Welcome to Neural Networks

Philipp Krähenbühl, UT Austin

What are Neural Networks?

- A old name for “Deep Learning”

What is Deep Learning?

- Machine Learning that works

Phone and laptop policy

- **Phones stay in your pocket**
  - Leave the room to use them
  - They are a distraction for everyone
- **Laptops in the last row only**
  - They are a distraction for people behind you
  - Flat tablets for notes are fine

Neural Networks class

- Webpage: https://ut.philkr.net/cs342/
- Friday: 1-4pm / GDC 2.216
  - With two 10min breaks
- Instructor: Philipp Krähenbühl (OH Friday 16:00 - 16:30 GDC 4.816)
- TAs: Haran Raajesh, Li-Yuan Tsao

Prerequisites

- Python
- Laptop with VSCode
- (Linear Algebra)
- (Basic ML background)

Your grade

- 6 x Homework (1/6 of grade each)
- Due midnight anywhere on earth (07:00 central time next day)
  - 1 day late: -10%
  - 2 days late: -30%
  - 3 days late: -100%
  - Solution will be released day 3. No exceptions. Plan accordingly.

Slip days

- Everybody gets 3 slip days. You may use them to waive
  - 1 day late penalty (cost: 1 slip day)
  - 2 day late penalty (cost: 2 slip day)
  - convert 2 into 1 day late (cost: 1 slip day)
  - we cannot waive a 3 day late penalty.
- Slip days are applied automatically and greedily. For example: if you're one day late on every homework. Late penalties on the first 3 homeworks are waived irrespective of your score.

Homeworks

- Coding
- Auto-graded (through canvas)
  - 5 submissions (**most recent submission counts**)

GenAI tools

- Use them (you'll use them in your job later too)
- Two rules:
  - Mark code written by GenAI: **# This code was written by XXXX** where XXXX is the name of the model you used.
    - Failure to do so might lead to plagiarism issues.
    - This has to be done **per-function** (ask the GenAI tool to do it)
  - You need to be able to **explain every piece of code you submit**.

Use GenAI tools responsibly

- Use GenAI tools to help you **grow and learn something**
  - Ask Questions
  - Ask it to explain code it writes
  - Use GenAI as a source of motivation
  - Use homeworks to familiarize yourself with GenAI tools

