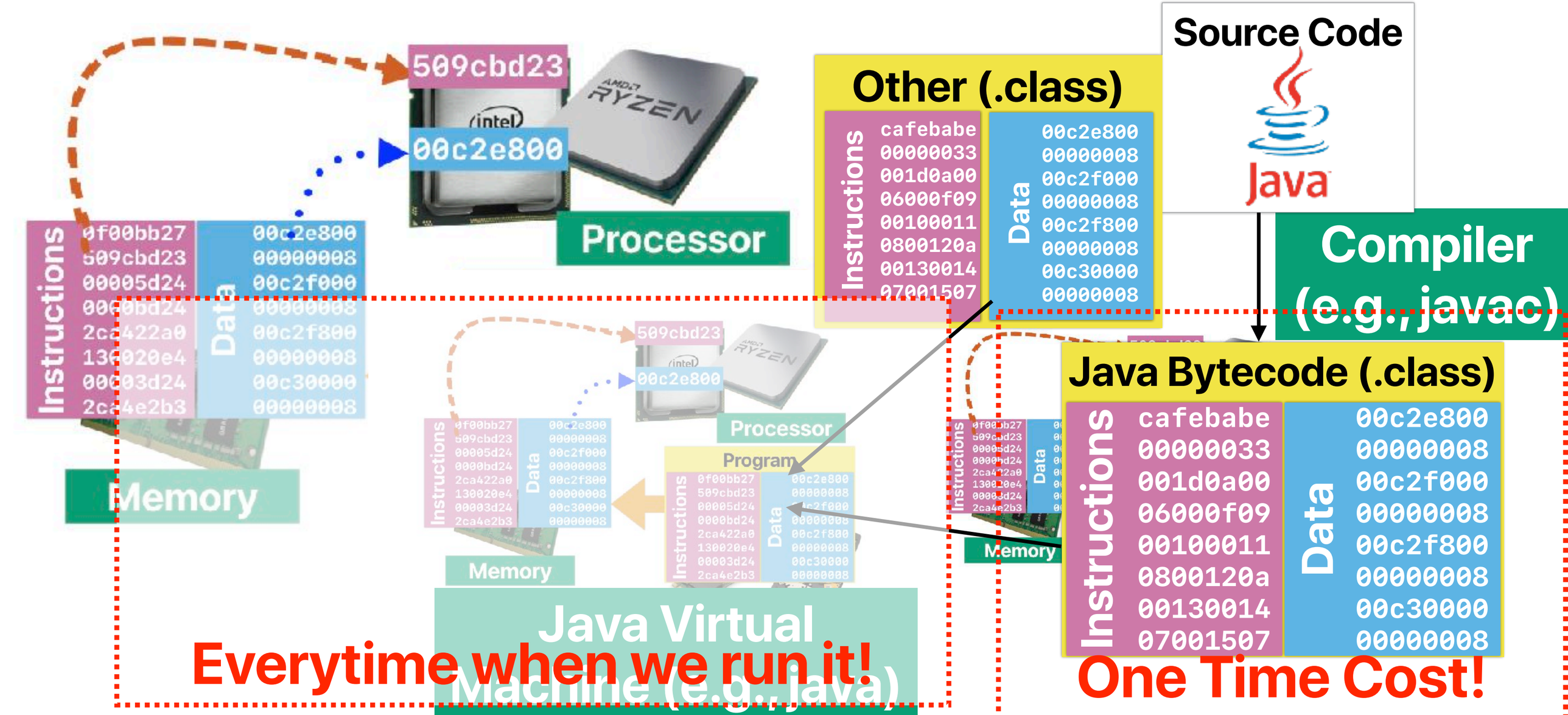
- How many instructions are there in “Hello, world!”

	Instruction count	LOC	Ranking
C	600k	6	1
C++	3M	6	2
Java	~145M	8	5
Perl	~12M	4	3
Python	~33M	1	4
GO (Interpreter)	~1200M	1	6
GO (Compiled)	~1.7M	1	
Rust	~1.4M	1	

