

# LLM Sys

## GPU Programming

Lei Li

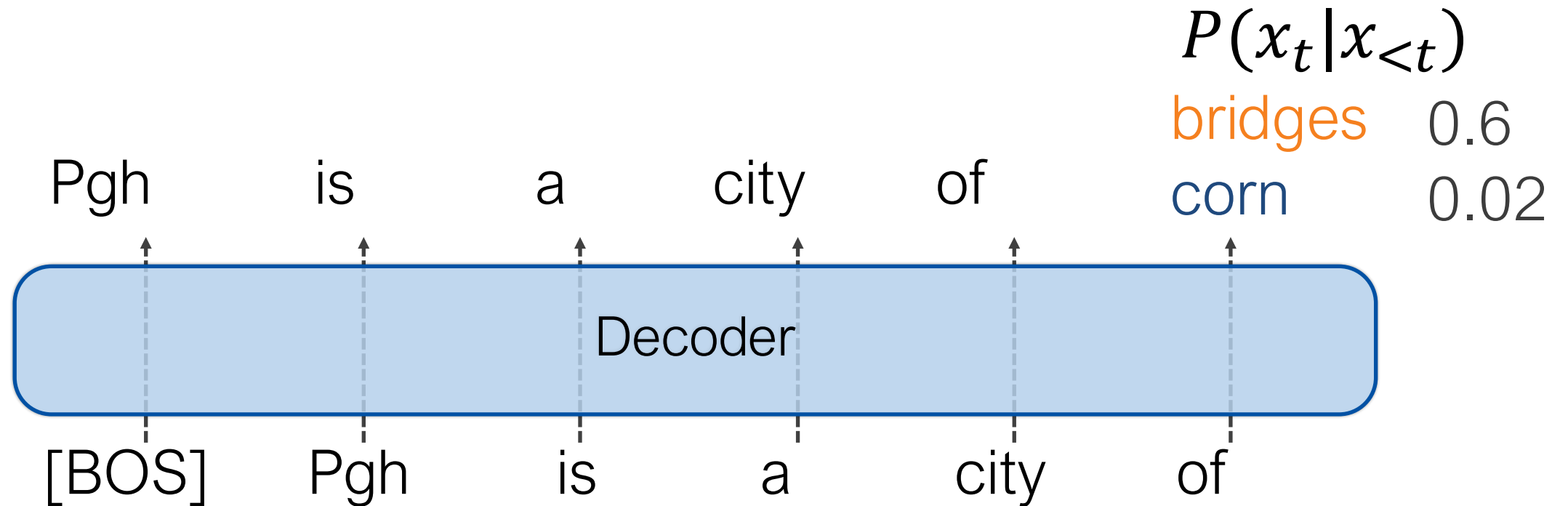


Language  
Technologies  
Institute

**Carnegie Mellon University**  
School of Computer Science

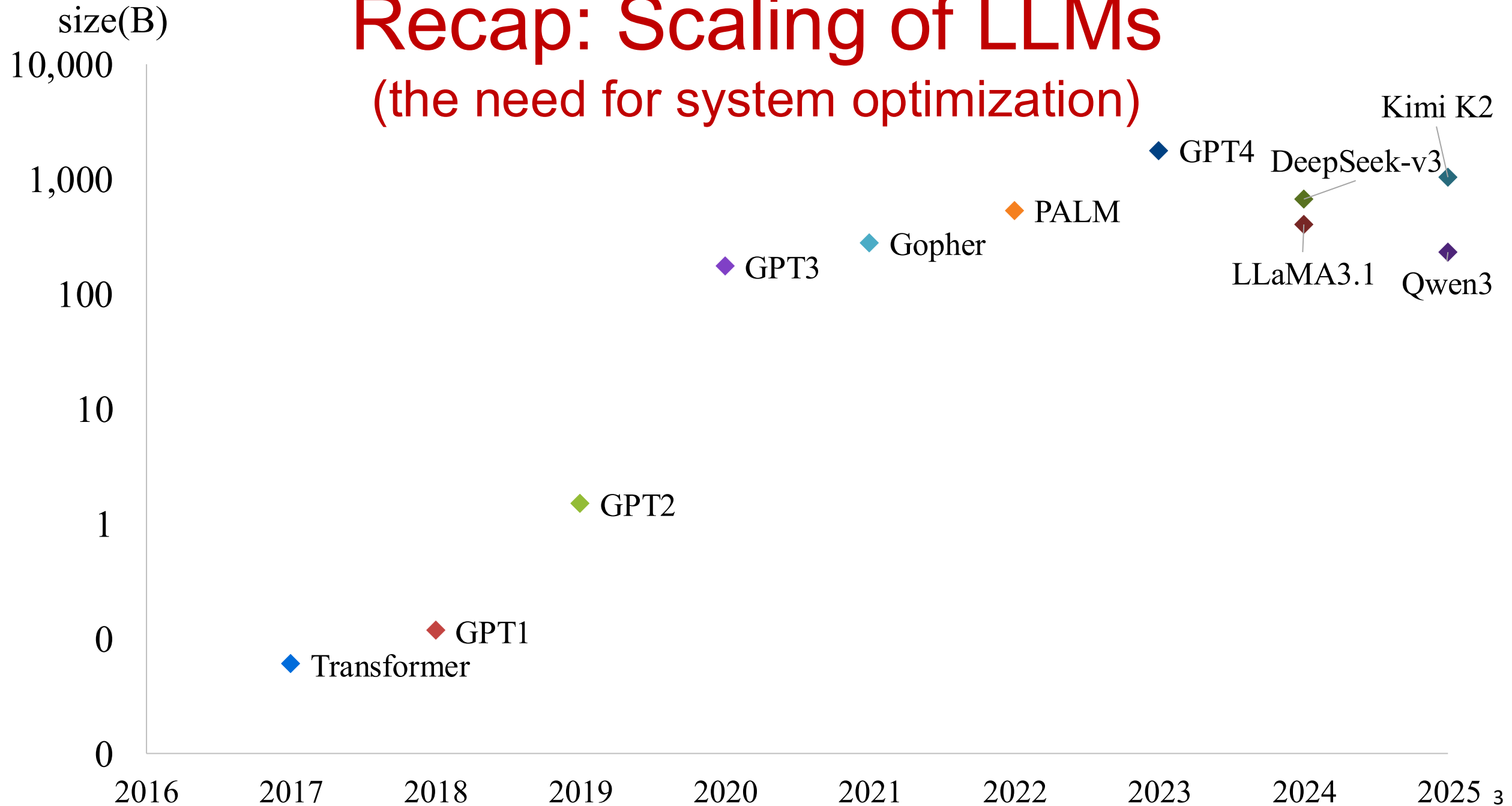
# Recap: (Autoregressive) Language Model

$$P(x_{1..T}) = \prod_{t=1}^T P(x_{t+1} | x_{1..t})$$



# Recap: Scaling of LLMs

(the need for system optimization)



# Recap: Important Topics in LLM systems

- Programming model
  - relies on good abstraction
- Latency/Throughput
  - Data movement