Use GenAI tools responsibly

- Things **NOT** to do
  - Hey Claude, do my homework

Accommodations

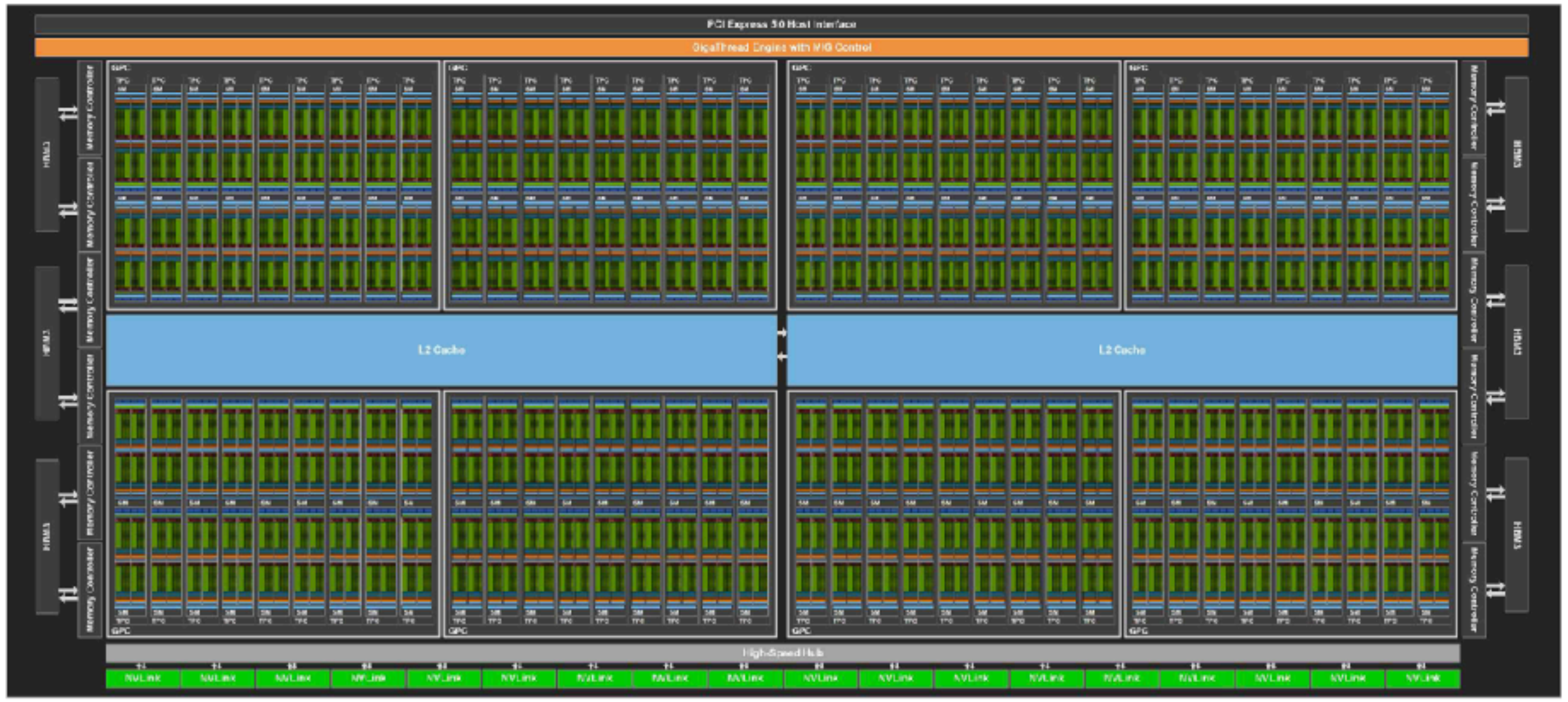
- Urgent matters: Dr’s note, waiving late days
- D&A
  - Classroom: Let me know during the break
  - No in-class, or scheduled exams
  - Slides online before class, I’ll try to record lectures, no attendance requirements
  - Homework: published in advance (we aim for 2-3 weeks)
    - If more time is required, we will work with individual to give them access earlier

Modern GPU architectures

Philipp Krähenbühl, UT Austin

GPUs

- Massively parallel processors
- H100 SXM5
  - 132 Streaming Multiprocessors (SM) per GPU
  - 128 FP32 cores per SM
  - 80GB HBM3 ram
  - 228 KB shared memory / SM



GH100 Full GPU with 144 SMs [1]

[1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022.