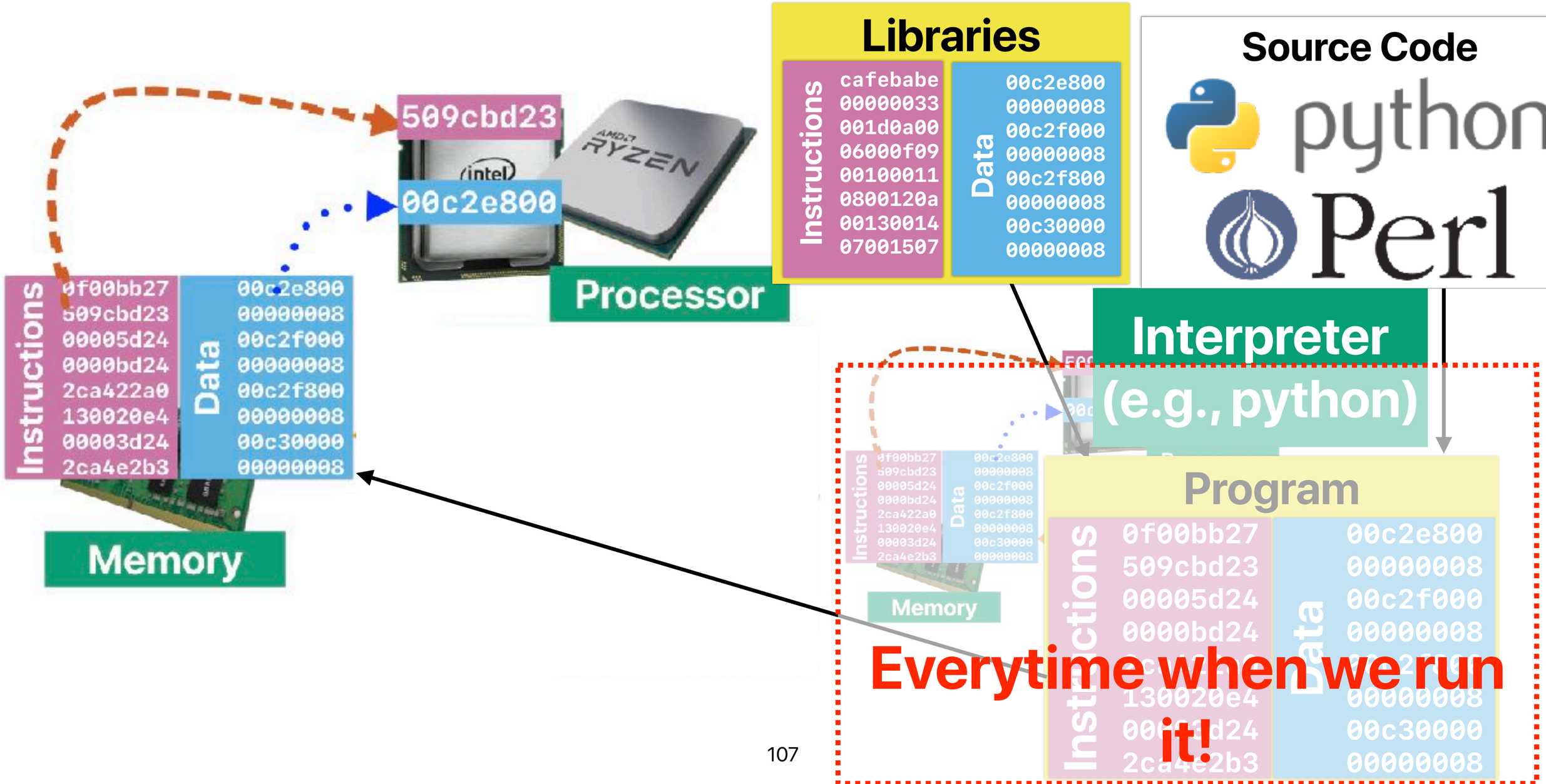
Recap: How my "C code" becomes a "program"



Recap: How my "Java code" becomes a "program"



Recap: How my "Python code" becomes a "program"



Takeaways: What matters?

- Programmers can control all three factors in the classic performance equation
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!

Takeaways: What matters?

- Programmers can control all three factors in the classic performance equation
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!
- Compiler optimization can help — only if programmers write code in a way facilitating optimizations!

How about complexity?

How about “computational complexity”

- Algorithm complexity provides a good estimate on the performance if —
  - Every instruction takes exactly the same amount of time
  - Every operation takes exactly the same amount of instructions

These are unlikely to be true

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**

Takeaways: What matters?

- Programmers can control all three factors in the classic performance equation
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!
- Compiler optimization can help — only if programmers write code in a way facilitating optimizations!
- Complexity does not provide good assessment on real machines due to the idealized assumptions

Quantitive Analysis of “Better”

Speedup

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

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

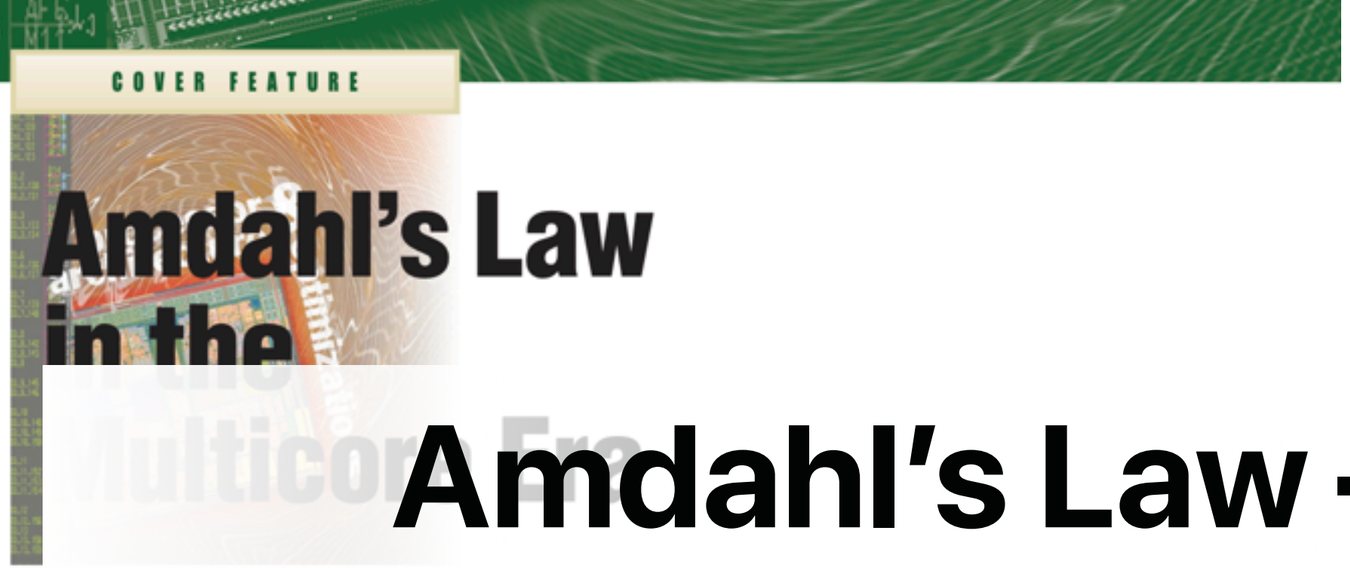
- The speedup of Y over X

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

Takeaways: What matters?

