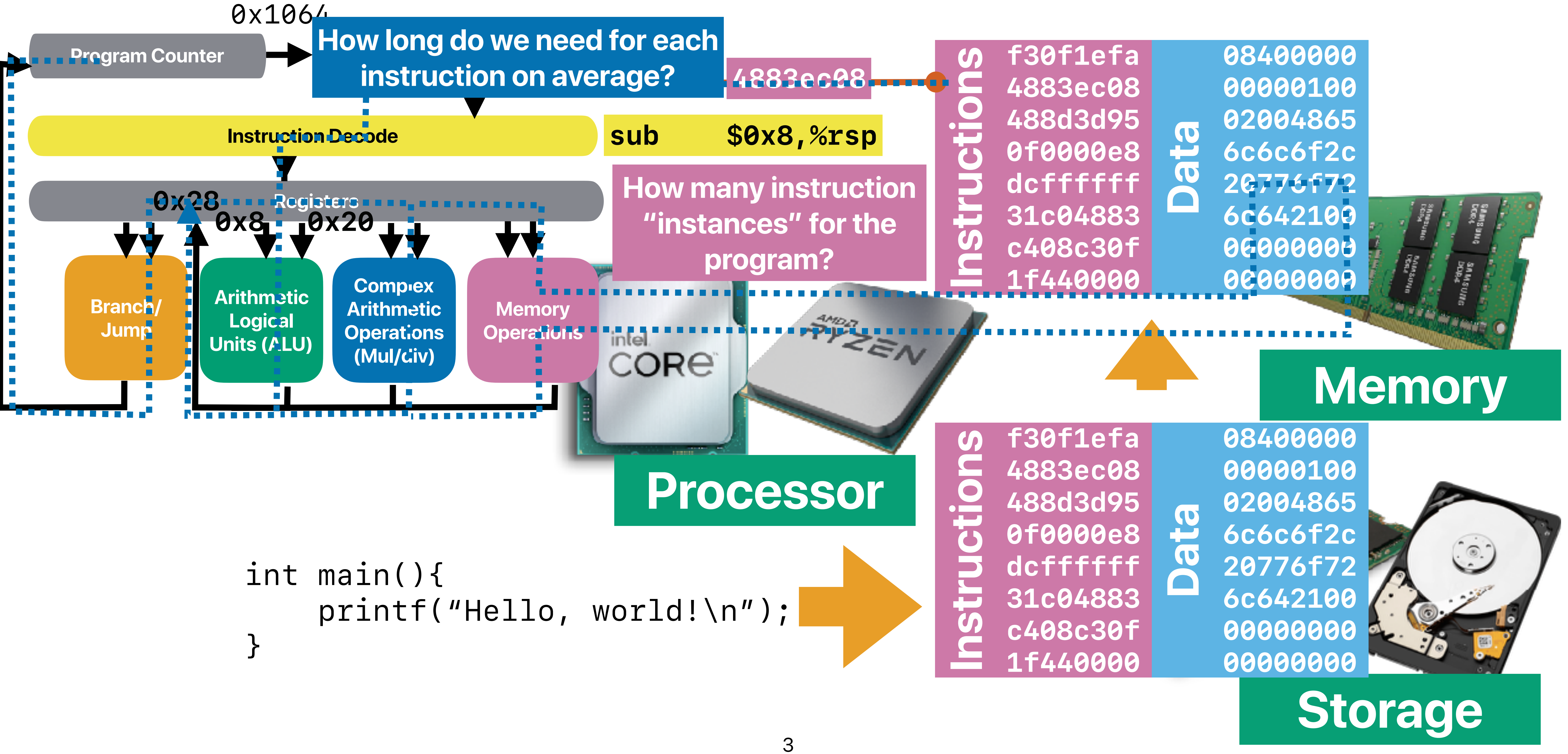
Performance (3): (Don’t) Be a Great Pretender

Hung-Wei Tseng

Execution time of a program in the von Neumann model




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

Takeaways: find the right thing to do

- 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!
    $Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$
    $Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$
    $Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$
    $Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$
  - Corollary 3 — Optimization has a moving target
  - 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
    $Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1-f_{parallelizable})}$

8cx
Qualcomm snapdragon

Extreme Multitasking Performance

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

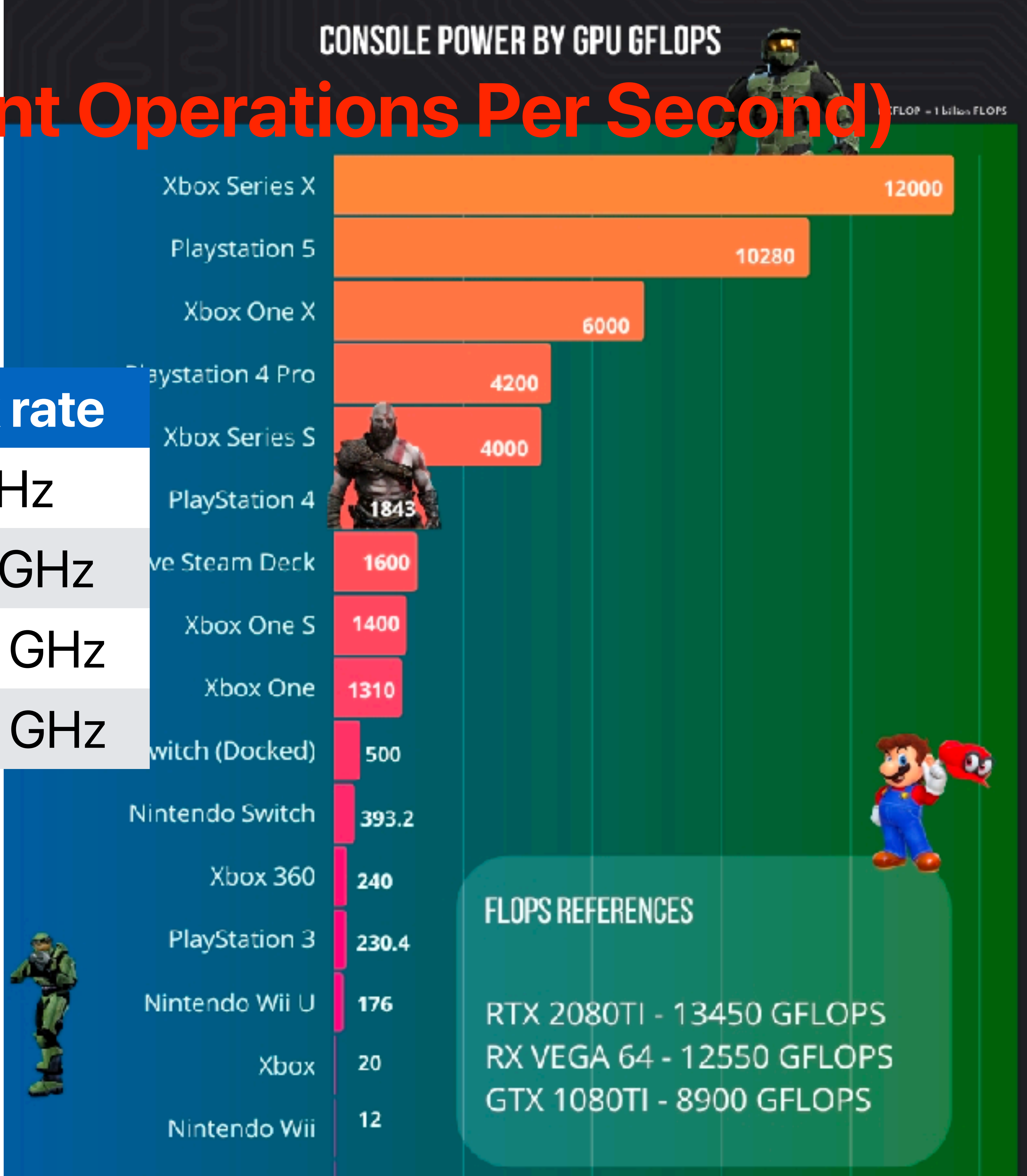
12 Ways to Fool the Masses When Giving Performance Results on Parallel Computers

- 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
- 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

Let’s talk about FLOPS
— PFLOPS, TFLOPS, GFLOPS

TFLOPS (Tera FLoating-point Operations Per Second)

	TFLOPS	clock rate
Switch	3.09	1 GHz
PlayStation 5 Pro	16.7	2.35 GHz
XBox Series X	12	1.825 GHz
GeForce RTX 3090	104.8	1.395 GHz



FLOPS are frequently cited

Figure 2. NVIDIA Blackwell Architecture's Technological Breakthroughs

A New Class of AI GPU

Built with 208 billion transistors, more than 2.5x the amount of transistors in NVIDIA Hopper GPUs, and using TSMC's 4NP process tailored for NVIDIA, Blackwell is the largest GPU ever built. NVIDIA Blackwell achieves the highest compute ever on a single chip, 20 petaFLOPS.