GPUs - SM

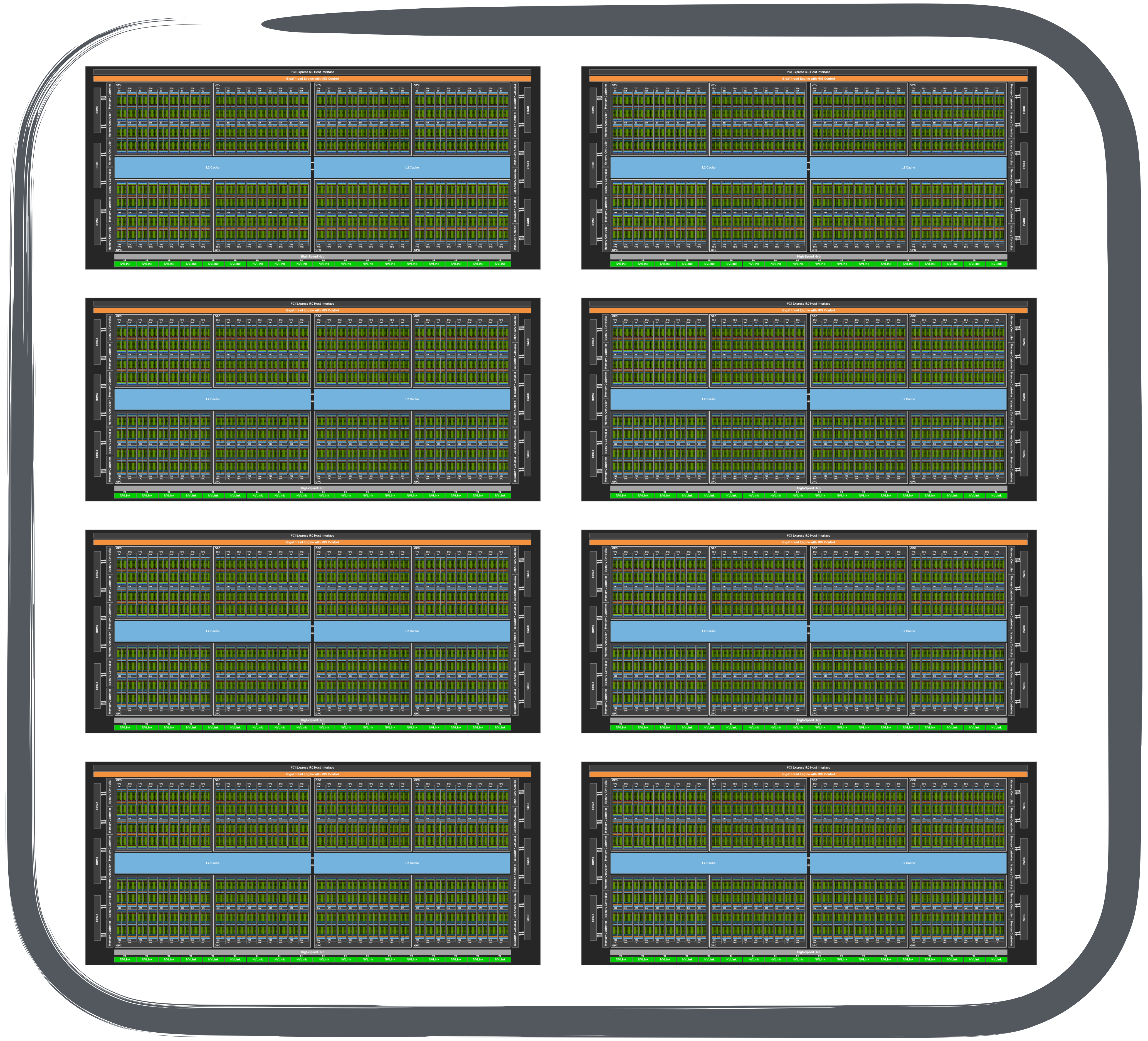
- Streaming Multiprocessors (SM)
  - Individual "CPUs" on chip
  - 4 warps (similar to CPU cores)
- Each warp
  - Tensor Core (matrix multiplier)
  - 32 threads (shared scheduler, dispatcher)

GH100 Streaming Multiprocessor (SM) [1]

[1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022.

