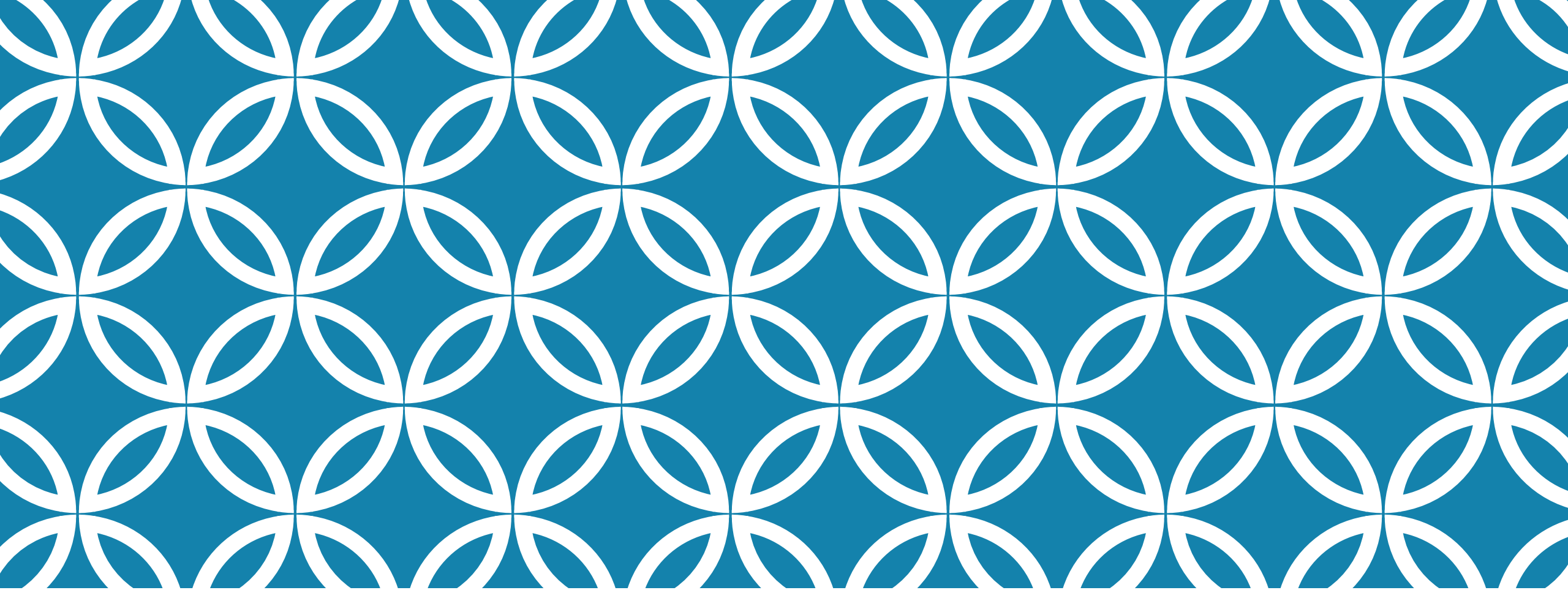


LEVERAGING GPUS

Professor Ken Birman

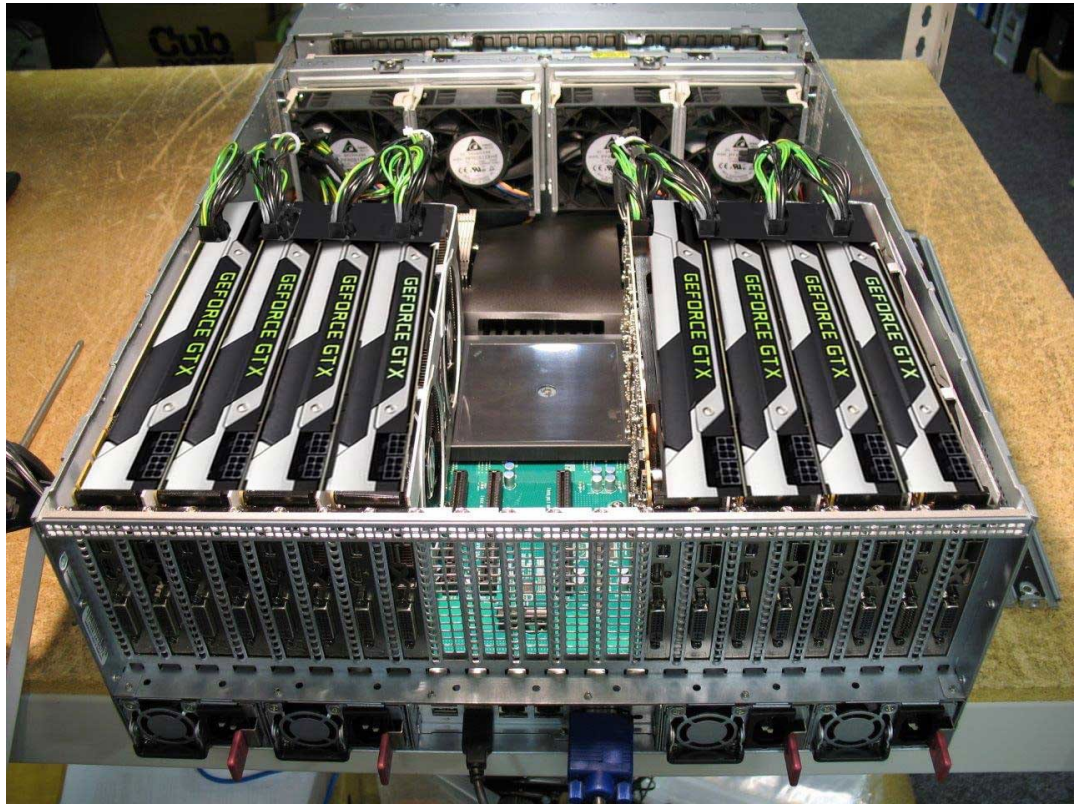
CS4414/5416 Lecture 25

OVERVIEW

Hardware architecture

Programming model

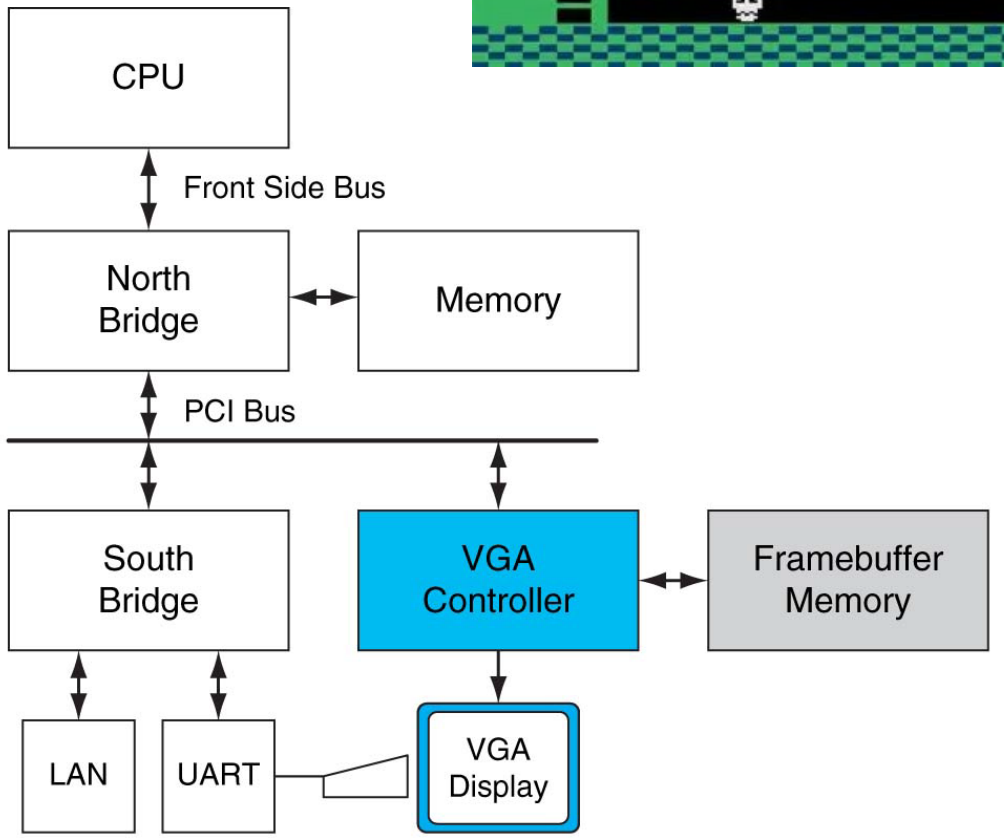
Example



HISTORICAL PC

Early PC architecture was focused on video graphics

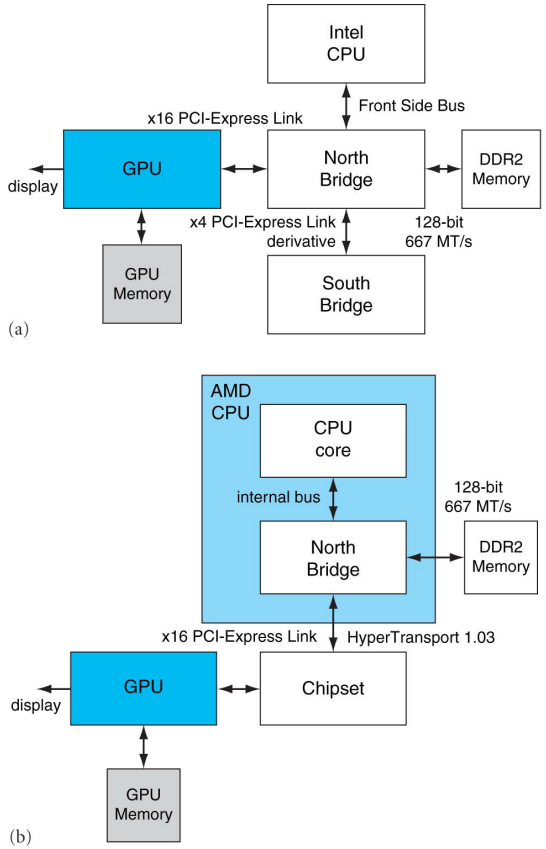
A video graphics accelerator card was used to assist the software with image rendering



INTEL/AMD CPU WITH GPU

GPUs were introduced to support more powerful styles of computer gaming as that market emerged

They were designed "like" CPUs but with a primary focus on SIMD compute: single instruction, but many data items at a time. The instructions were picked to accelerate video games



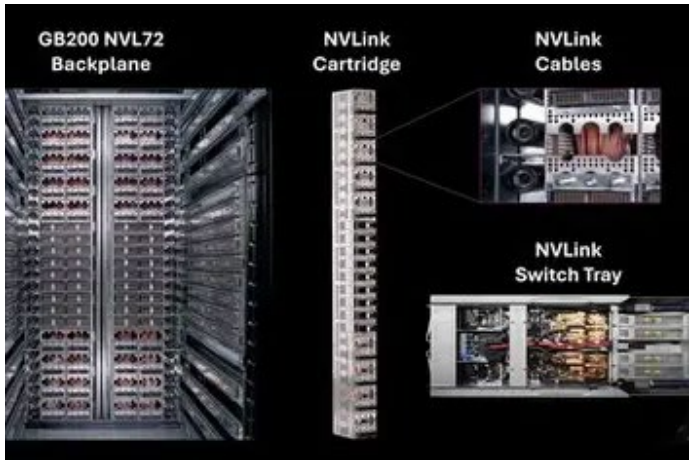
GPU EVOLUTION

Even with early GPUs, the main tasks were graphics and animation. But then NVIDIA made a hugely successful bet.

During 2000-2015, successive generations of NVIDIA GPUs were introduced, and the coverage spread to include parallel computing (HPC) in addition to gaming, and then AI.

GPUs for machine learning became critical as ML took off

WITH ML TRAINING, DEMAND FOR GPUS SKYROCKETED!



Today we have NUMA host computers with 100 cores that can also support 8 or more GPUs per host.

- Each host has a special memory bus dedicated to the GPUs
- Each "server" can be networked to others in the same rack.
- Effect is to offer a massive NUMA GPU compute framework

HOW SHOULD WE THINK OF GPUS?

They are a bit like computer-operated calculators!

Like with a calculator, you need to download your data, but can store it in the memory of the device.

Then there are fancy “buttons” you can press to request computations of various kinds

CUDA PROGRAMMING LANGUAGE

GPUs are programmed with CUDA. Based on C, but with some features that feel more like assembler language and even microcoding.

The idea is to design instructions that operate in a maximally parallel way on vectors of data.

CUDA automates some aspects but algorithms must be designed by hand for “flexible” parallelism (because GPUs have varying power)

THE BASIC IDEA MIMICS THE SIMD COMPUTING WE DISCUSSED IN LECTURE 7

A GPU operates on a very wide vector, much more than a host computer can compute on in a single cycle.

- Data must be a multiple of the GPU cache-line size. Most GPUs have a 64-byte cache-line, although some double this to 128.
- But there are *multiple connections to memory*. So whereas a host computer is limited to one cache-line per CPU at a time, a GPU can have a single instruction that runs on many cache lines in one action.
- The same instruction will be performed on every element in a single clock-cycle. The instruction itself is just like a normal CPU instruction
- We end up with a vector of results, like with MMX, but "wider"

HOW DO GPUS DIFFER FROM MMX?

With host parallelism from the SIMD instructions (MMX), the size of the vector is 64 bytes. So one instruction is limited to 64 bytes