This architecture can incorporate a significant amount of computing power by merging

7

NVIDIA Blackwell Architecture Technical Brief

https://resources.nvidia.com/en-us-blackwell-architecture

https://www.pollev.com/hungweitseng close in 1:30

How useful is FLOPS (FLoating-point Operations Per Second)?

- If we're given the FLOPS of the underlying GPU of the machine, how many situations below can the FLOPS be representative to the real performance?
  ① The FLOPS remains the same if we change the dataset for the same program
  ② The FLOPS remains the same if we change the data type to double
  ③ The FLOPS remains the same if we run a different the algorithm
  ④ Running the same program on a hardware with higher FLOPS will lead to better performance

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

How useful is FLOPS (FLoating-point Operations Per Second)?

- If we're given the FLOPS of the underlying GPU of the machine, how many situations below can the FLOPS be representative to the real performance?
  ① The FLOPS remains the same if we change the dataset for the same program
  ② The FLOPS remains the same if we change the data type to double
  ③ The FLOPS remains the same if we run a different the algorithm
  ④ Running the same program on a hardware with higher FLOPS will lead to better performance
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

TFLOPS (Tera FLoating-point Operations Per Second)

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

Let's measure the TFLOPS of matrix multiplications

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

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

$$TFLOPS = \frac{2^{34}}{ET_{seconds} \times 2^{12}}$$

Floating point operations (FLOP"s"):

$$M \times N \times K \times 2$$

Given $M = N = K = 2048$

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

TFLOPS (Tera FLoating-point Operations Per Second)

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

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!

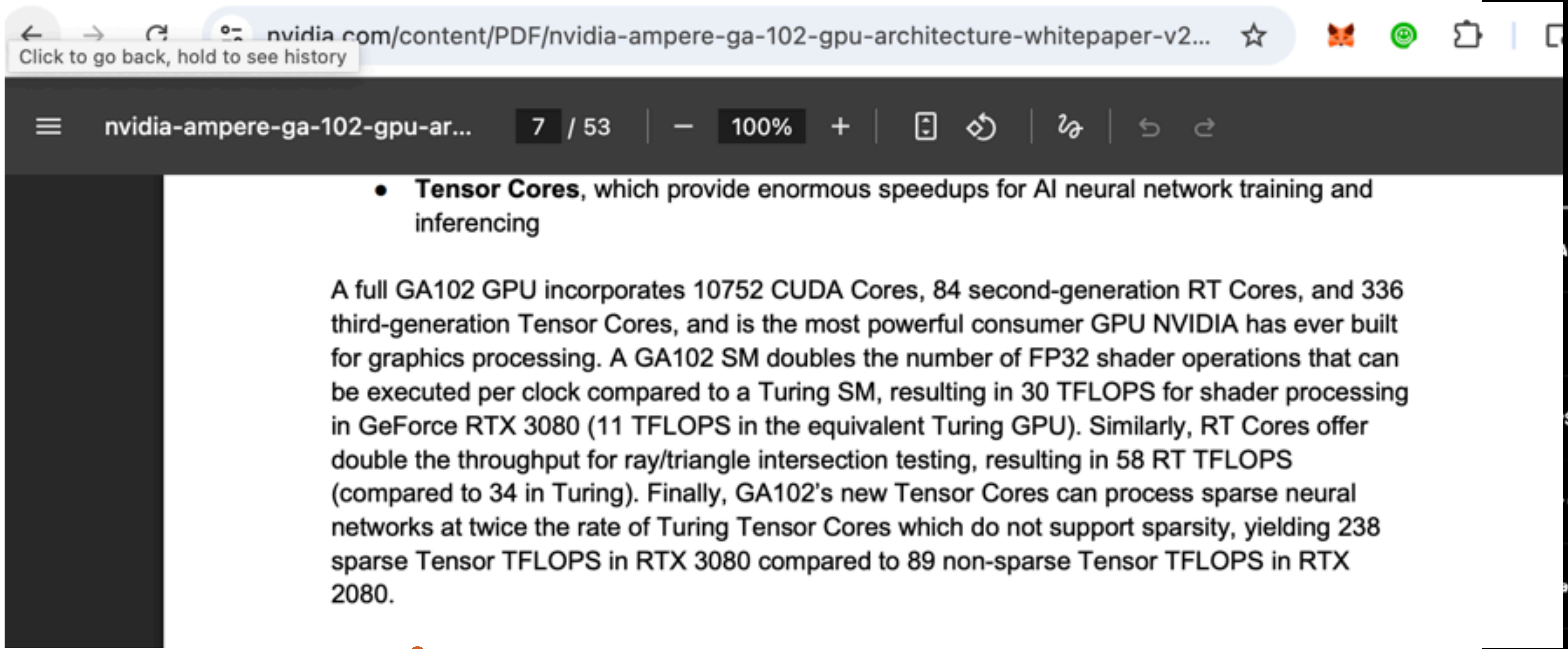
- If we have more iterations? Larger datasets? — potentially changes the IC
- What if the hardware trade (cheat) performance with accuracy?
- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- If floating point operations are not critical in the target application?

How useful is FLOPS (FLoating-point Operations Per Second)?

- If we're given the FLOPS of the underlying GPU of the machine, how many situations below can the FLOPS be representative to the real performance?
  ① The FLOPS remains the same if we change the dataset for the same program — IC changed
  ② The FLOPS remains the same if we change the data type to double — CPI changed
  ③ The FLOPS remains the same if we run a different the algorithm — IC changed
  ④ Running the same program on a hardware with higher FLOPS will lead to better performance — Only true if the program is floating point intensive