GPUs in a node

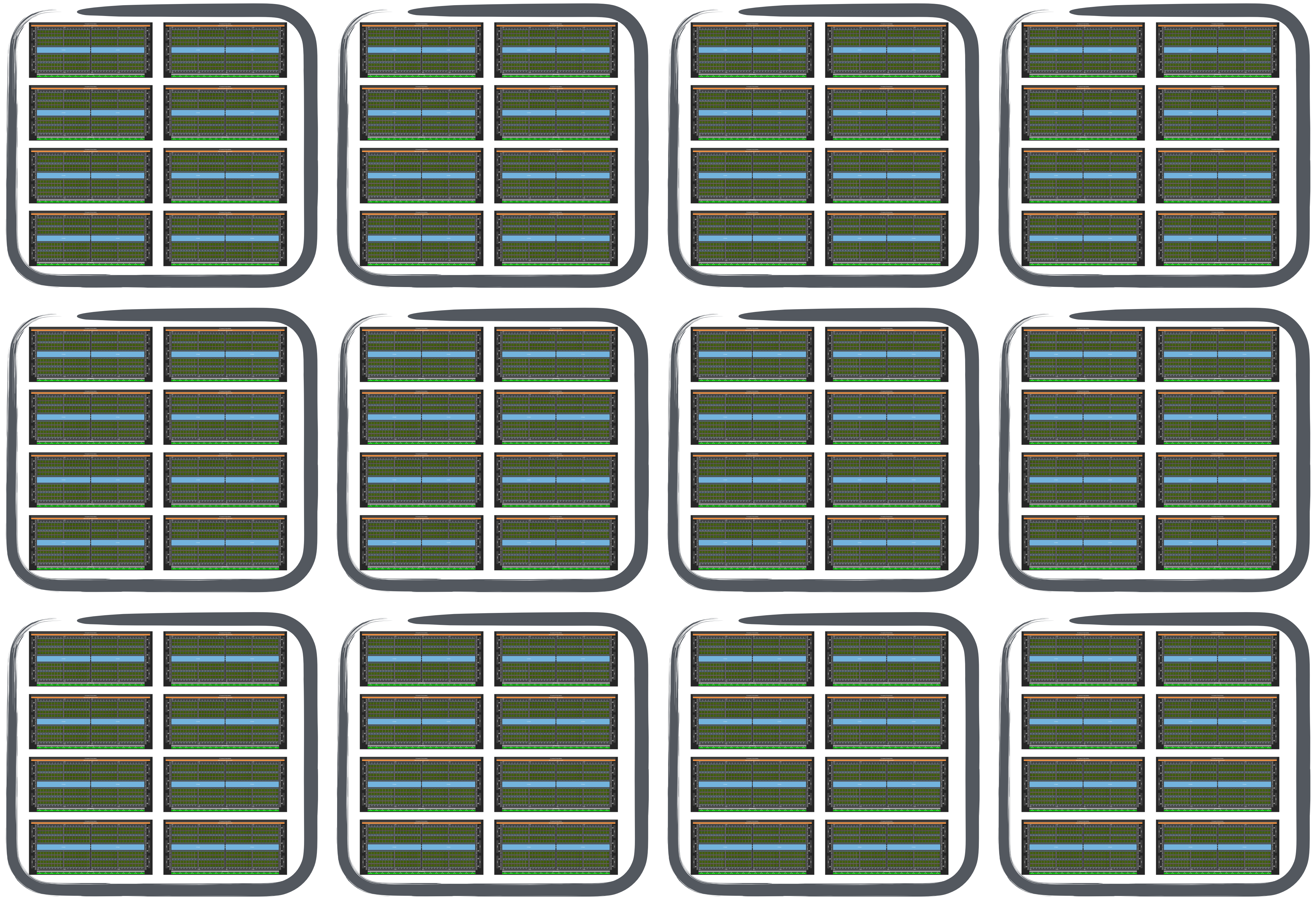
- Compute node
  - 8-16 GPUs per server / node
- Fast / specialized communication between GPUs (NVlink)