- Programmers can control all three factors in the classic performance equation
- Different programming languages can generate machine operations with different orders of magnitude performance — programmers need to make wise choice of that!
- Compiler optimization can help — but programmers need to write code in a way facilitate optimizations!
- Complexity does not provide good assessment on real machines due to the idealized assumptions
- The only definition of Y is Speedup times faster than X —

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



Amdahl's Law — and It's Implication in the Multicore Era

Augmenting Amdahl's law with a corollary for multicore hardware makes it relevant to future generations of chips with multiple processor cores. Obtaining optimal multicore performance will require further research in both extracting more parallelism and making sequential cores faster.

Mark D. Hill, University of Wisconsin-Madison
Michael R. Marty, Google

In IEEE Computer, vol. 41, no. 7

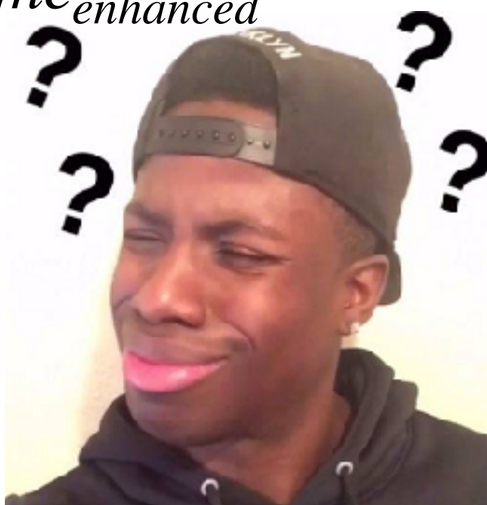
Amdahl's Law

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

f — The fraction of time in the original program

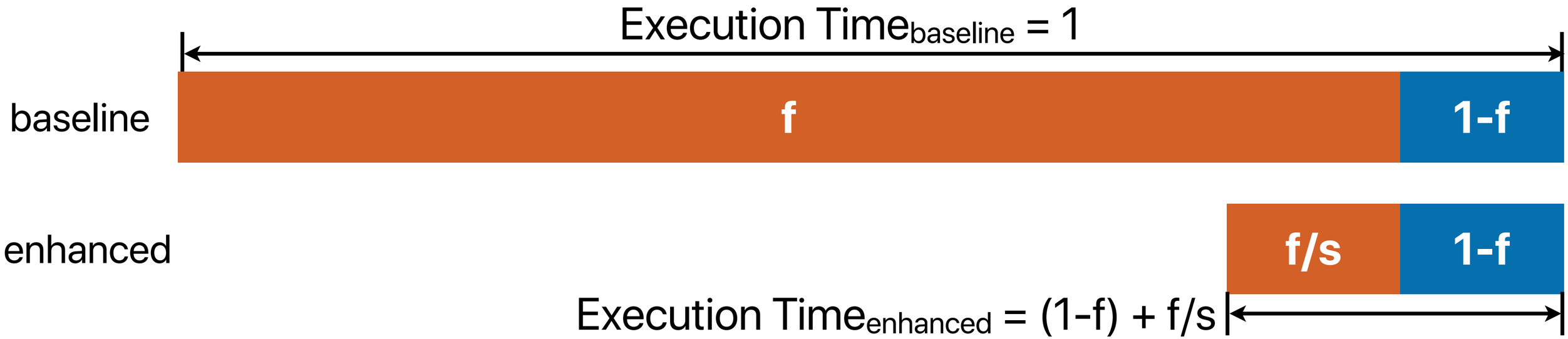
s — The speedup we can achieve on f

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



Amdahl's Law

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



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

Amdahl's Law on Multiple Optimizations

- We can apply Amdahl's law for multiple optimizations
- These optimizations must be dis-joint!
  - If optimization #1 and optimization #2 are dis-joint:

$$Speedup_{enhanced}(f_{Opt1}, f_{Opt2}, s_{Opt1}, s_{Opt2}) = \frac{1}{(1 - f_{Opt1} - f_{Opt2}) + \frac{f\_Opt1}{s\_Opt1} + \frac{f\_Opt2}{s\_Opt2}}$$

  - If optimization #1 and optimization #2 are not dis-joint:

$$Speedup_{enhanced}(f_{OnlyOpt1}, f_{OnlyOpt2}, f_{BothOpt1Opt2}, s_{OnlyOpt1}, s_{OnlyOpt2}, s_{BothOpt1Opt2}) = \frac{1}{(1 - f_{OnlyOpt1} - f_{OnlyOpt2} - f_{BothOpt1Opt2}) + + \frac{f\_BothOpt1Opt2}{s\_BothOpt1Opt2} + \frac{f\_OnlyOpt1}{s\_OnlyOpt1} + \frac{f\_OnlyOpt2}{s\_OnlyOpt2}}$$

Amdahl's Law Corollary #1

- The maximum speedup is bounded by

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

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

NEWS

Intel kills the remnants of Optane memory

The speed-boosting storage tech was already on the ropes.