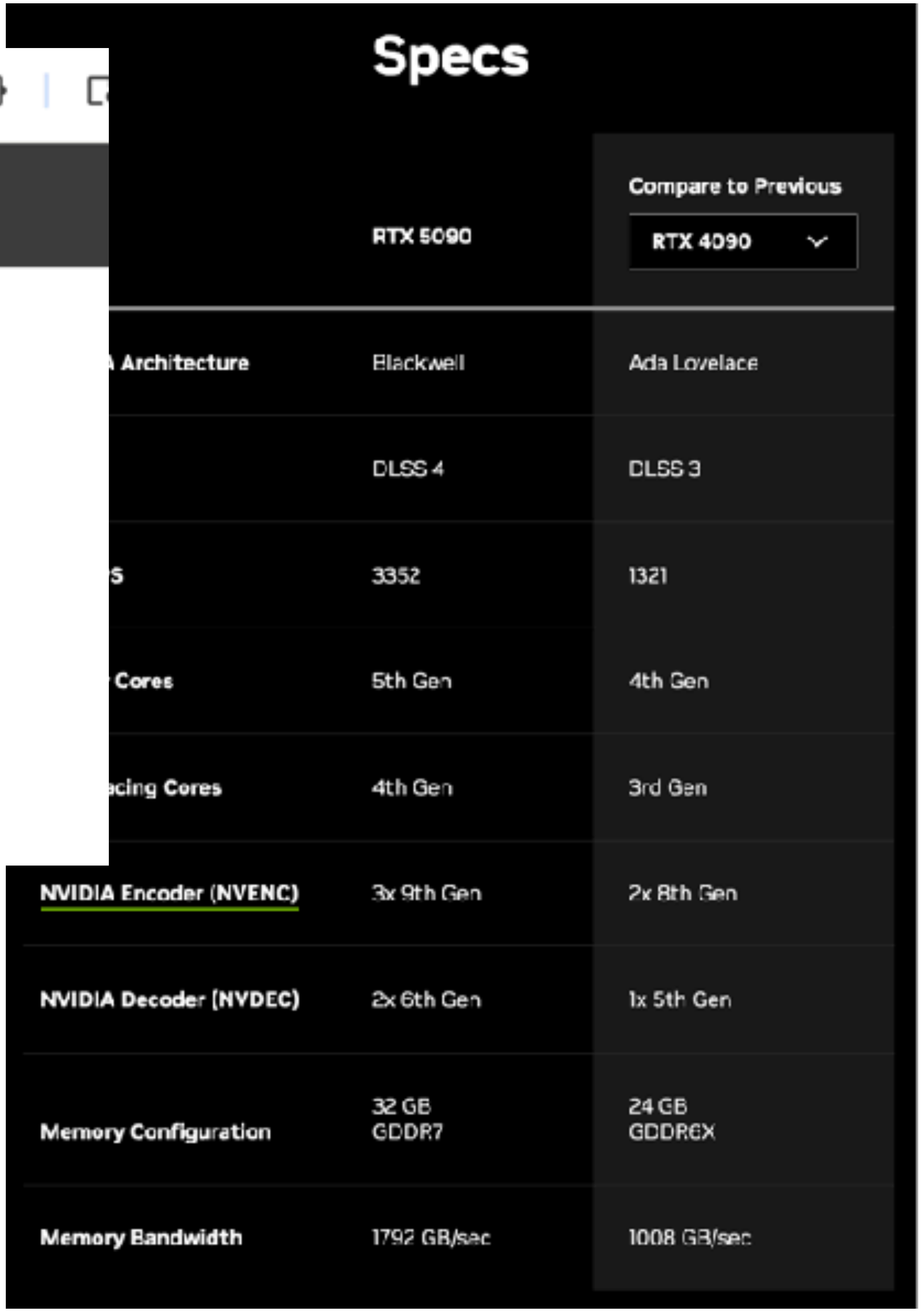
A. $\boxed{0}$
B. 1
C. 2
D. 3
E. 4

OPS are now frequently cited



- **Tensor Cores**, which provide enormous speedups for AI neural network training and inferencing

A full GA102 GPU incorporates 10752 CUDA Cores, 84 second-generation RT Cores, and 336 third-generation Tensor Cores, and is the most powerful consumer GPU NVIDIA has ever built for graphics processing. A GA102 SM doubles the number of FP32 shader operations that can be executed per clock compared to a Turing SM, resulting in 30 TFLOPS for shader processing in GeForce RTX 3080 (11 TFLOPS in the equivalent Turing GPU). Similarly, RT Cores offer double the throughput for ray/triangle intersection testing, resulting in 58 RT TFLOPS (compared to 34 in Turing). Finally, GA102's new Tensor Cores can process sparse neural networks at twice the rate of Turing Tensor Cores which do not support sparsity, yielding 238 sparse Tensor TFLOPS in RTX 3080 compared to 89 non-sparse Tensor TFLOPS in RTX 2080.



Specs

	RTX 5090	Compare to Previous RTX 4090
Architecture	Blackwell	Ada Lovelace
	DLSS 4	DLSS 3
s	3352	1321
Cores	5th Gen	4th Gen
cing Cores	4th Gen	3rd Gen
NVIDIA Encoder (NVENC)	3x 9th Gen	2x 8th Gen
NVIDIA Decoder (NVDEC)	2x 6th Gen	1x 5th Gen
Memory Configuration	32 GB GDDR7	24 GB GDDR6X
Memory Bandwidth	1792 GB/sec	1008 GB/sec

Does it solve the problem?

https://resources.nvidia.com/en-us-blackwell-architecture

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

- Quote only 32-bit performance results, not 64-bit results. **— FLOPS changes as datatype changes**
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application **— Not the whole program uses FLOPs intensively**
- 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 **— FLOPS changes as dataset size changes**
- 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 **— FLOPS changes as algorithm changes**
- 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

Take-aways: being an honest engineer

- FLOPS can only be meaningful in very limited cases as FLOPS does not consider all aspects of the performance equation

Case study: AI inference

forbes.com/sites/arm/2023/12/04/the-compute-four

Click to go back, hold to see history

Forbes

AI training vs. inference

Much of the news coverage recently has been on LLMs, their development and their training – and the high cost and energy consumption required to do so. A recent study[1] estimated that Chat GPT3, comprised of 175 billion parameters, needed 1,287 MWh to train, and also emitted 552 tons of CO_2. This is roughly the equivalent to driving a car 1.3 million miles – but in one hour!

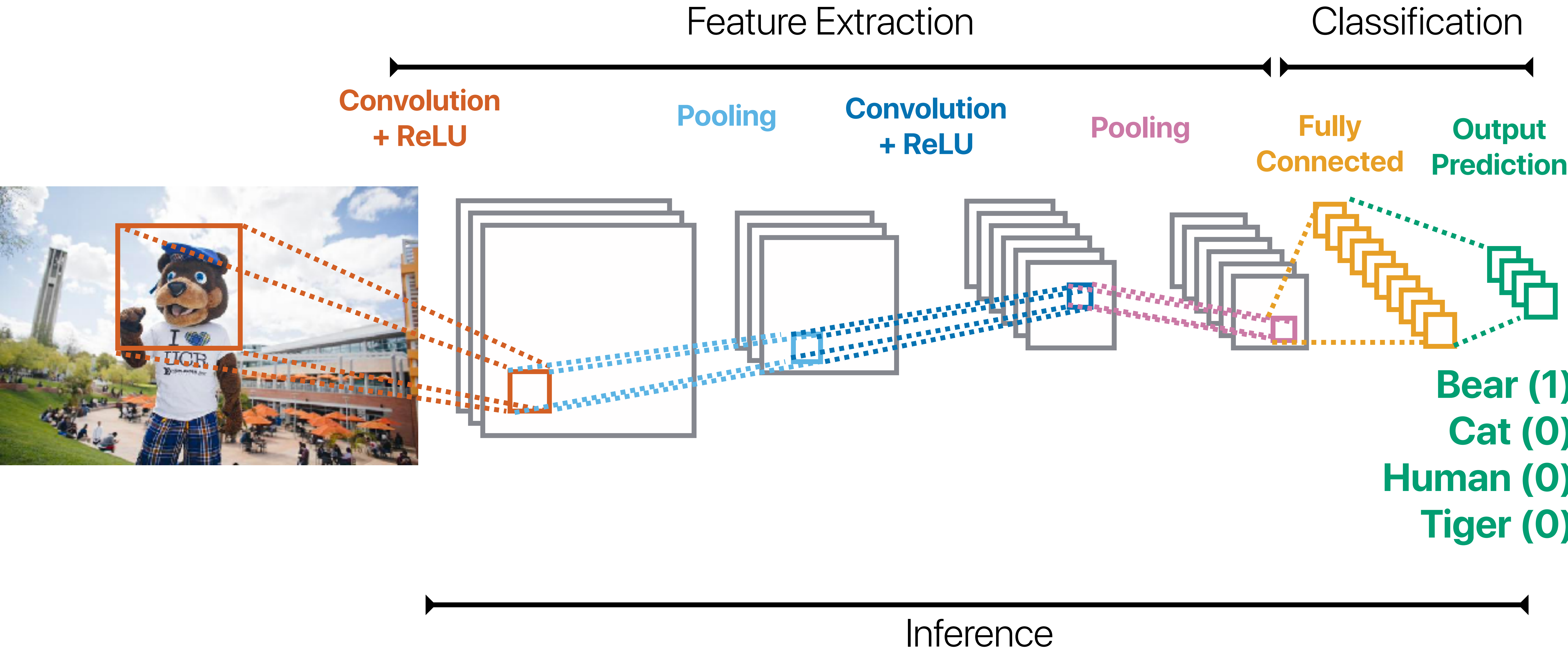
But those huge training workloads running in data centers represent just 15 percent of AI workloads today, according to Omdia's Data Center Compute Intelligence Service.[2]