Node

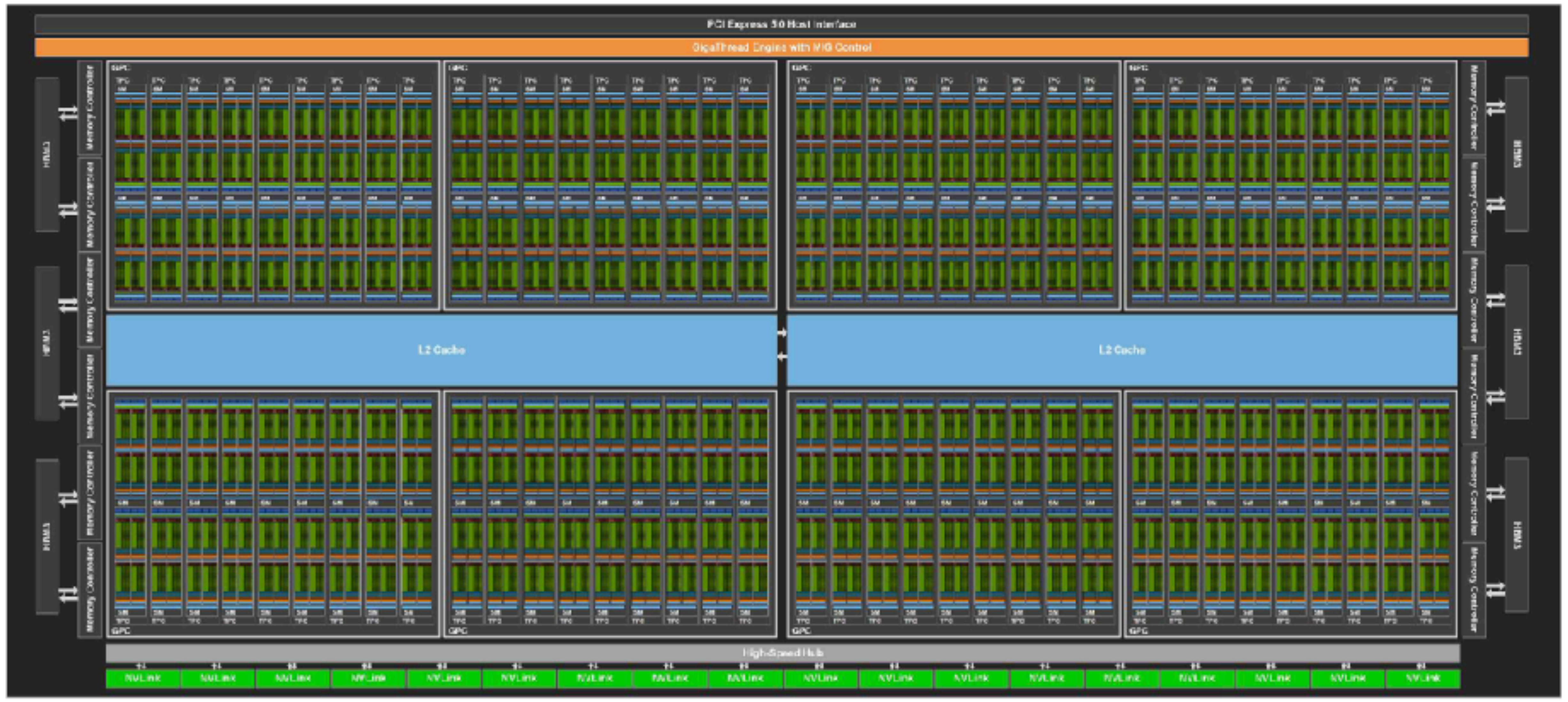
GPUs in a datacenter

- Nodes networks in a datacenter
- Up to 40k nodes with 16 GPUs each
  - 0.42 GigaWatt
  - 40% of nuclear power plant, excluding cooling, other hardware
- We have peaked



[1] Meta. Building Meta's GenAI Infrastructure. 2024.
[2] https://en.wikipedia.org/wiki/Nuclear_power.

GPUs - Mental model



GH100 Full GPU with 144 SMs [1]

- Massively parallel processors
  - Intuitions from CPUs and theoretical CS are often wrong
  - Nearly endless compute
    - On a restricted set of operations
  - Limited memory and memory bandwidth

[1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022.

GPUs - Mental model

A simple example

x_1	x_2	x_3	x_4	x_5	x_6

- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W

- Find the maximum number value for all possible windows

  - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$

- What deep learning operation is this?

GPUs - Mental model

A simple example

x_1	x_2	x_3	x_4	x_5	x_6

- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W

- Find the maximum number value for all possible windows

  - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$

- What deep learning operation is this?

```
def maxpool_1d_brute(x: torch.Tensor, window_size: int):
    """A windowed maximum pooling operation for 1D
tensors."""
    output = x.new_zeros(x.size(0) - window_size + 1)
    for i in range(output.size(0)):
        for j in range(window_size):
            output[i] = max(output[i], x[i + j])
    return output
```

Compute: $O(|\mathbf{x}|W)$

GPUs - Mental model

A simple example

x_1	x_2	x_3	x_4	x_5	x_6

- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W
- Find the maximum number value for all possible windows
  - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$
- What deep learning operation is this?