By Michael Crider
Staff Writer, PCWorld | JUL 29, 2022 6:59 AM PDT

Image: Intel

Corollary #1 on Multiple Optimizations

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

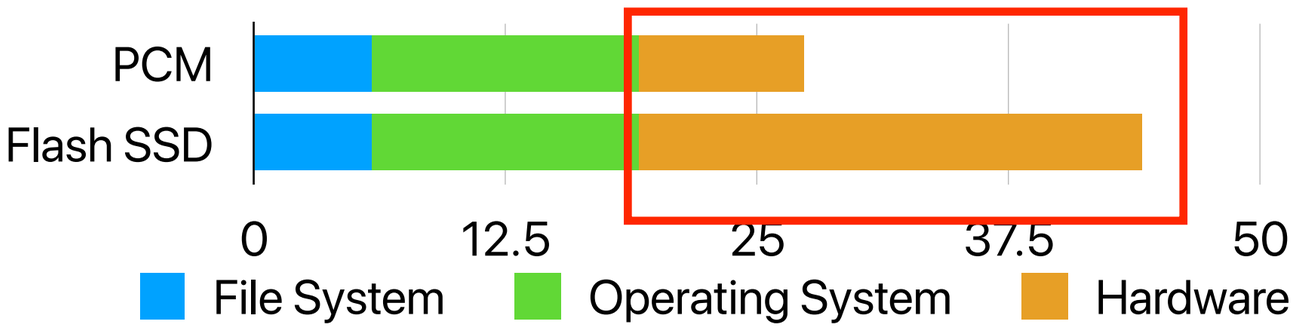
$$Speedup_{max}(f_1, \infty) = \frac{1}{(1 - f_1)}$$
$$Speedup_{max}(f_2, \infty) = \frac{1}{(1 - f_2)}$$
$$Speedup_{max}(f_3, \infty) = \frac{1}{(1 - f_3)}$$
$$Speedup_{max}(f_4, \infty) = \frac{1}{(1 - f_4)}$$

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.

Speedup further!

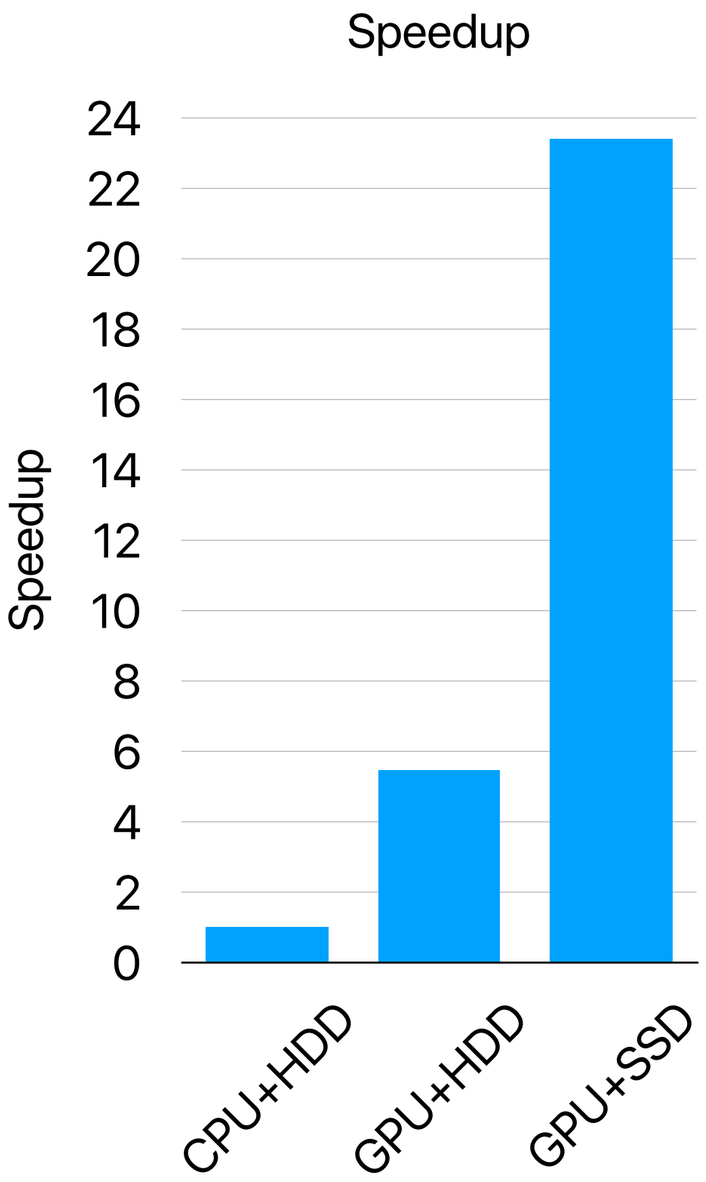


With optimization, the common becomes uncommon.