But a GPU instruction can operate on as many as 72 cache lines (12 per memory "stack," but a high end GPU like NVIDIA's H100 has 6 memory stacks), and each cache line is 512 bits wide

For example, this could be 1502 32-bit floats

BUT GPUS ARE VERY HOT DEVICES

Recall our "toaster" analogy:

- Active circuitry of any kind on a chip is a bit like your toaster when you are toasting a bagel.
- Heat is radiated – and GPUs have a lot of active circuitry!

We talked about how the clock rate determines the heat: in effect, the energy radiated runs as $O(R^2)$ in the rate R. The intuition is that a chip is a 2D surface covered with wires.

SO A GPU…

Uses a very slow clock cycle, but can do a lot of work in a single instruction

A typical GPU clock rate might be 1GHz, whereas a typical cloud-quality CPU has a clock rate more like 3GHz or 4GHz

Moreover, host CPUs do fancy data and instruction fetch reordering, which GPUs don't support.

HOST VERSUS GPU

We always use host code to perform file access

It would be unreasonably slow for a GPU to deal with the file system data structure, system calls, asynchronous I/O with block by block interrupts, etc.

Mostly, the host does network I/O too.

HOST VERSUS GPU

Each host core will be at least 3x or 4x faster *per core* for "straight line" logic that doesn't use vector instructions. With host prefetch, caching and branch/value prediction, each core may be 10x faster. And we may have 100 cores, all doing different things.

So the host could be 1000x faster for tasks we would code in C++.

A GPU has very few control threads but will be dramatically faster for very wide vector instructions where it can "show its stuff" to the maximum.

HOST VERSUS GPU

We also need to transfer data into and out of a GPU, bringing data transfer overheads.

Those costs won't matter for big objects and long mathematically intense tasks. The overheads can seem huge for tiny tasks. So we only use GPUs for big computations, not small ones.

CAN A GPU DO "IF" STATEMENTS?

Definitely! But not element by element during vector computing

Sometimes there are clever ways to transform an element by element if statement into some form of parallel vector math operation… cuda uses these ideas

But in general element by element logic just isn't parallelizable

INSTEAD, ML USE OF GPUS CENTERS ON GEMM

GEMM stands for Generalized Matrix Multiplication

$$C = \alpha AB + \beta B$$

A and B are input matrices, while α and β are scalar weights

Much of the heavy lifting in modern ML can be expressed in terms of GEMM computations. Thus while GPUs can do many things, the "arms race" has centered on GEMM.

IN FACT MOST ML IS GEMM MATHEMATICS CONTROLLED BY HOST LOGIC

We code in PyTorch, Tensor Flow, C++, etc

But the ML algorithm is very high level

Any heavy lifting is done by calling a wrapper that turns around and transfers data to the GPU, then "presses the run button".