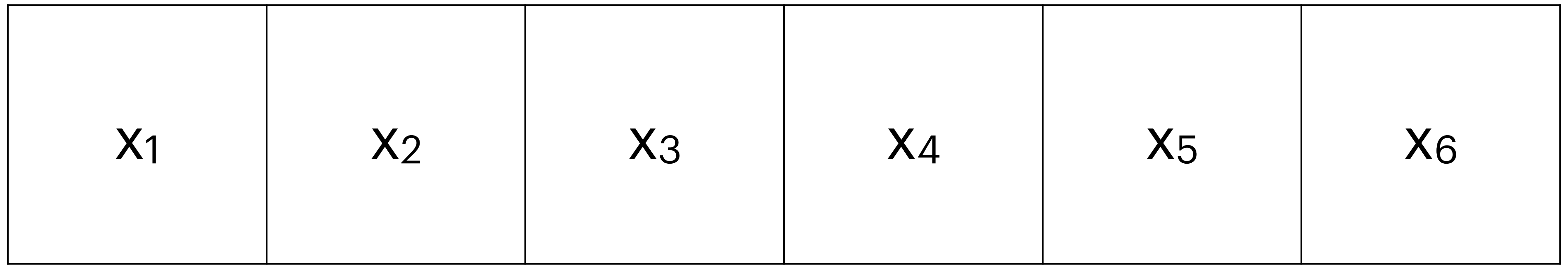
```
def maxpool_1d_heap(x: torch.Tensor, window_size: int):
    """A windowed maximum pooling operation for 1D 
tensors."""
    output = x.new_zeros(x.size(0) - window_size + 1)

    h = []
    for i in range(x.size(0)):
        heapq.heappush(h, (-x[i].item(), i))
        if i >= window_size - 1:
            while h[0][1] <= i - window_size:
                heapq.heappop(h)
            output[i - window_size + 1] = -h[0][0]
    return output
```

Compute: $O(|\mathbf{x}|\log W)$

GPUs - Mental model

A simple example in CUDA



- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W

- Find the maximum number value for all possible windows

  - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$

- What deep learning operation is this?

Compute: $O(|\mathbf{x}|W)$

Memory access: $O(|\mathbf{x}|W)$

GPUs - Mental model

A simple example in CUDA

x_1	x_2	x_3	x_4	x_5	x_6

- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W
- Find the maximum number value for all possible windows
  - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$
- What deep learning operation is this?

Compute: $O(|\mathbf{x}|W/G)$

Memory access: $O(|\mathbf{x}|W/G)$

GPUs - Mental model

A simple example in CUDA

x_1	x_2	x_3	x_4	x_5	x_6

- You are given a series of numbers $\mathbf{x}$ and a **fixed** window size W
- Find the maximum number value for all possible windows
    - $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$
- What deep learning operation is this?