The rest – 85 percent of all data center AI workloads – lies in AI inference, and that's not accounting for inference that's happening outside the data center on the edge. AI workloads like inference will remain a key workload in the cloud, but an increasing number will move to the edge as more efficient models continue to evolve. Inference, which is the process of using a trained model that has been deployed into a production environment to make predictions on new real-world

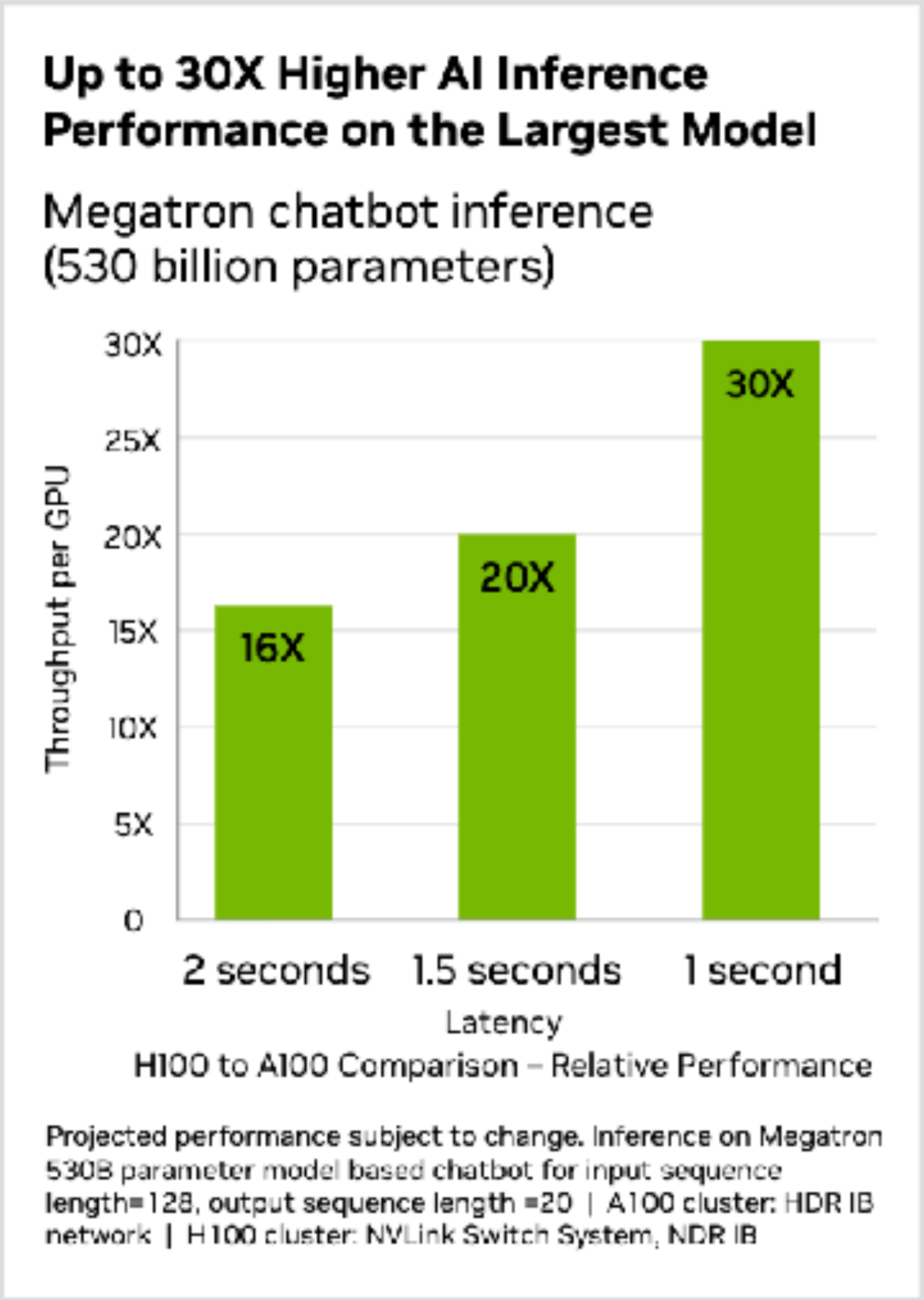
The rest – 85 percent of all data center AI workloads – lies in AI inference, and that's not accounting for inference that's happening outside the data center on the edge. AI workloads like inference will remain a key

https://www.forbes.com/sites/arm/2023/12/04/the-compute-foundation-powering-ais-present-and-future/

The Machine Learning Inference Pipeline



Inference per second (IPS) is increasingly popular



https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet

https://www.pollev.com/hungweitseng close in 1:30

How reflective is "inferences per second"

- Regarding inferences per second (IPS), please identify how many of the following statements are correct
  - ① IPS can change if the application changes the ML model used
  - ② IPS can become worse if the application adds more features to improve the quality of the answer or the precision
  - ③ IPS will improve if the system applies a hardware that offers higher FLOPS
  - ④ IPS remains the same if the system/application answers questions 20x slower but can answer 20x more questions in parallel
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

How reflective is “inferences per second”

- Regarding inferences per second (IPS), please identify how many of the following statements are correct
  ① IPS can change if the application changes the ML model used
  ② IPS can become worse if the application adds more features to improve the quality of the answer or the precision
  ③ IPS will improve if the system applies a hardware that offers higher FLOPS
  ④ IPS remains the same if the system/application answers questions 20x slower but can answer 20x more questions in parallel