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

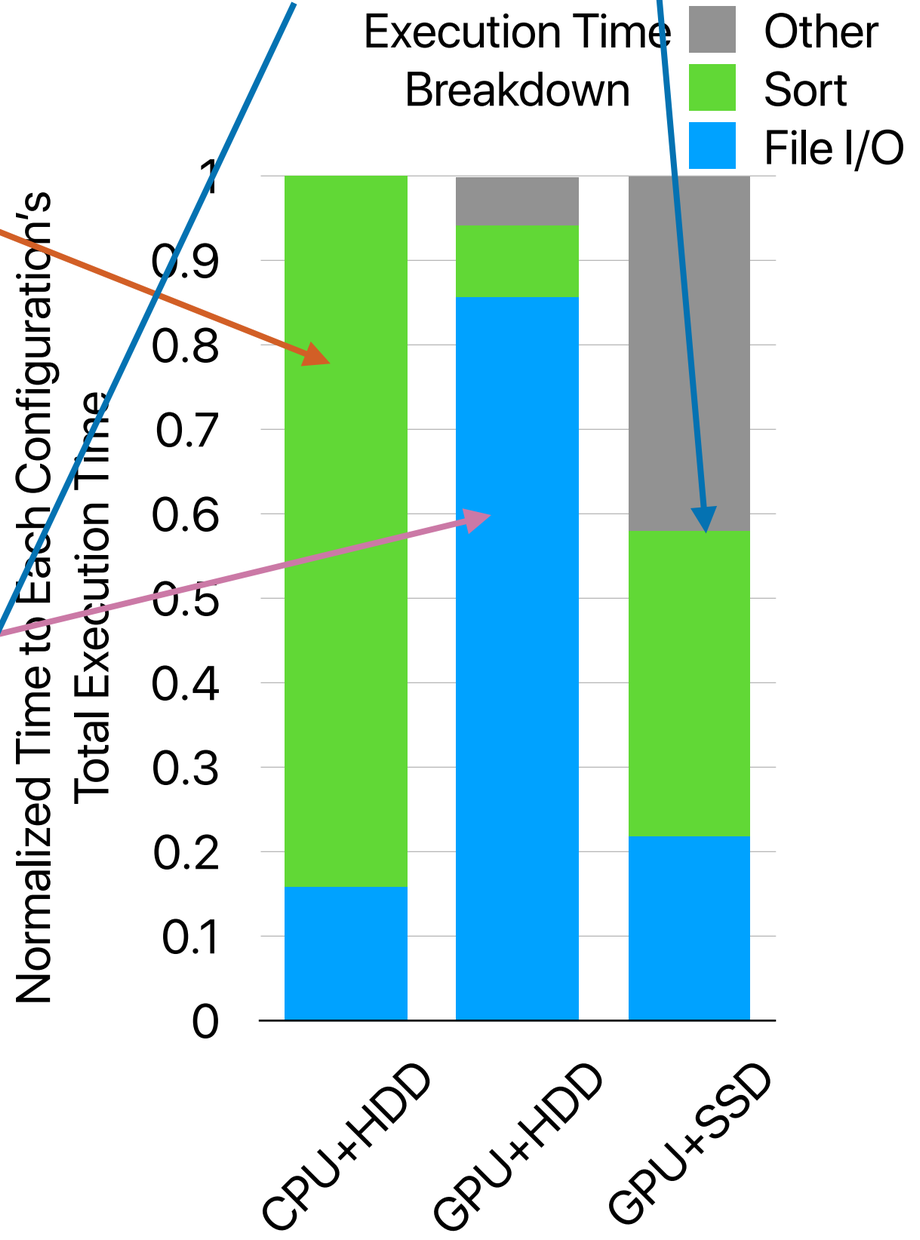
Demo — sort



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)