AGGREGATION CAN HELP A LOT

Many ML operations involve an input vector and GEMM computations.

We can glue lots of input vectors for different input requests together to create an input matrix, compute on the batch, then split out the separate results for each input request.

This will scale far better than doing *n* vector-matrix operations.

GPUS HAVE HUGE MEMORY BANDWIDTHS

GPUs have incredible memory bandwidth, much more than the host computer. Yet they are bottlenecked by memory access!

Part of the issue is that the form of many-way memory used for this is inherently very expensive

As a result, GPUs are surprisingly NUMA in design: memory speeds vary greatly for different GPU tasks.

GPU NUMA EFFECTS

A GPU can typically directly access a small amount of “on chip” memory. This is used for a GPU memory cache

It can then access local memory on the same “circuit board”, but the amount is limited, typically a few GB

And it can access extension memory off the board but in the same GPU unit. **Each “step” is slower.**

EXAMPLE: NVIDIA H100

This unit is cutting edge… it has a maximum of 80GB memory

But NVIDA offers a special device called an NVLINK that allows us to put 8 H100 units on a single host. All the GPUs can access memory from one-another (but at a higher delay!)

So one host could have 8 GPUs and as much as 640GB GPU memory

H100 CAN ALSO ACCESS HOST MEMORY

Host memory is very slow from the GPU's perspective: data literally has to be copied into the GPU cache in order for GPU instructions to work on it

Yet host memory it is incredibly cheap compared to GPU memory

It is common for an H100 to directly access host memory – and on a cloud server, 900GB memory is not unusual.

RDMA CARRIES THIS EVEN FURTHER

In upcoming lectures we will discuss RDMA networking

One option permits a GPU on host A to read or write memory from the GPU on host B.

The delays are huge compared to local memory, but the transfer rates can be insanely high, so the feature is widely used

NVIDIA NCCL LIBRARY

This becomes so complex that the GPU vendor typically has to create all the needed software.

For example NVIDIA (today's GPU leader) offers a collective communications library called NVIDIA CCL: NCCL.

The AllReduce primitive in NCCL is probably the most important distributed computing tool in modern ML.

GPUS CAN HAVE MULTIPLE THREADS, TOO

Somewhat confusingly, they use the term thread for something else, but they do have a form of lightweight thread just like C++

However, unlike a 100-core NUMA CPU that can run 100 side by side parallel threads flat out and doing different things, a GPU allows just a very small number of true threads, like 4 or 8

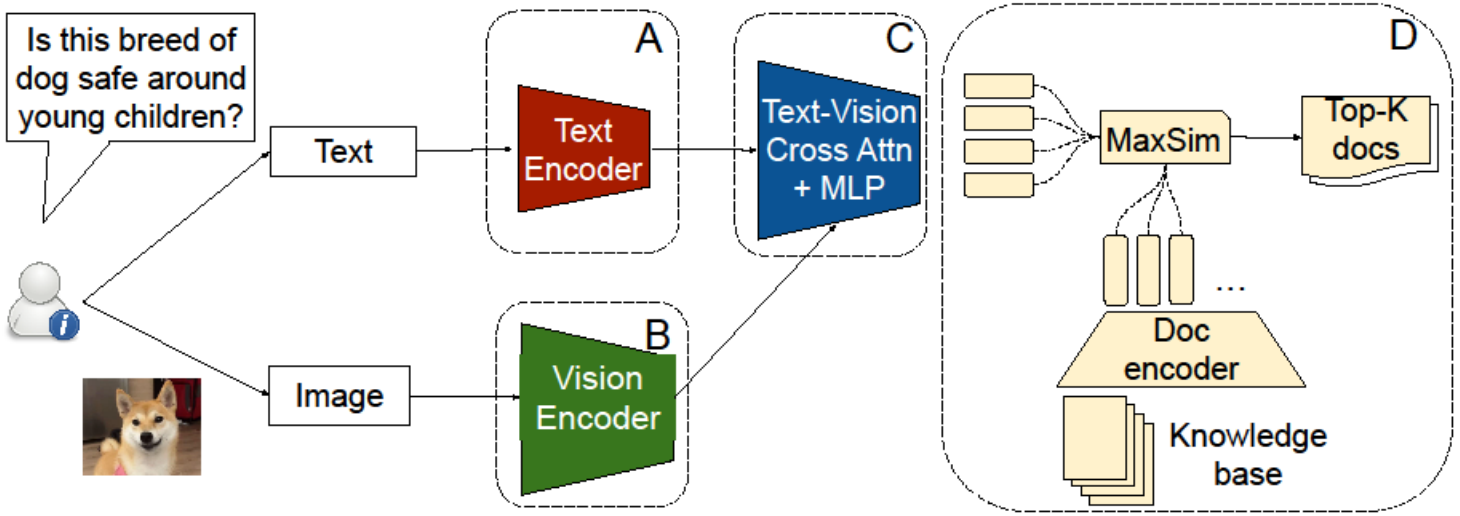
THESE ARE CALLED MINIGPUS (MIGS) OR SLICES

With so much compute power, sometimes we want to run more than one task concurrently on a single shared GPU

For this, NVIDIA offers a way to subdivide a GPU into 2 partitions each with half the CPUs, or we can subdivide further into fourths or eighths. MIGs.

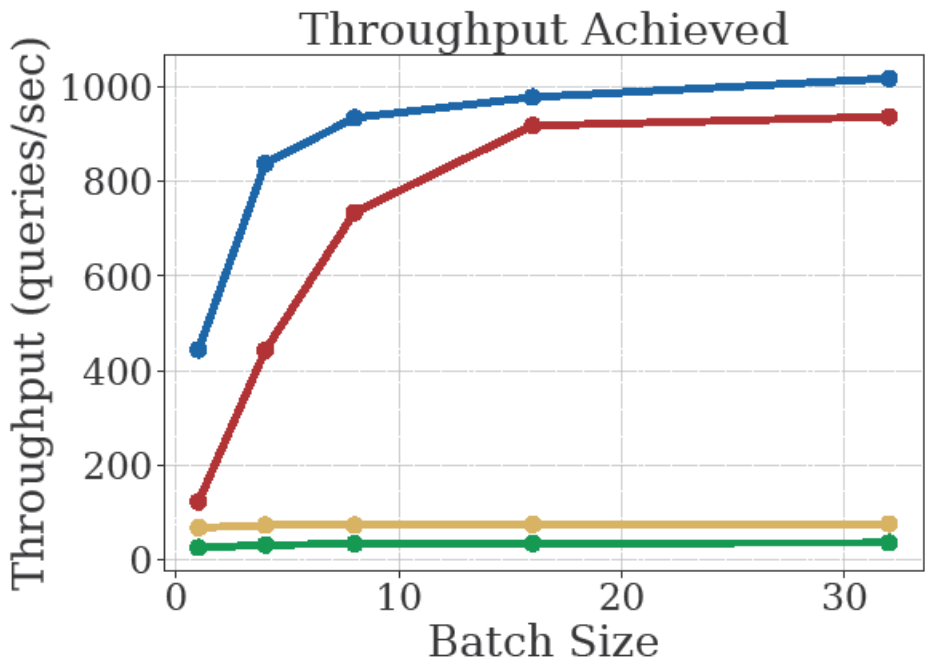
Each has a single control thread (in the CPU sense of the term)