  - Computation vs memory
- Reliability
- Security

# Outline

- ➡ • Neural Network Layer and low-level operators
- Components of A GPU Server
- GPU Architecture
- Program Execution on GPU

# Text Classification

Classifying the sentiment of online movie reviews. (Positive, negative, neutral)

Spider-Man is an almost-perfect extension of the experience of reading comic-book adventures.

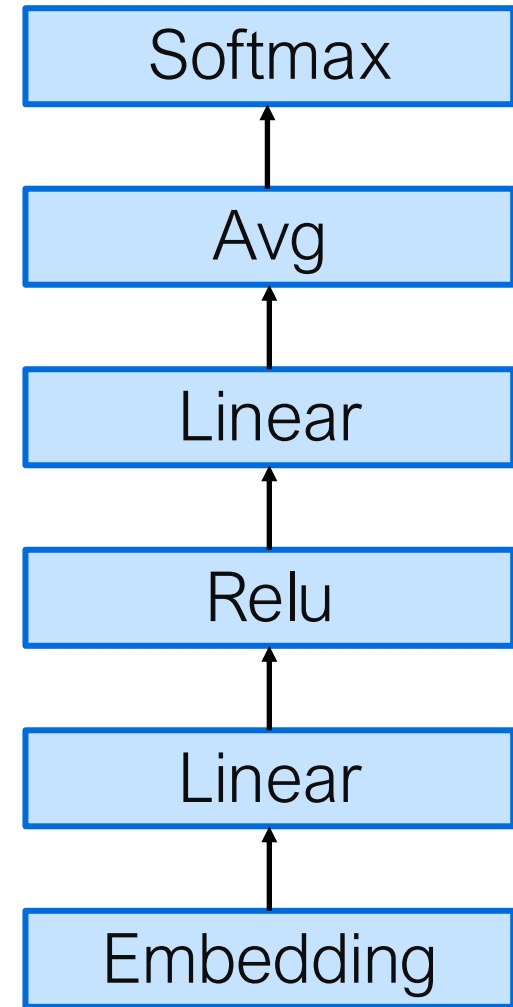


It was a boring! It was a waste of a movie to even be made. It should have been called a family reunion.