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

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

Merge Sort

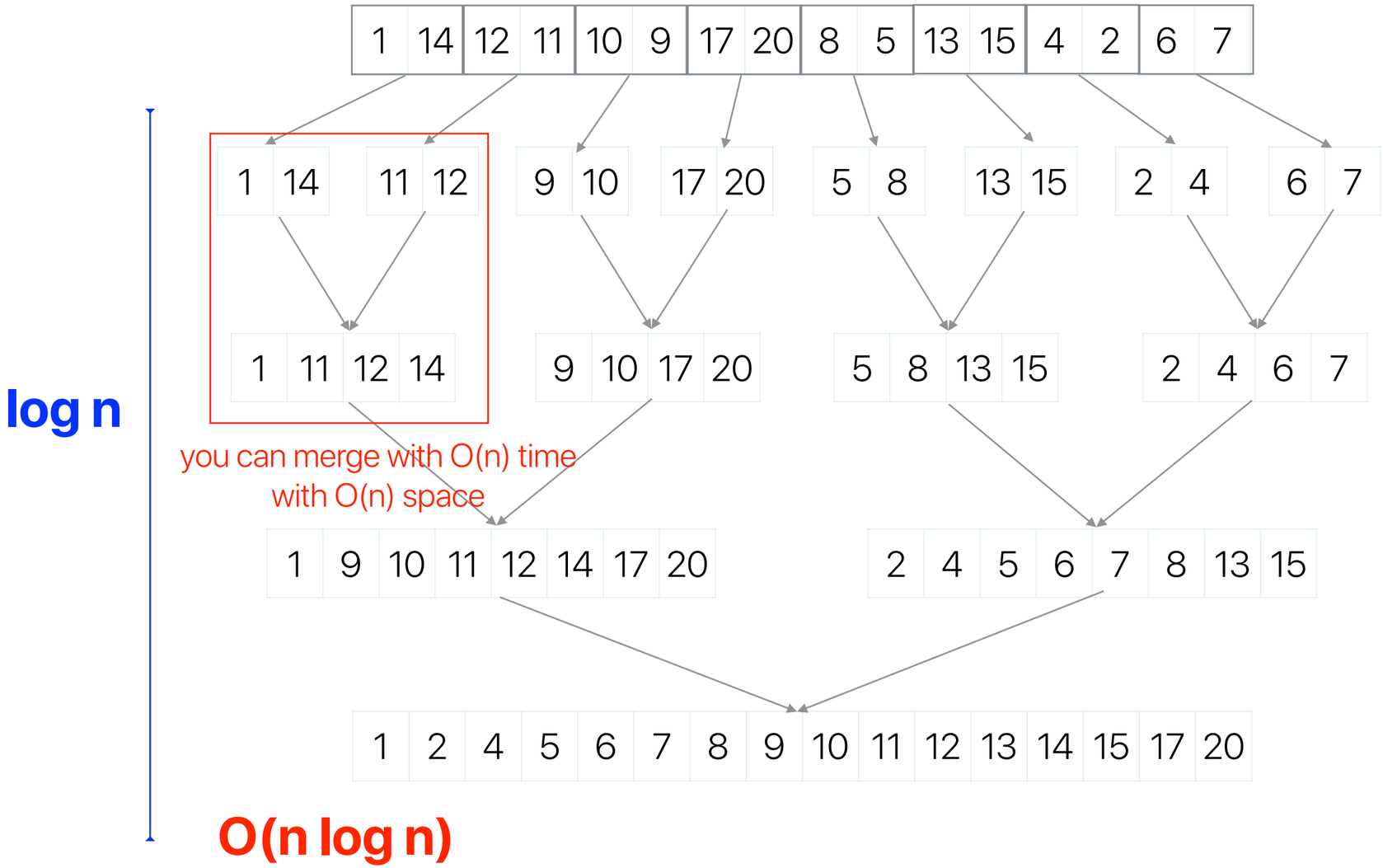
$O(nlog_2n)$

Bitonic Sort

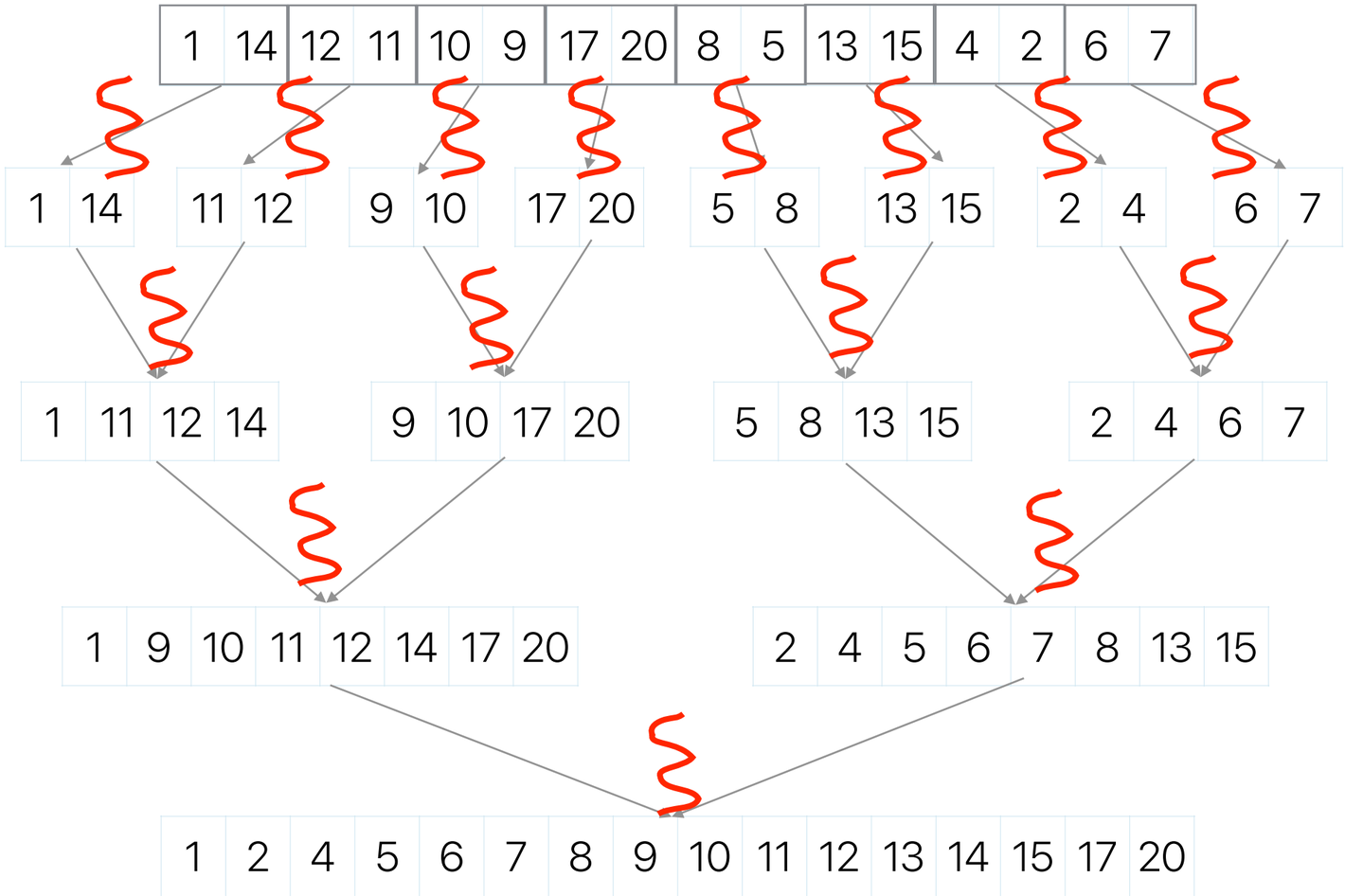
$O(nlog_2^2n)$

```
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


1 14 12 11 10 9 17 20 8 5 13 15 4 2 6 7
1 14 11 12 9 10 17 20 5 8 13 15 2 4 6 7
1 11 12 14 9 10 17 20 5 8 13 15 2 4 6 7
you can merge with O(n) time
with O(n) space
1 9 10 11 12 14 17 20 2 4 5 6 7 8 13 15
1 2 4 5 6 7 8 9 10 11 12 13 14 15 17 20
log n
O(n log n)

Parallel merge sort



What's the speedup of merge sort using Amdahl's Law

The degree of parallelism is $1, 2, 4, \ldots, \frac{n}{2}$

at step $1, 2, 3, \ldots, log_2(n)$

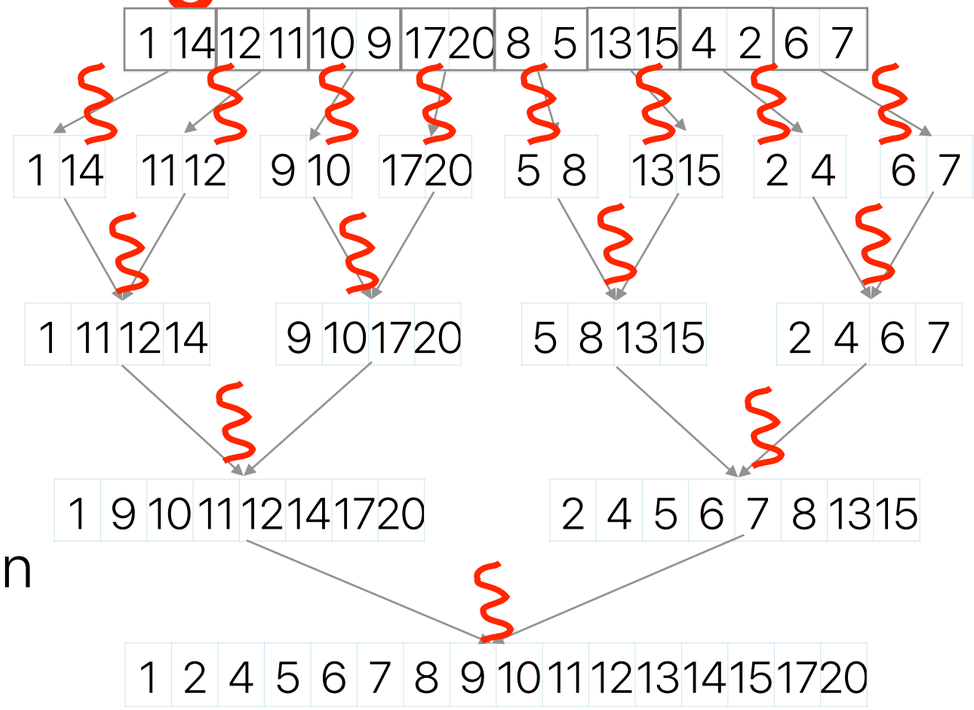