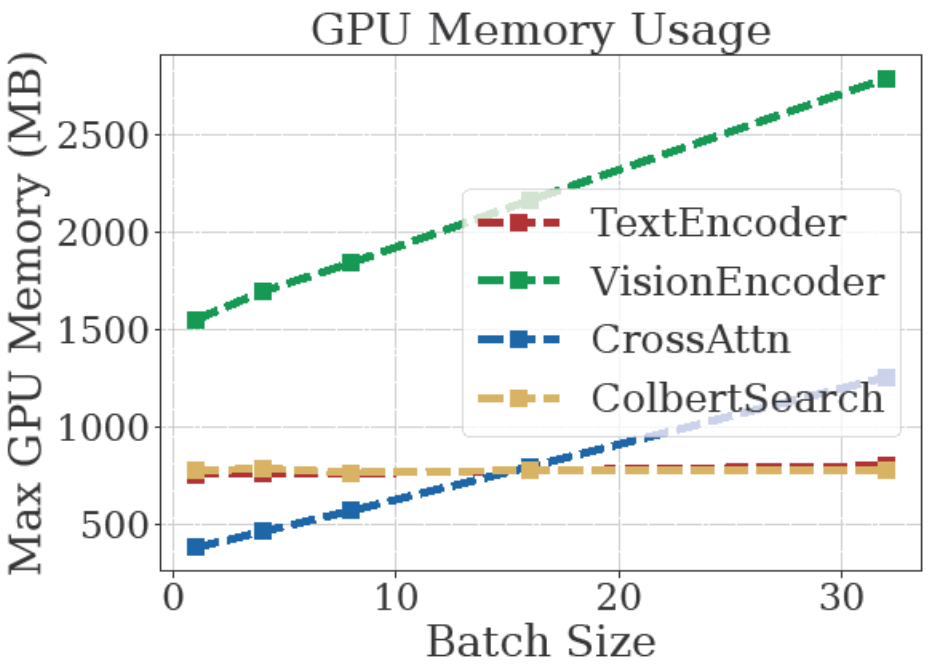
THIS CREATES AN INTERESTING SCHEDULING CHALLENGE! ALICIA HAS STUDIED IT



PreFLMR Pipeline

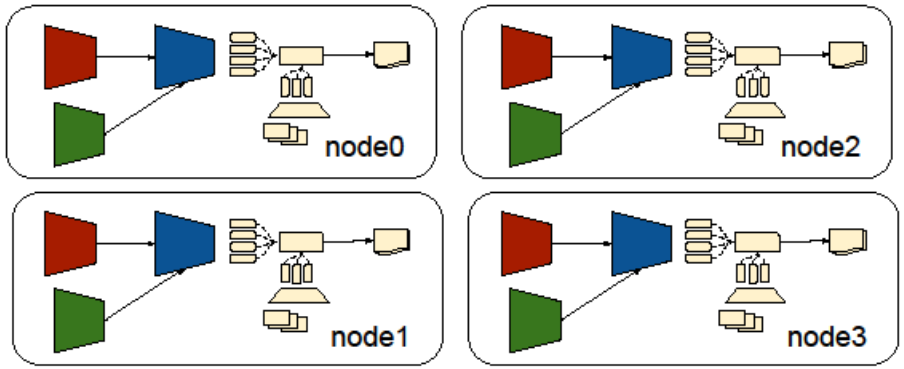
DIFFERENT COMPONENTS OF PREFLMR NEED DIFFERENT AMOUNTS OF GPU MEMORY

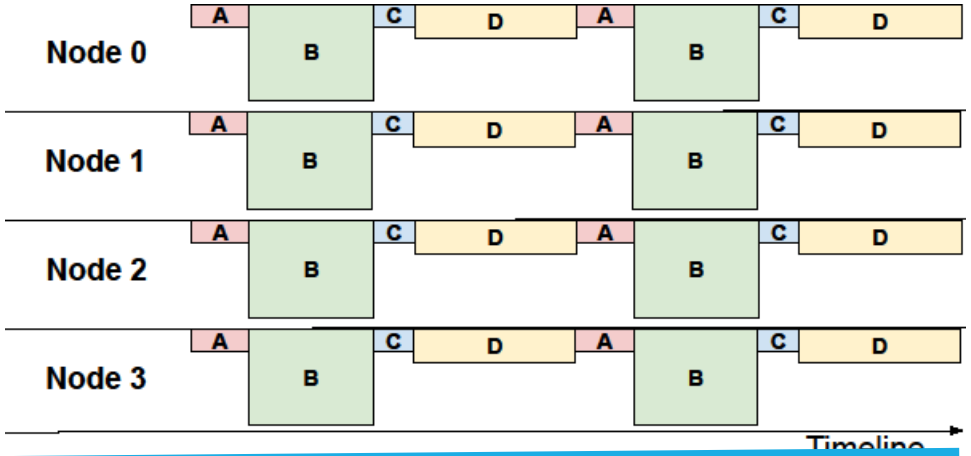




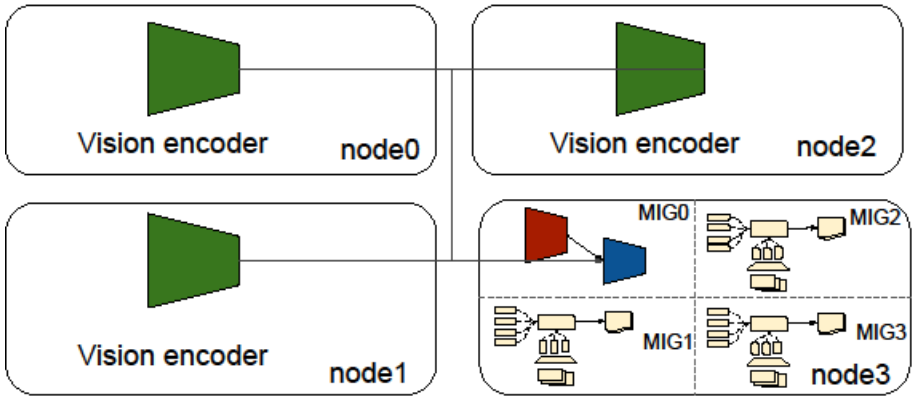
RUN EVERYTHING ON ONE SERVER OR SPLIT ACROSS MULTIPLE SERVERS?

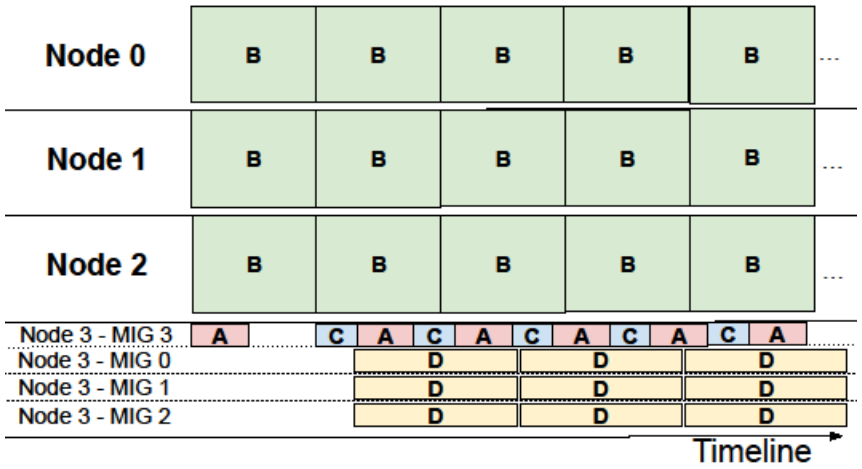
Monolithic: every node runs all four components





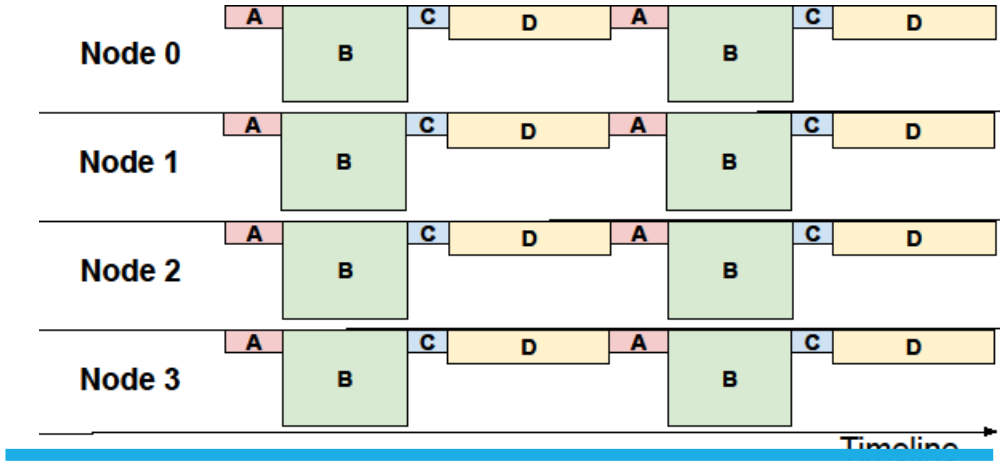
Microservices: Nodes 0,1,2 only run B Node 3 runs A, C and D