# A Simple Feedforward Neural Network

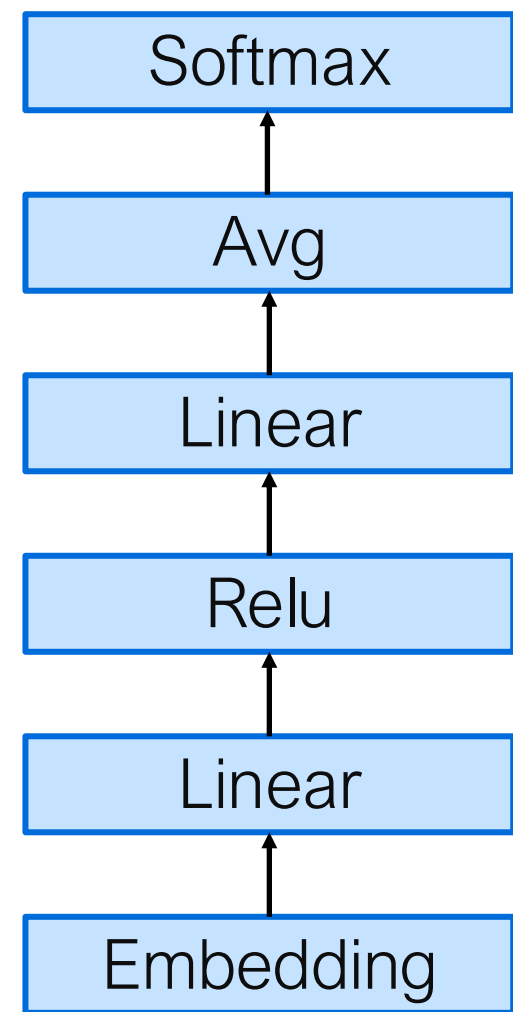
- Neural network layers
  - Embedding (lookup table)
  - Linear
  - Relu
  - Average pooling
  - Softmax



“It is a good movie”

# Low-level Computing Operators

- Matrix multiplication
- Element-wise ops (add, scale, relu)
- Reduce ops (sum, avg)
- Efficient computation requires GPUs



“It is a good movie”