The ideal speedup of each step is $1, 2, 4, \ldots, \frac{n}{2}$ or say

$1, 2, 4, \ldots, 2^{log_2(n)-1}$ if we have **unlimited** parallelism

if we assume equal amount of time in each step in the baseline,

each step is going to take $\frac{1}{lg(n)}$ portion of time in the baseline, an

the first $\frac{1}{lg(n)}$ is not parallelizable (i.e., $(1-x) = \frac{1}{lg(n)}$)

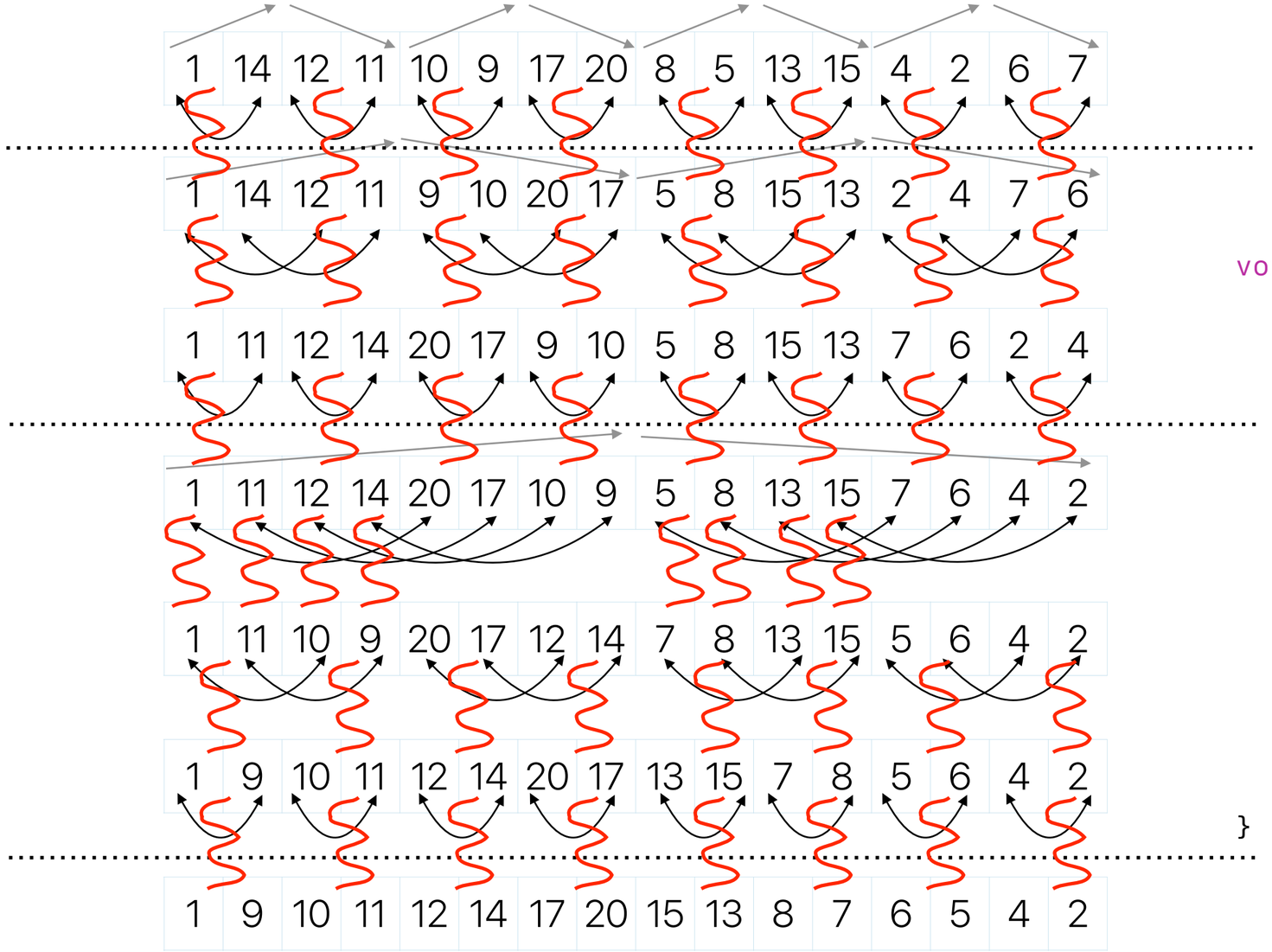


So the Amdahl's Law's evaluation will become

$$\frac{1}{\frac{1}{lg(n)} + \frac{1}{lg(n)\times 2} + \frac{1}{lg(n)\times 4} + \ldots + \frac{1}{lg(n)\times 2^{lg(n)-1}}} = \frac{1}{\frac{1}{lg(n)}(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{lg(n)-1}})}$$

$$= \frac{1}{\frac{1}{lg(n)}(1 + 1 - \frac{1}{2^{lg(n)-1}})} = \frac{1}{\frac{2}{lg(n)}} = \frac{lg(n)}{2}$$

Bitonic sort

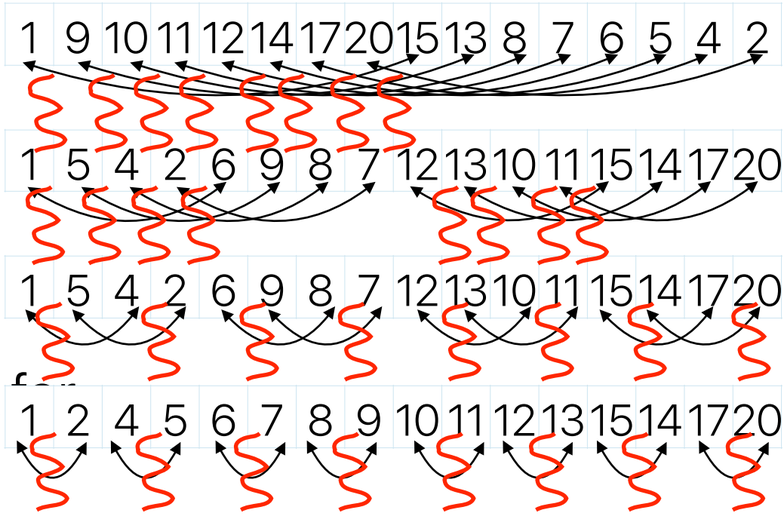