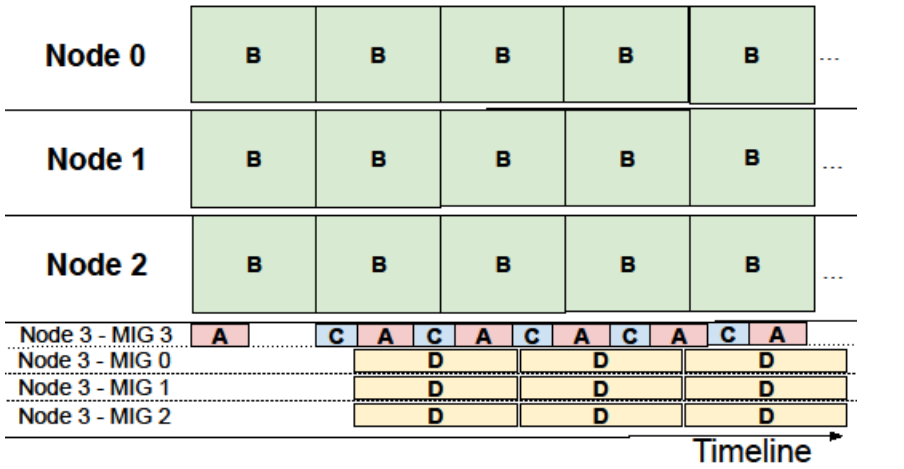


RUN EVERYTHING ON ONE SERVER OR SPLIT ACROSS MULTIPLE SERVERS?

The Microservice deployment is already at work on queries 16-18 when the Monolithic one is still finishing up queries 5-8



This illustrates that with MIGs and smart scheduling, we can often squeeze much more work from a single platform!



THE STORY ISN'T JUST CENTERED ON GPUS

Programming environments for ML matter a lot too, and really automate placing tasks onto GPU for the user

We saw that RDMA network acceleration is valuable (we have two lectures coming soon on that topic)

The memory used to hold ML models has become a focus

PYTORCH AND TENSOR FLOW

These specialized packages for Python emerged as the preferred options for coding ML algorithms.

Tensor Flow has the larger market share if we consider all forms of GPU compute, but the two are pretty much tied for ML applications.

A diverse set of other languages and packages are also used

BANDWIDTH LIMITED?

Early push towards GPUs focused primarily on arithmetic intensity of ML: the need to do vast numbers of floating point operations (flops).

But as ML models soared in size the balance shifted towards memory: with models that can have hundreds of billions of parameters, a single GPU could no longer hold the whole model

THIS LED TO NEW ALGORITHMS

Modern implementations of GEMM use the AllReduce pattern to perform work on multiple GPUs in parallel.

The library breaks the huge tensors into blocks and performs block by block parallel compute, but then needs to exchange and combine intermediary results

But there is debate about the bottleneck…

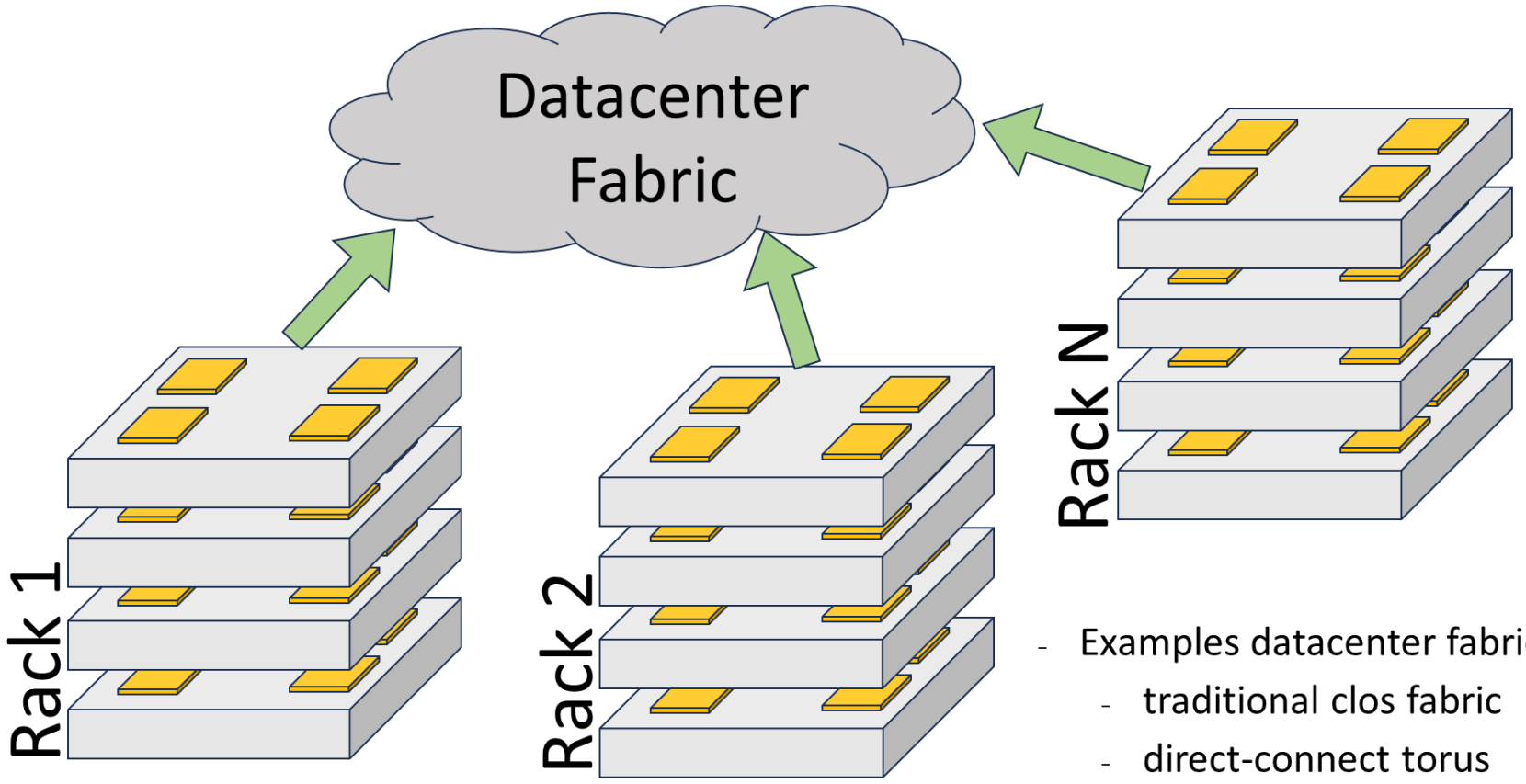
DEVELOPERS MOSTLY DON'T SEE THIS IN ANY DIRECT WAY

ML developers work with higher level ML packages that can be used like subroutines, from libraries

The package developers express ML compute in terms of APIs focused on the primary operations seen in DNNs, and these are then mapped to CUDA implementations of GEMM

Just the same, if GEMM is slow the resulting systems can be inefficient

RACHEE SINGH WORKS ON NETWORKING



- Examples datacenter fabrics:
  - traditional clos fabric
  - direct-connect torus