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

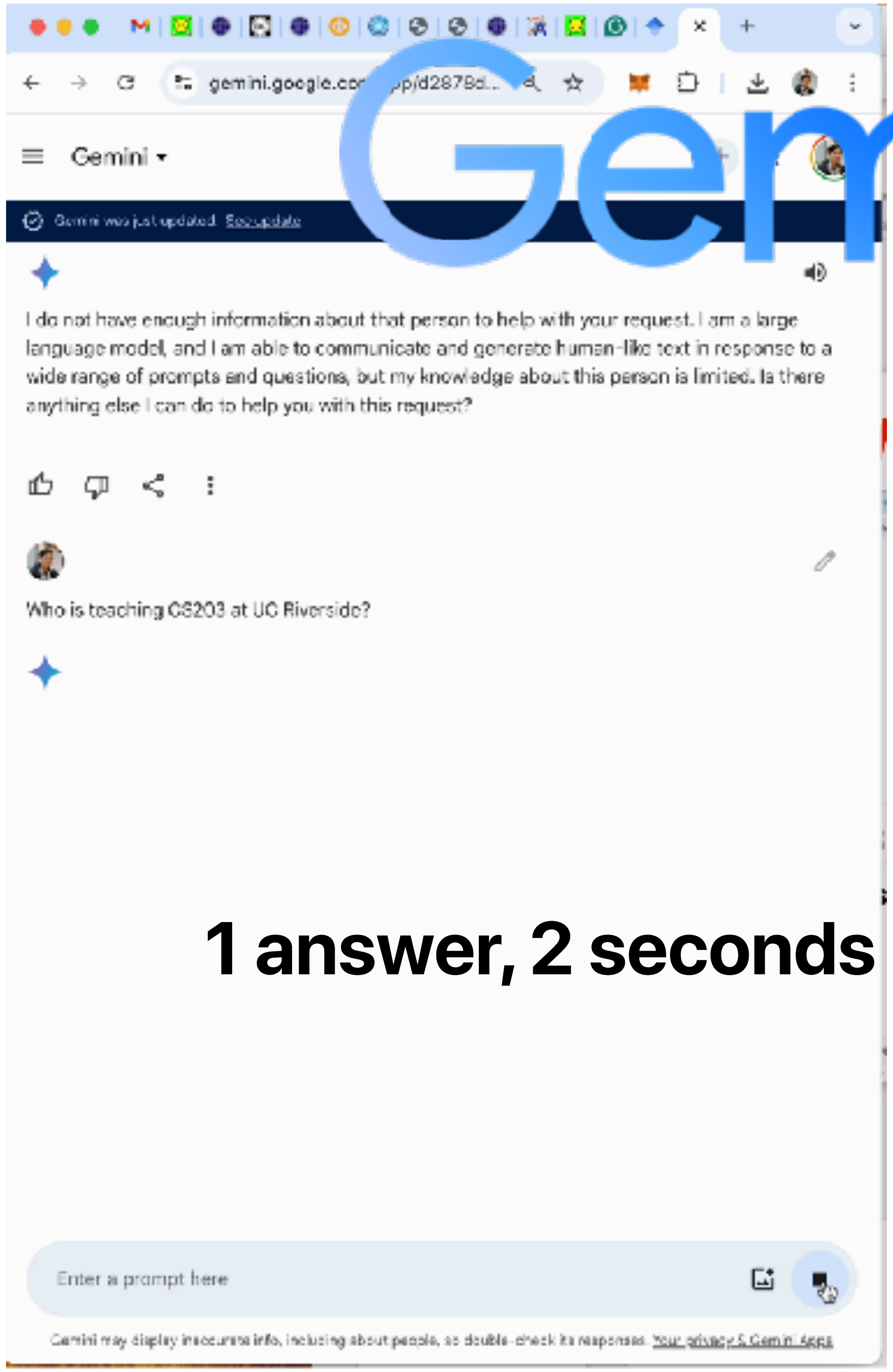
Inferences per second

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

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

What about inference per second?

1 answer, 2 seconds



1 answer, 2 seconds

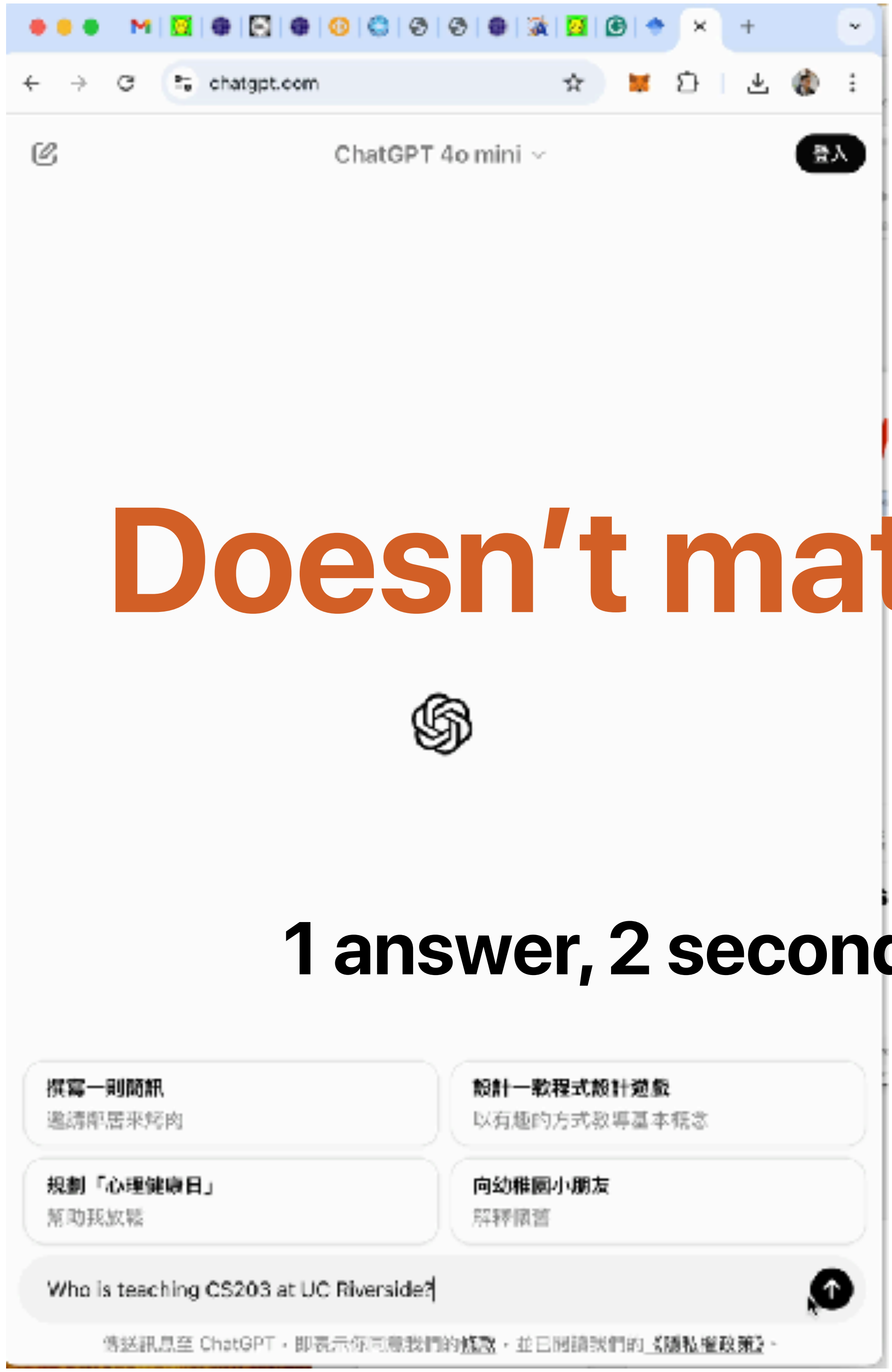
What's wrong with inferences per second?

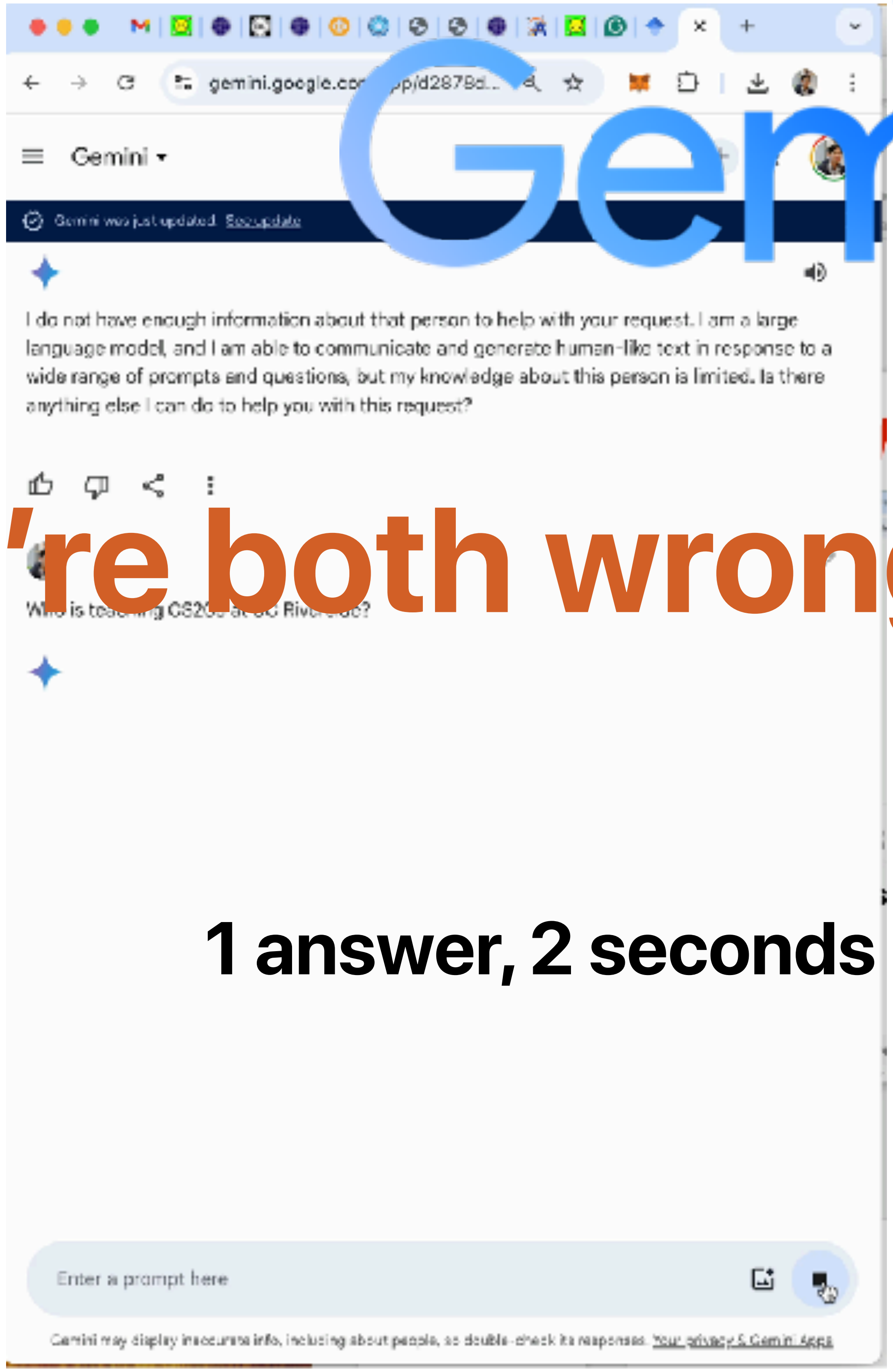
- There is no standard on how they inference — but these affect!
  - What model?
  - What dataset?
  - Quality?
- That's why Facebook is trying to promote an AI benchmark — MLPerf