Compute: $O(|\mathbf{x}|W)$

Memory access: $O\left(|\mathbf{x}|\frac{W}{S}\right)$

S: shared memory size

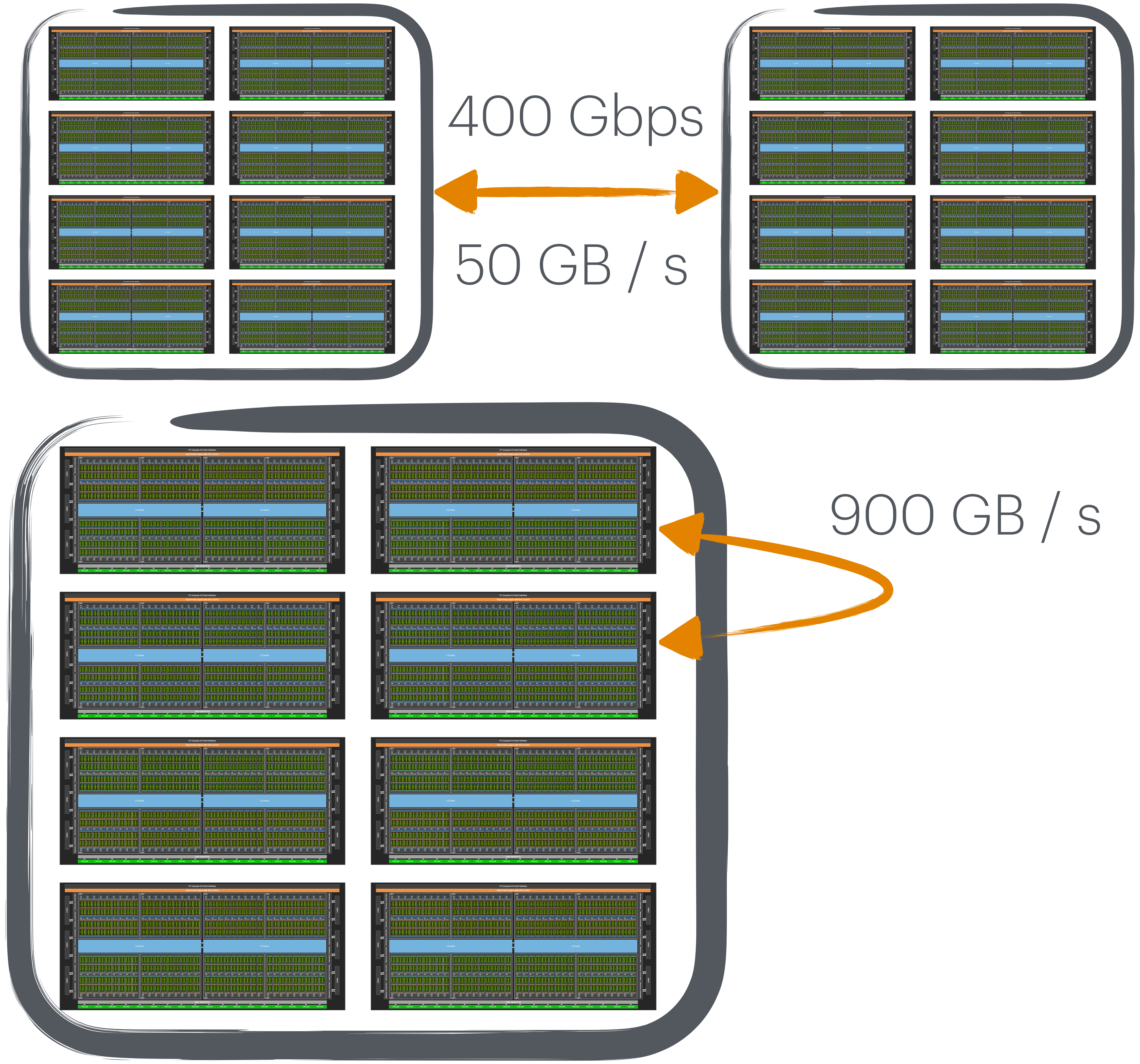
GPUs - Mental model

What have we learned?

x_1	x_2	x_3	x_4	x_5	x_6

- Memory access matters
  - Reads from global memory are expensive
  - Computation is cheap