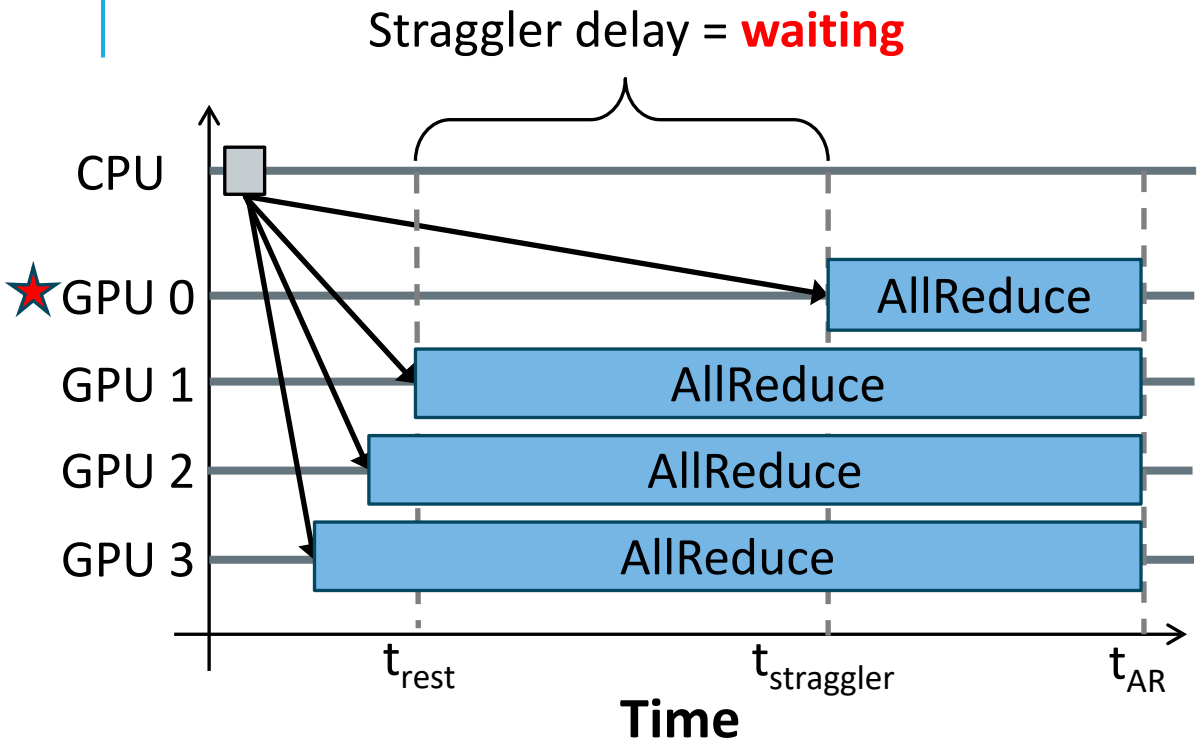
REMINDER: THE HOST IS THE DIRECTOR.

The GPUs are workers but the host is in charge.

Rachee looked at scaled out cases that require work on multiple GPUs, perhaps even spread over multiple hosts.

The AllReduce CCL pattern is dominant for such computations.

STRAGGLER GPUS IN BULK-SYNCHRONOUS ML PIPELINES



In a 4 GPU server, straggler is late by 10 milliseconds

- AllReduce ends with each worker collecting data from all the others and then reducing (aggregating)
- We think of this as a synchronous computation limited in part by network speed. Rachee found that late results (here, from GPU 0) often are a far bigger problem!

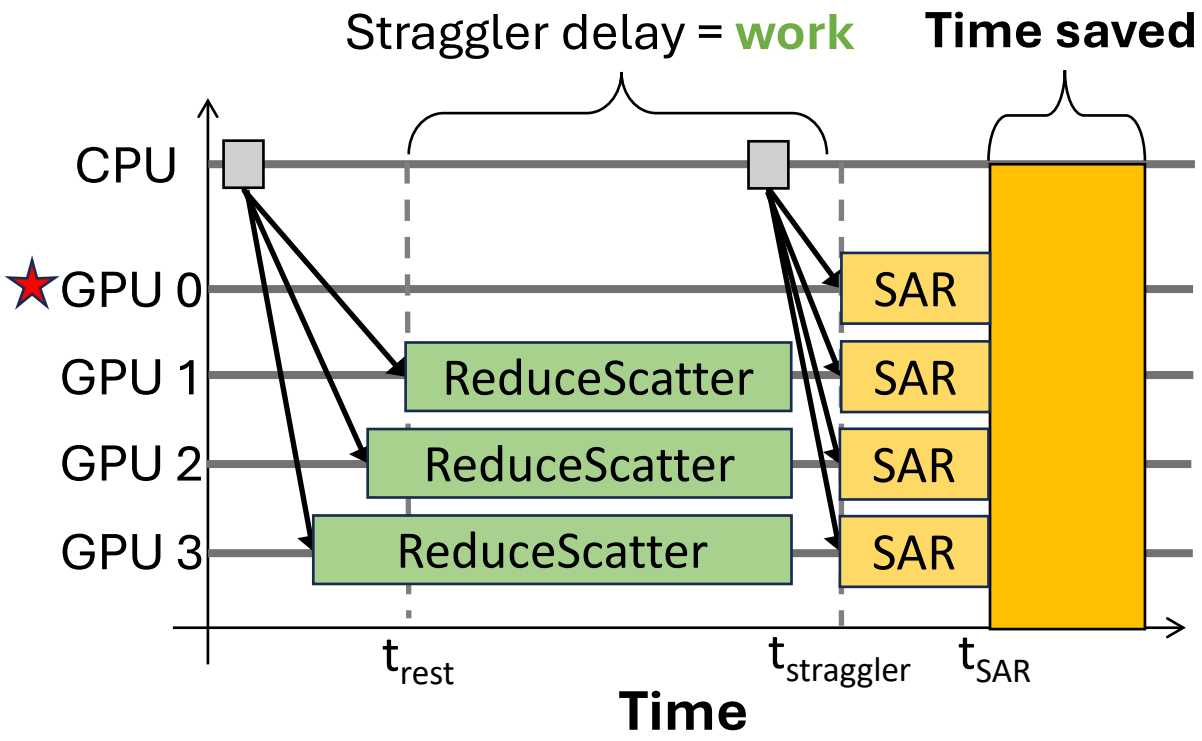
How bad is this for ML workloads?

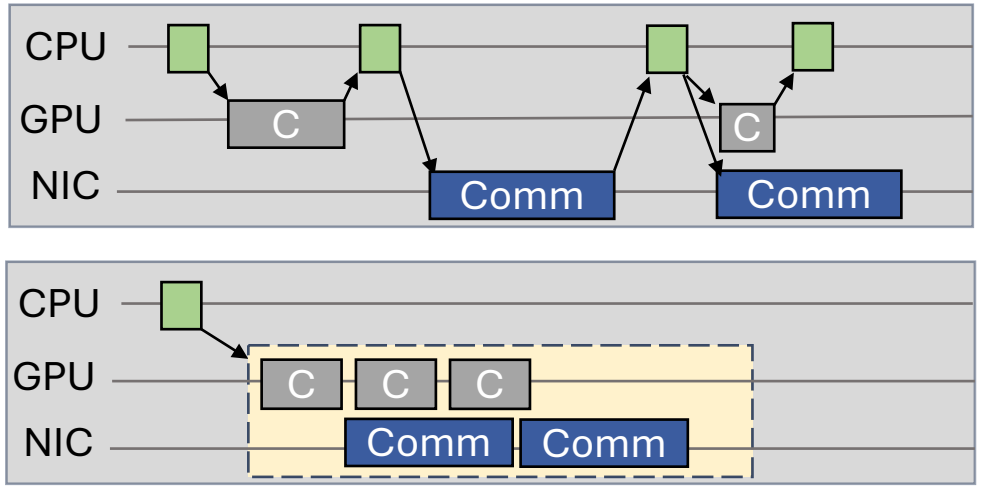
- Let's use realistic configs:
  - Batch size = sequences per GPU
  - Sequence length = 2048
  - $d_{model} = 8192$
  - Tensor parallel size $p = 8$
  - FP16 (2 bytes per element)
- **Buffer size per GPU**
  - $B \times S \times (d_{model}/p) \times 2\ bytes = 8 \times 2048 \times (8192/8) \times 2 = 268{,}435{,}456\ bytes \approx 256\ MB$
- Each GPU's **AllGather or ReduceScatter buffer** is about **256 MB**
- A Blackwell GPU in NVL72 has **18 links → ≈900 GB/s one-way** aggregate per GPU.
  - **All 18 links aggregated (≈900 GB/s one-way):**

$$t = \frac{256 \times 10^6}{900 \times 10^9} \approx 2.84 \times 10^{-4}\ s = \mathbf{284\ \mu s}$$

Less than *a millisecond to move the bits* and potential delays of *10s of milliseconds in synchronization*?