- *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].

What about inference per second?





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

1 answer, 2 seconds

1 answer, 2 seconds

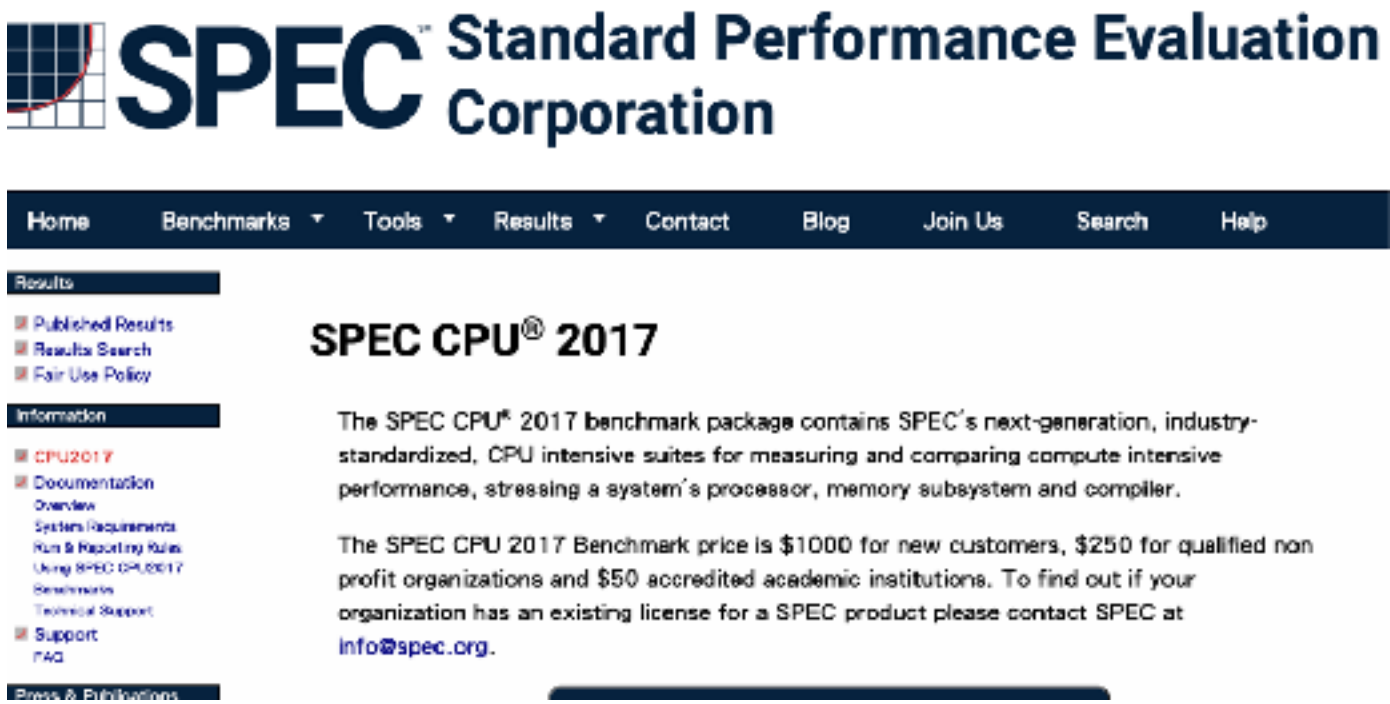
How reflective is "inference per second"

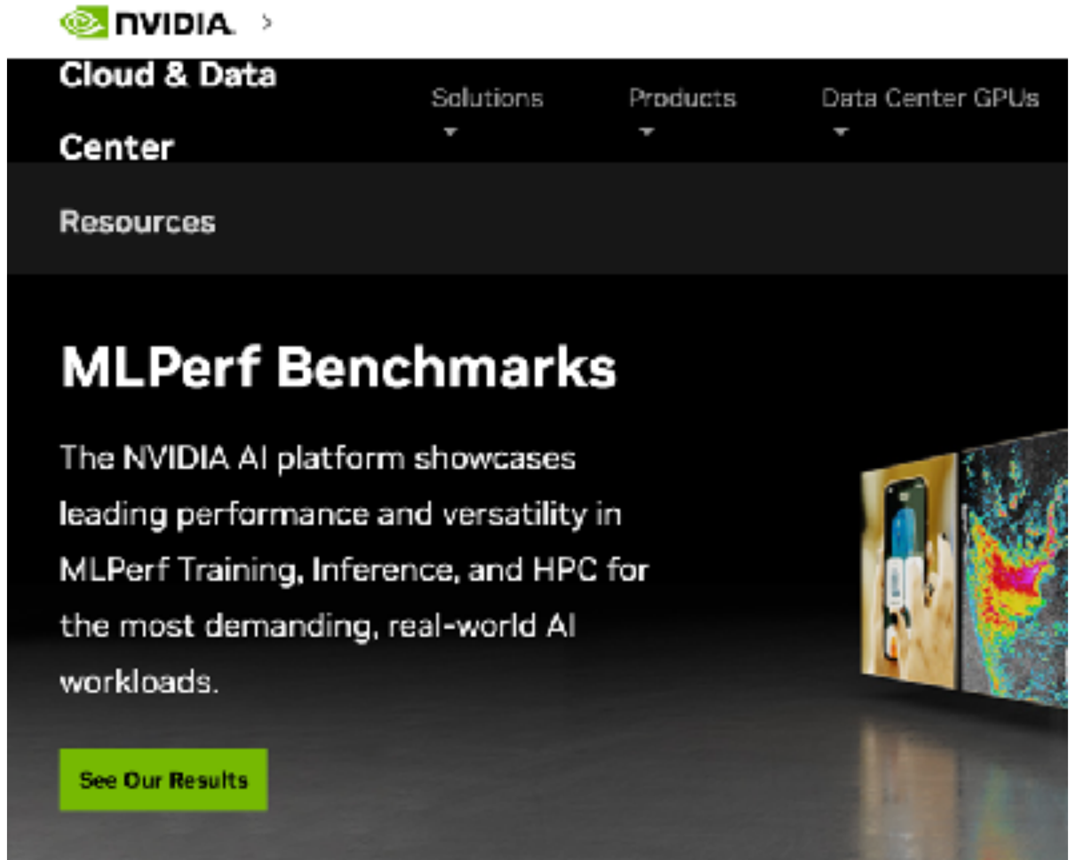
- Regarding inferences per second (IPS), please identify how many of the following statements are correct
  - ① IPS can change if the application changes the ML model used
  - ② IPS can become worse if the application adds more features to improve the quality of the answer or the precision
  - ③ IPS will improve if the system applies a hardware that offers higher FLOPS
  - What if the model uses integers? At lower precisions?
  - ④ IPS remains the same if the system/application answers questions 20x slower but can answer 20x more questions in parallel