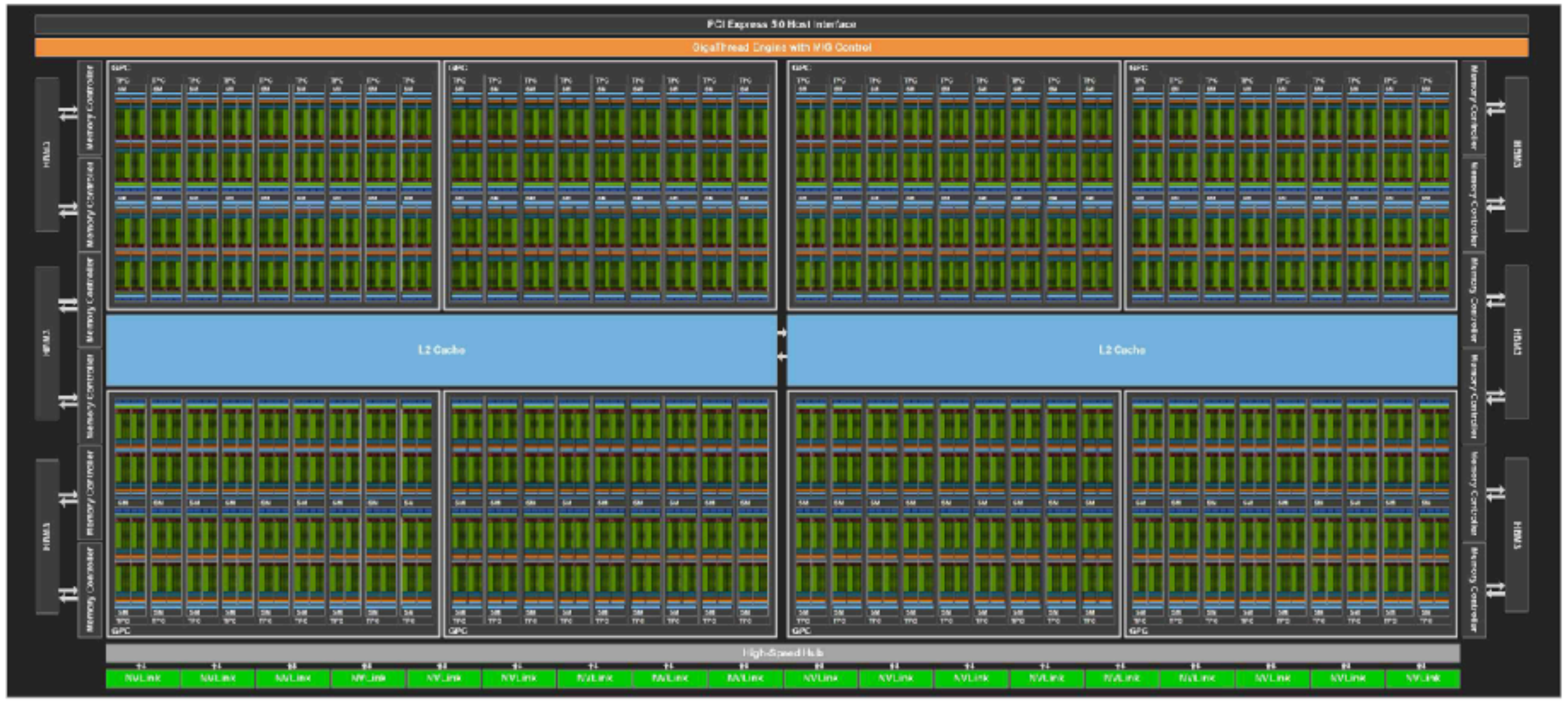
GPUs - Memory Bandwidth



- Node to node communication
  - RDMA/IB: 50GB / s
- GPU to GPU communication (within node)
  - NVLink: 900 GB / s
- GPU memory bandwidth
  - HBM3->shared mem: 3.35 TB / s
- Peak flops: 130-1000 teraFLOPS @ BF16

[1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022.

Modern GPU architectures



GH100 Full GPU with 144 SMs [1]

- Near infinite compute
- Memory bandwidth and size limits
  - Order of magnitude slower
    GPU -> Node -> Datacenter
- Approaching limits of power consumption, and physical limits in manufacture

[1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022.

References

- [1] NVIDIA. NVIDIA H100 Tensor Core GPU Architecture. 2022. (link)
- [2] Meta. Building Meta's GenAI Infrastructure. 2024 (link)

Tensors

Philipp Krähenbühl, UT Austin

What is a Tensor?

- An array of numbers (of the same type)
- Examples:
  - 1D tensor: Vector, Waveform
  - 2D tensor: Matrix
  - 3D tensor: Image
  - 4D tensor: Video

Tensors in PyTorch

- Notebook

GPUs - Mental model

The secret solution

x_1	x_2	x_3	x_4	x_5	x_6

- $e_i = \max(x_i, x_{i+1}, \ldots, x_{i+W-1})$
- $e_i = \max\left(\max(x_i, \ldots, x_K), \max(x_{K+1}, \ldots, x_{i+W-1})\right)$
- $e_{i+1} = \max\left(\max(x_{i+1}, \ldots, x_K), \max(x_{K+1}, \ldots, x_{i+W})\right)$

Compute: $O(|\mathbf{x}|)$

Memory access: $O(|\mathbf{x}|)$