Synchronization, Not Bandwidth, Bottlenecks ML Scale-Up Domains





Design “straggler-aware” algorithms

https://arxiv.org/abs/2505.23523

Build asynchronous ML pipelines

https://www.arxiv.org/abs/2506.04667

IMPLICATIONS OF RACHEE'S STUDY?

For 25 years GPU compute has focused on a form of synchronous parallel computing on a single GPU

The move to distributed compute using AllReduce is much more asynchronous and GPUs behave less predictably

A consequence is that we will see more and more focus on asynchronous parallelism in algorithms, but also in new memory architectures for individual GPUs

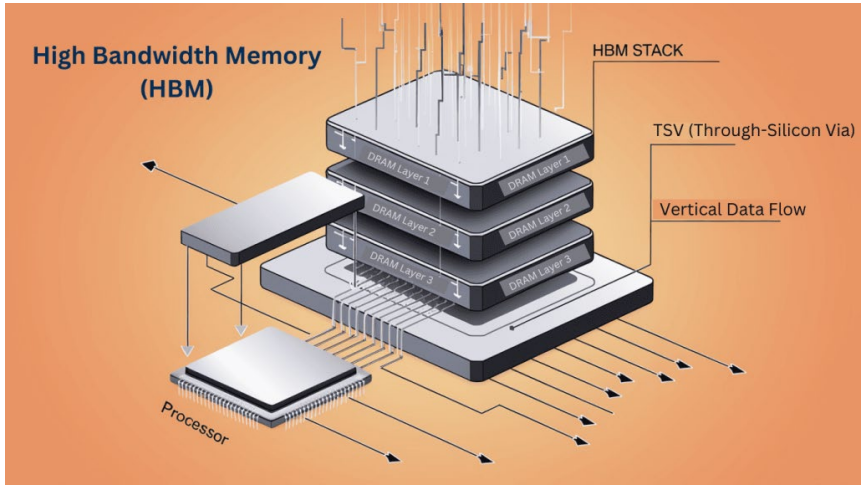
BUT EVEN SO…

Rachee's work points to opportunities at the CUDA and GEMM kernel levels, and the network

But the industry is hoping for an end run: breaking the 80GB limit on GPU memory size

The issue is physical memory location and density

3D MEMORY FOR GPUS



Called High Bandwidth DRAM, or

Involves stacking memory modules (and often, requires using liquid cooling because the modules would melt otherwise)

Very likely to be the next big boom in hardware for ML

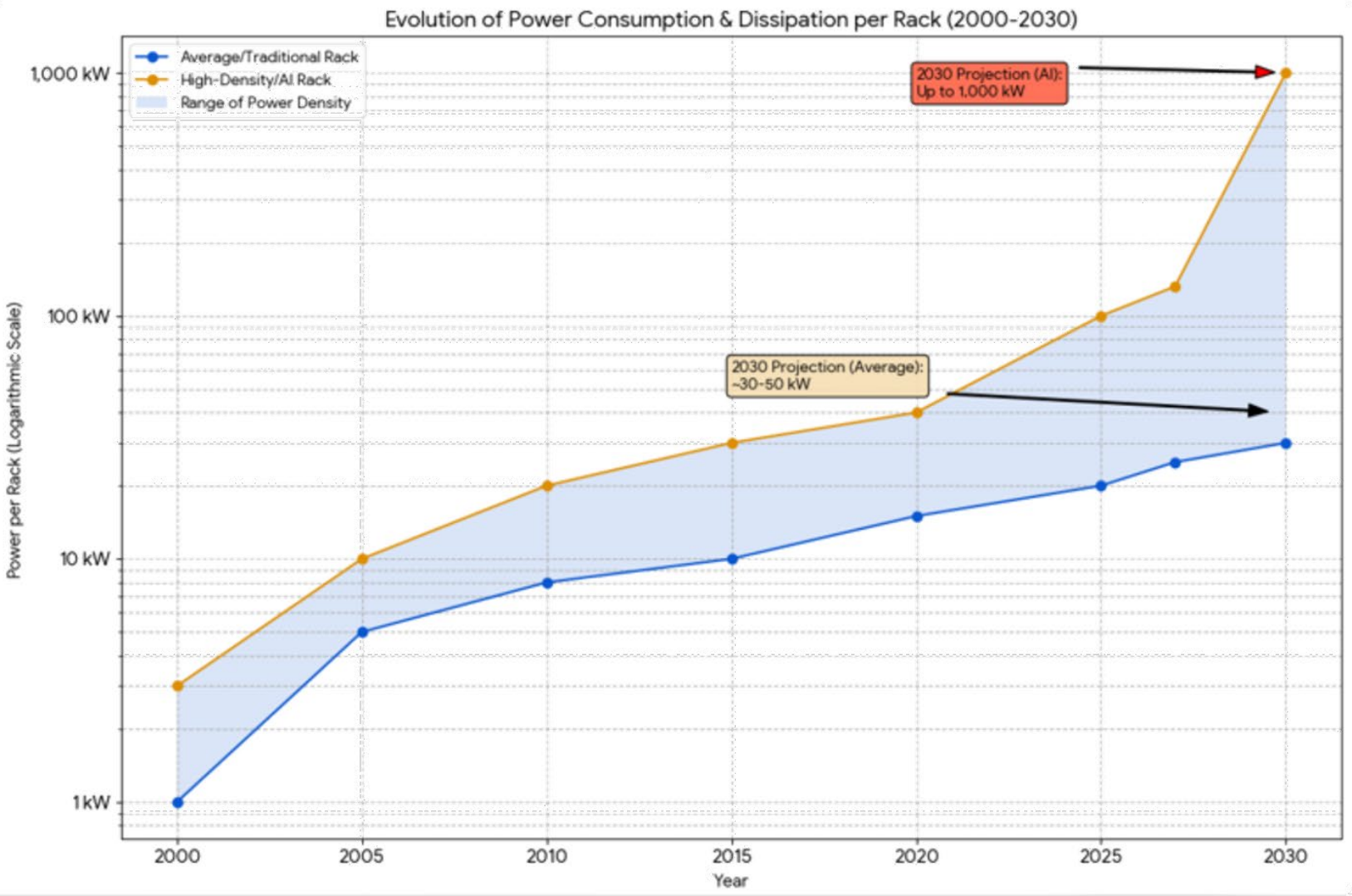
THIS IS MAKING ML EVER MORE POWER INTENSIVE AND EXPENSIVE!

In fact the technology is splitting into ML applications on the one hand, which are a bit less demanding, and the race for AGI, which is advancing ever faster.

Metrics like “tokens/month” are becoming a common way to compare the market share of different AI players

AGI is of debatable value (some people think it could end humanity, at last as we’ve known it). Today it still seems to be out of reach.

TECHNOLOGY LEADERS ARE WORRIED!

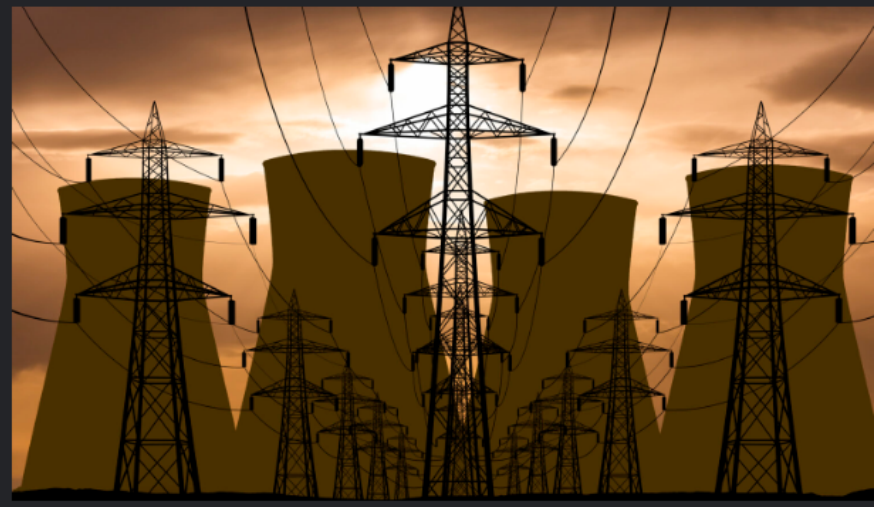


POWER COUPLING

OpenAI and Nvidia's $100B AI plan will require power equal to 10 nuclear reactors

"This is a giant project," Nvidia CEO said of new 10-gigawatt AI infrastructure deal.