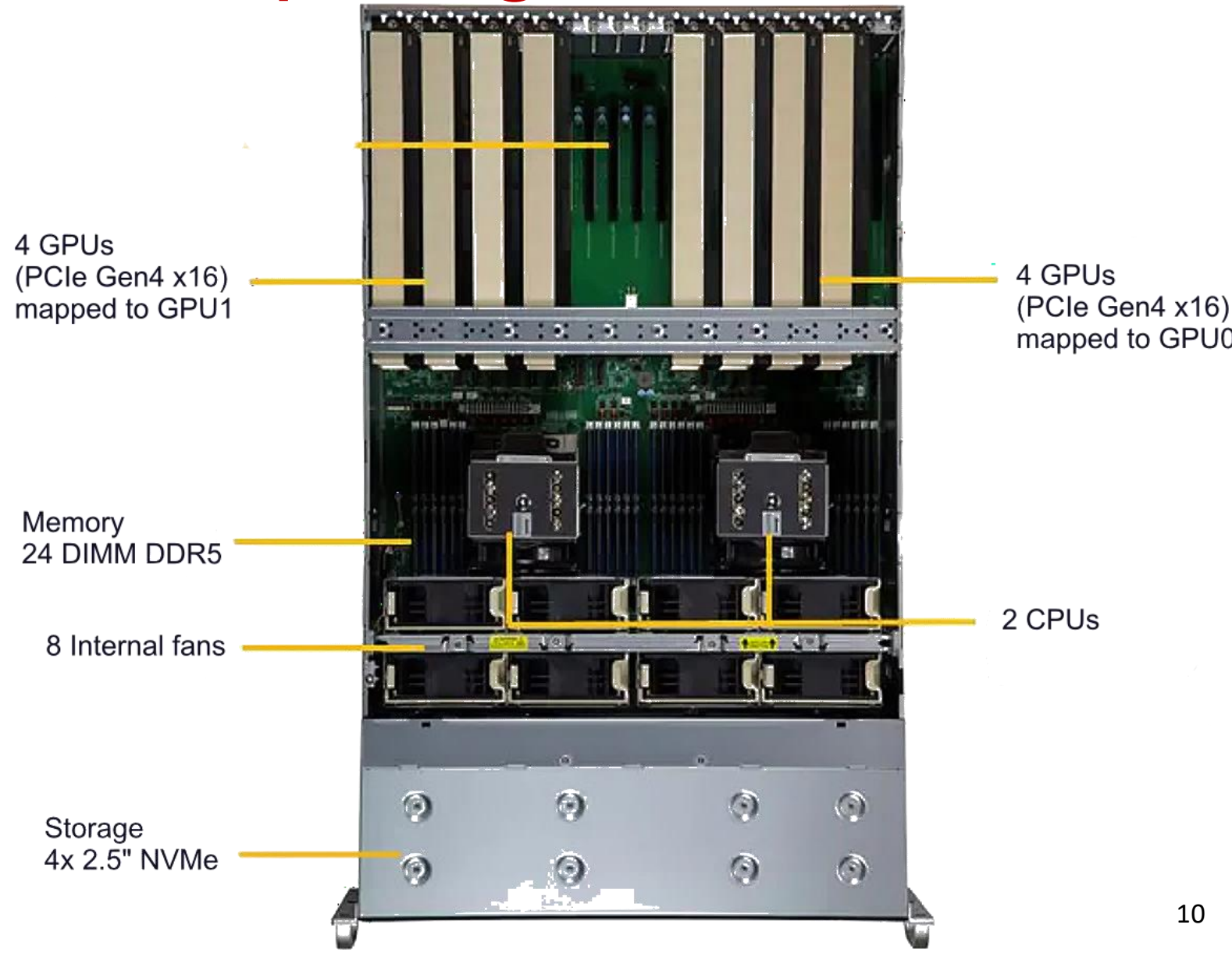
# Outline

- Neural Network Layer and low-level operators
- ➔ • Components of A GPU Server
  - GPU Architecture
  - Program Execution on GPU

# A Modern Computing Server

## A Sample Config (my lab)

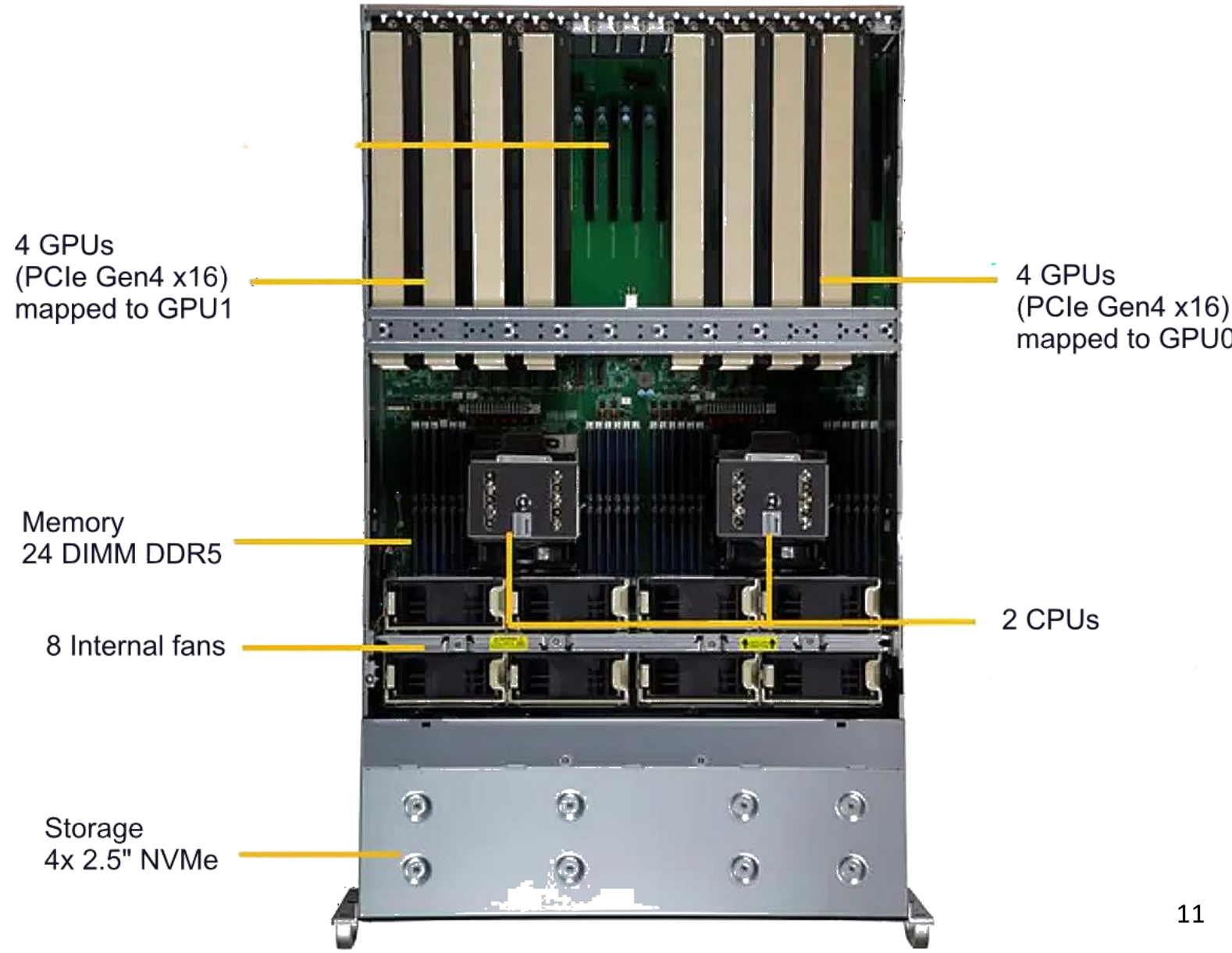
- CX4860s-EK9 4U server
- 2x AMD EPYC 9354 CPU
- 16x 64GB DDR5 mem
- 4x Intel D7 P5520
- 15.36TB Gen4 NVMe SSD
- 8x Nvidia A6000 48GB
- 4x 2slot NVLink