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

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





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

- Quote only 32-bit performance results, not 64-bit results. — FLOPS changes as datatype changes
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application — Not the whole program uses FLOPs intensively
- 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 — FLOPS changes as dataset size changes
- Compare your results against scalar, unoptimized code on Crays — IPS cannot reflect if the quality changes
- 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 — FLOPS changes as algorithm changes
- 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 — IPS cannot reflect if the model/algorithm changes
- 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

Take-aways: being an honest engineer

- FLOPS can only be meaningful in very limited cases as FLOPS does not consider all aspects of the performance equation
- Without mentioning the specific conditions or changing more than one factors in the evaluations, the performance result is meaningless

https://www.pollev.com/hungweitseng close in 1:30

Why end-to-end latency, not FLOPS/IPS

- Regarding the following statements about the relationship between FLOPS and end-to-end latency, how many of them is/are correct
  ① Offloading the main computation kernel where the baseline spend the most time from CPU to another computing resource with higher FLOPS will improve the end-to-end latency
  ② If we place the file on a disk with higher bandwidth, we will spend less time in file access.
  ③ A computing device with higher parallelism (e.g., more cores) can potentially deliver higher FLOPS even though each parallel instance is slower
  ④ For real-time applications, we should prefer computing resources that deliver more results per second

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

Why end-to-end latency, not FLOPS/IPS

- Regarding the following statements about the relationship between FLOPS and end-to-end latency, how many of them is/are correct
  ① Offloading the main computation kernel where the baseline spend the most time from CPU to another computing resource with higher FLOPS will improve the end-to-end latency
  ② If we place the file on a disk with higher bandwidth, we will spend less time in file access.
  ③ A computing device with higher parallelism (e.g., more cores) can potentially deliver higher FLOPS even though each parallel instance is slower
  ④ For real-time applications, we should prefer computing resources that deliver more results per second
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

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

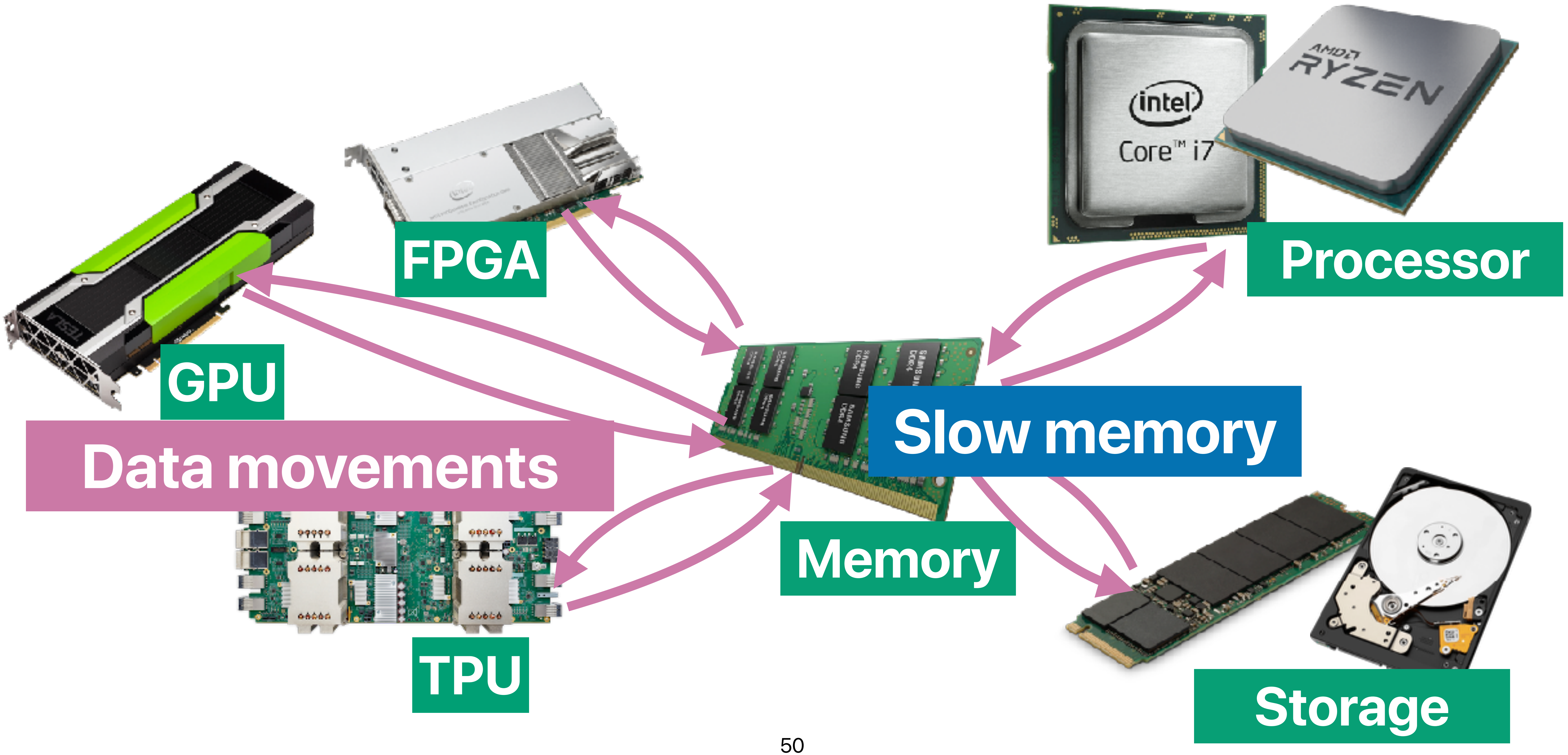
Larger throughput doesn't mean shorter latency!

Throughput and end-to-end latency

- Regarding the following statements about the relationship between FLOPS and end-to-end latency, how many of them is/are correct
  - ① Offloading the main computation kernel where the baseline spend the most time from CPU to another computing resource with higher FLOPS will improve the end-to-end latency
  - ② If we place the file on a disk with higher bandwidth, we will spend less time in file access.
  - ③ A computing device with higher parallelism (e.g., more cores) can potentially deliver higher FLOPS even though each parallel instance is slower
  - ④ For real-time applications, we should prefer computing resources that deliver more results per second

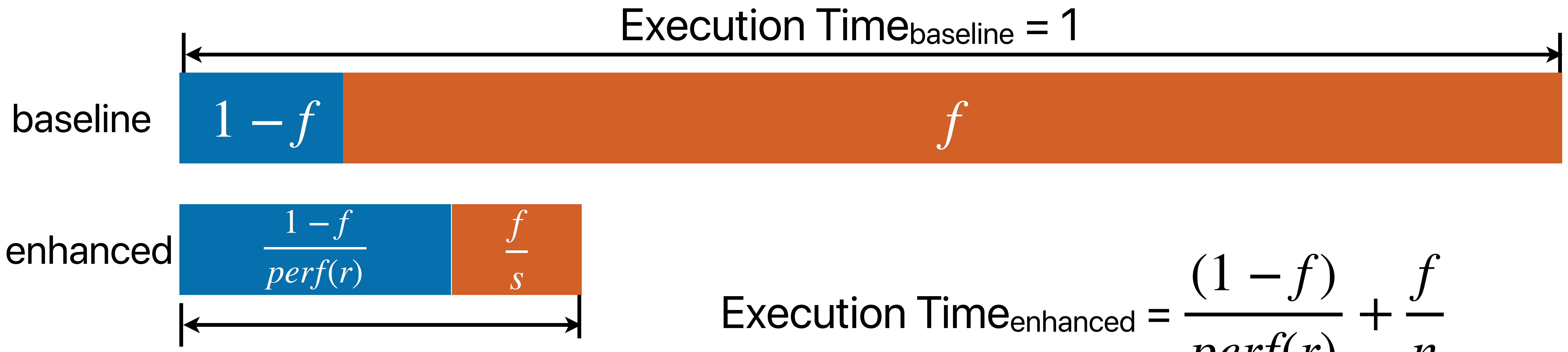
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Most cases, parallelism is not "tax-free"



What if parallelism is not "tax-free"?