BENJ EDWARDS – SEP 22, 2025 3:17 PM | 142



Credit: Anton Petrus via Getty Images

On Monday, OpenAI and Nvidia jointly announced a letter of intent for a strategic partnership to deploy at least 10 gigawatts of Nvidia systems for OpenAI's AI infrastructure, with Nvidia planning to invest up to $100 billion as the systems roll out. The companies said the first gigawatt of Nvidia systems will come online in the second half of 2026 using Nvidia's Vera Rubin platform.

"Everything starts with compute," said Sam Altman, CEO of OpenAI, in the announcement. "Compute infrastructure will be the basis for the economy of the future, and we will utilize what we're building with NVIDIA to both create new AI breakthroughs and empower people and businesses with them at scale."

COULD SOME ML TASKS MOVE TO HOST COMPUTE?

Now that hosts have 100 cores and large DRAM memories, there is growing interest in taking a new look at ML to see if some tasks can be shifted to host compute

For example, in ZenFlow (a Berkeley project), LLM inference is split so that some occurs asynchronously on the host while only the arithmetically intense work runs on GPU, yielding big wins

WHAT ABOUT FPGA OR ASICS?

FPGA and ASICs are ways to compile a specific function into a dedicated chip. Only the logic used for that function is present

FPGAs are reprogrammable, so they fit well in cloud computing settings. An ASIC is for a specific circuit but are often faster and cheaper.

These can yield a simple pass to higher performance at less cost and less energy expense, but not for identical tasks – on GEMM tasks, GPUs will remain the leaders for at least this "tech cycle"

COULD PURE OPTICAL COMPUTING BE AN "END-RUN" WINNER?

There has been a lot of talk about moving to purely optical GEMM computing, at least in the future

- Photonic circuits can be created in silicon, just like electronics
- Advantage isn't speed: speed of light and speed of electrons in a fast conductor like copper are actually pretty similar.
- But **photonic computers dissipate less heat**, possibly enabling much higher clock rates – if photonic memories can be created
- Storage based on "magneto-photonics" are showing promise

COULD QUANTUM COMPUTERS HELP?

Quantum computers are very fast for a particular task, which centers on intersections of quantum probability distributions

Some people believe this will eventually turn out to be the answer for scaling ML to the next level, but others are skeptics

Beware science fiction portrayed as news stories or breakthroughs…

COULD QUANTUM C(

Quantum computers are very
centers on intersections of qu

Some people believe this wi
answer for scaling ML to the

Beware science fiction portra
breakthroughs…

TECHNOLOGY

This Startup Wants to Harness Quantum Computing to Make AI More Affordable

Quantum computing is capable of mind-bending tricks. With Sygaldry, Chad Rigetti hopes it can lower the costs for AI development.

BY CLAIRE CAMERON, FREELANCE WRITER

AUG 12, 2025

SHARE

Chad Rigetti. Photo: Courtesy company

COULD Q

Quantum cor
centers on int

Some people
answer for sc

Beware scier
breakthrough

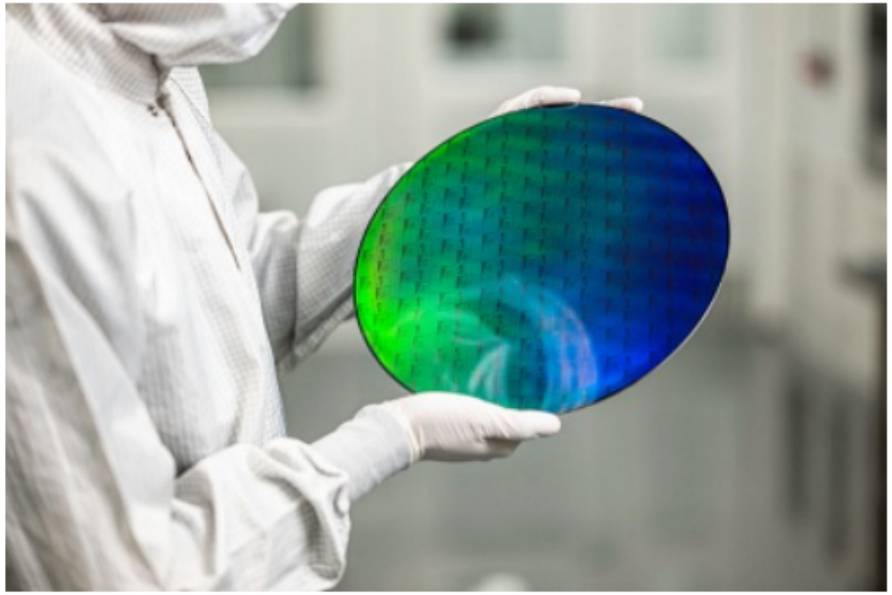
Research & Development

PsiQuantum claims silicon photonics breakthrough for quantum computing

27 Feb 2025

Nature paper details manufacturable platform with silicon nitride waveguides and barium titanate switches.

PsiQuantum, the Silicon Valley photonic quantum computing startup, is claiming a significant breakthrough with a new high-volume process it says will be capable of manufacturing million-qubit-scale systems.



Wafer-scale silicon photonics for quantum

The Palo Alto firm, which raised $450 million in a BlackRock-led investment in 2021, has revealed details of "**Omega**", a quantum photonic chipset purpose-built for utility-scale quantum computing, in a **scientific paper just published in the journal *Nature***

It details a chipset designed and fabricated on full-size silicon wafers at the GlobalFoundries silicon photonics fab in New York, with key components including high-performance, single-photon sources, superconducting single-photon detectors, and a "next-generation" optical switch based on barium titanate.

"Every photonic component is demonstrated with beyond-state-of-the-art performance," claims the firm. "The paper shows high-fidelity qubit operations, and a simple, long-range chip-to-chip qubit interconnect – a key enabler to scale that has remained challenging for other technologies.

"The chips are made in a high-volume semiconductor fab, representing a new level of technical maturity and scale in a field that is often thought of as being confined to research labs."

Citing high-performance operating figures including a chip-to-chip quantum interconnect fidelity of 99.72 per cent, PsiQuantum added that it will break ground this year on two data center-scale quantum computing centers, one in Brisbane, Australia, and another in Chicago.

Pushing boundaries

PsiQuantum co-founder and CEO Jeremy O'Brien said of the latest advance: "For more than 25 years it has been my conviction that in order for us to realize a useful quantum computer in my lifetime, we must find a way to fully leverage the unmatched capabilities of the semiconductor industry. This paper vindicates that belief."

GlobalFoundries CEO Thomas Caulfield added: "Semiconductor manufacturing will inevitably be a large part of any solution to building

SUMMARY

Modern ML reduces to GEMM computing. GPUs (and TPUs) are optimized for this task, and RDMA helps a lot too.

But the massive size of ML models has become a limiting factor

It is causing GPUs to evolve into NUMA computers, DRAM to evolve into 3D stacked DRAM, networks into RDMA… and is rapidly transforming the computing landscape.

SELF-TEST

Assume you've been hired by an insurance company that is exploring the value of AIs in their insurance web sites.

Are there any computing tasks that GPUs can solve but that cannot be solved (even with much longer running) on a normal CPU?

In light of your answer, how would you present the business case for and against acquiring access to (or purchasing) some GPUs?

SELF-TEST

We learned about SIMD instructions for host compute in lecture 7 and now we've learned that GPUs mostly are the same, but

- They have much larger "local memory" than a NUMA core
- They have a lot more cores ("threads") than a normal host

If you could buy a normal host with a lot of cores (like 100, or 500) and with a larger cache-line size like 256 bytes, would that begin to compete with an expensive attached GPU unit?

SELF-TEST

Read about Intel's new AMX features (advanced matrix extensions). Beyond the Wikipedia link be sure to read articles in the press about this feature.

In your job for the insurance company, would you recommend trying these features before acquiring an expensive new compute cluster with NVIDIA's cutting edge GPUs? Why or why not? Try not to be "shallow" in your analysis!