```
void BitonicSort() {

    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);
                }
            }
        }
    }
}
```

What's the speedup of bitonic sort using Amdahl's Law

The degree of parallelism is always $\frac{n}{2}$

at step $1,\ 2,\ 3,\ \ldots,\ log_2^2(n)$



The ideal speedup of each step is $\frac{n}{2}$ if we have **unlimited** parallelism & for each step of bitonic sort. However, bitonic sort will have lg(n)x more steps than merge sort.

If the baseline is merge sort — the speedup of using bitonic sort is

$$\frac{1}{\frac{1 \times lg(n)}{\frac{n}{2}}} = \frac{n}{2lg(n)} > \frac{lg(n)}{2}$$

What if we have only p processors?

For **bitonic sort**,The degree of parallelism is always $\frac{n}{2}$

at step $1, 2, 3, \ldots, log_2^2(n)$, but we have only p processors...

The theoretical speedup of each step is $max(\frac{n}{2}, p) = p$ since n is very likely to

be larger than **p** & assume equal amount of time in each step in the baseline

So the Amdahl's Law's evaluation will become $\frac{1}{\frac{p}{lg(n)}} = \frac{p}{lg(n)}$ $= \frac{1024}{30} = 34.1333333$

What about merge sort? $= \frac{1}{\frac{1}{lg(n)}(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{p})}$

What if p=1024, n=1M=2^{30}? $= \frac{1}{\frac{1}{30}(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{1024} + 20 \times \frac{1}{30})} = 14.862119$

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**

Is it the end of computational complexity?

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
  - 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"

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

$$\text{Execution Time}_{\text{enhanced}} = \frac{(1-f)}{perf(r)} + \frac{f}{n}$$

Corollary #6: 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 (i.e., an overhead that is 9x longer than the original execution time)…

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

Takeaways: Are we there yet?

- Definition of "Speedup of Y over X" or say Y is n times faster than X:

$$speedup_{Y\_over\_X} = n = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- Amdahl's Law — $Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$
  - Corollary 1 — each optimization has an upper bound $Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$
  - Corollary 2 — make the common case (the most time consuming case) fast!
  - Corollary 3: Optimization has a moving target

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

  - Corollary 4: Exploiting more parallelism from a program is the key to performance gain in modern architectures $Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$
  - Corollary 5: Single-core performance still matters $Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$
  - Corollary 6: Don't hurt the non-common case too much $Speedup_{enhanced}(f, s, r) = \frac{1}{(1-f) + perf(r) + \frac{f}{s}}$

Announcement

- Programming assignment 1 due this Thursday
  - We cannot help you at the last minute — please start early
- Assignment 1 due next Thursday
- Reading quiz due next Tuesday before the lecture — we will drop two of your least performing reading quizzes
- Check our website for slides, Gradescope for assignments, discord for discussions
- Youtube channel for lecture recordings: https://www.youtube.com/c/ProfUsagi/playlists

Computer
Science &
Engineering

203

つづく