- Parallelization Overhead
  - Preparing/exchanging/synchronizing data
  - Additional function calls/control overhead
- The "$1-f$" will potentially slowdown by a factor of $perf(r)$



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

Amdahl's Law considering overhead

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

- r is some other parameter that affects the overhead
  - input size?
  - degree of parallelism?
- *perf(r)* should be <=1 (i.e., slowdown)
- The overhead may scale differently than the original problem — that's why we introduce "*perf()*" function

Take-aways: being an honest engineer

- FLOPS can only be meaningful in very limited cases as FLOPS does not consider all aspects of the performance equation
- Without mentioning the specific conditions or changing more than one factors in the evaluations, the performance result is meaningless
- Most of time, there is no free lunch in optimizations — don’t ignore those overhead. Don’t let those overhead out-weight their benefits

GEMM is not the whole LLM

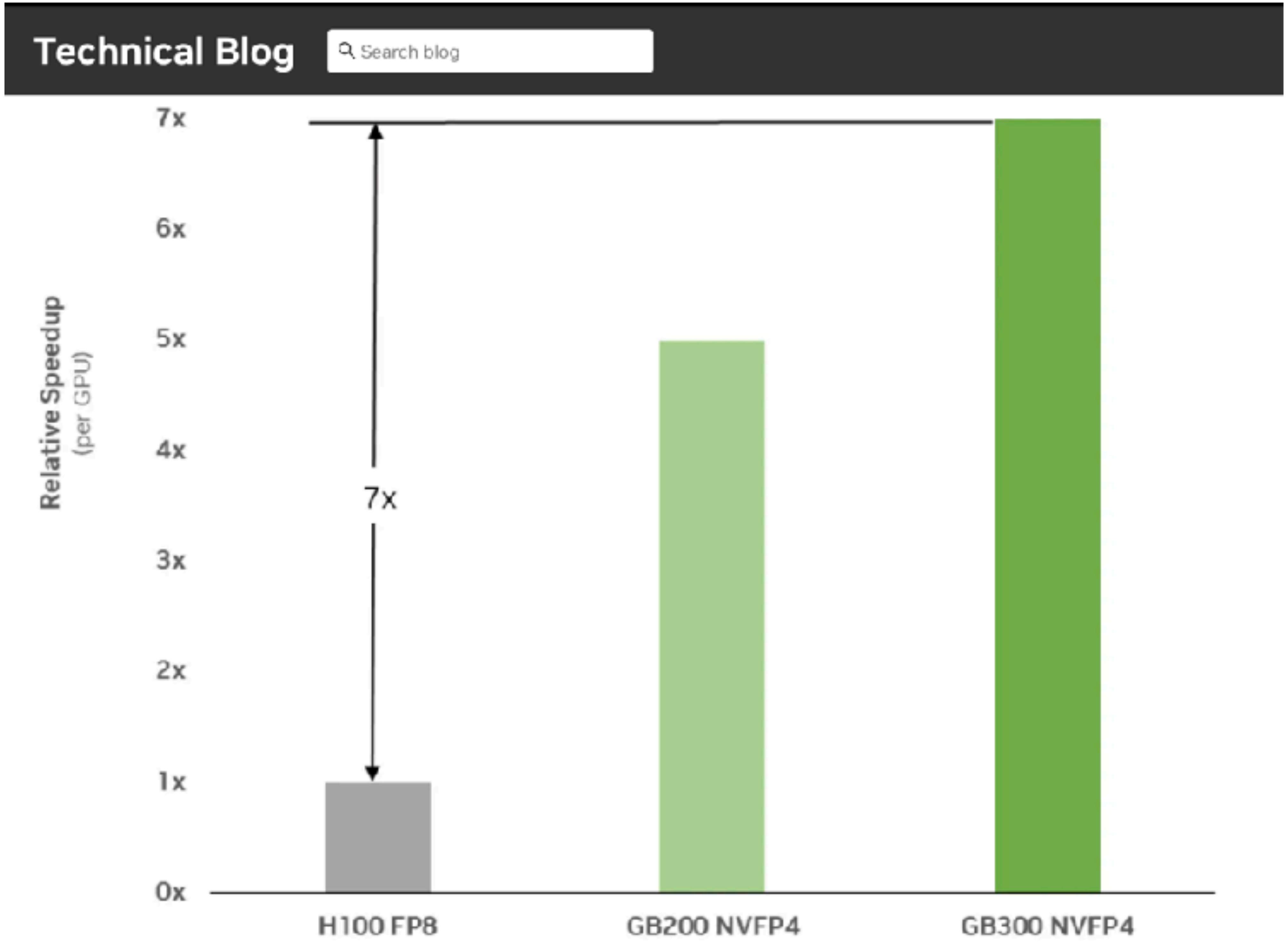


Figure 1. Measured GEMM performance shows GB300 delivering a 7x speedup over Hopper, accelerating core LLM training operations through faster FP4-optimized matrix multiplications.

https://developer.nvidia.com/blog/nvfp4-trains-with-precision-of-16-bit-and-speed-and-efficiency-of-4-bit/

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., a speedup of $\frac{1}{10}$)...

$$Speedup = \frac{1}{\frac{(1-f)}{perf(r)} + \frac{f}{s}} = \frac{1}{\frac{(1-0.9)}{\frac{1}{10}} + \frac{0.9}{9}} = 0.91\times$$

Takeaways: find the right thing to do!

- 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!

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

  - Corollary 3 — Optimization has a moving target
  - 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**

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

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

- Quote only 32-bit performance results, not 64-bit results. — FLOPS changes as datatype changes
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application — Not the whole program uses FLOPs intensively
  — An inner kernel is not everything — There may be a cost introduced by the optimization!
- 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
  — FLOPS changes as dataset size changes
- Compare your results against scalar, unoptimized code on Crays
  — IPS cannot reflect if the quality changes
- 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 — FLOPS changes as algorithm changes
- 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
  — IPS cannot reflect if the model/algorithm changes
- 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

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)

Take-aways: being an honest engineer

- FLOPS can only be meaningful in very limited cases as FLOPS does not consider all aspects of the performance equation
- Without mentioning the specific conditions or changing more than one factors in the evaluations, the performance result is meaningless
- Most of time, there is no free lunch in optimizations — don't ignore those overhead. Don't let those overhead out-weight their benefits
- End-to-end latency is still the most faithful way to assess the real application performance

Announcement

- Assignment 1 **due this Thursday** — submit to the right item on Gradescope
- Reading quiz due next Tuesday before the lecture
  - We will drop two of your least performing reading quizzes
  - Please read the required readings
  - You cannot switch IP address, reopen the browser during the quiz
- Check our website for slides/quiz links, Gradescope to submit assignments, discord for discussions
- Check your grades at https://www.escalab.org/my_grades
  - If you don't have any grade, you need to make sure your gradescope account is associated with your UCRNetID@ucr.edu
  - You have to submit a course agreement to receive scores
- Youtube channel for lecture recordings: https://www.youtube.com/c/ProfUsagi/playlists

Computer
Science &
Engineering

203

つづく