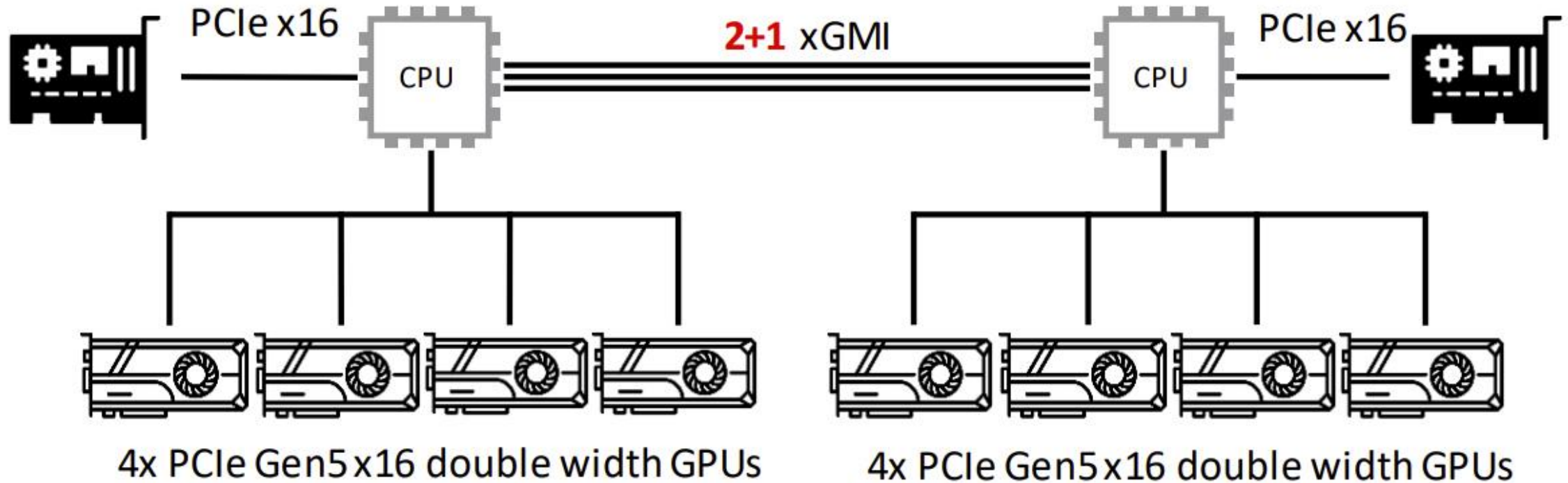
# Communication

## GPU-to-GPU

- NVLink 112.5 GB/s
- PCIe Gen4 32 GB/s (16 lanes x 2 GB/s per lane)



# Modern Computing Server Architecture



Why is it relevant?

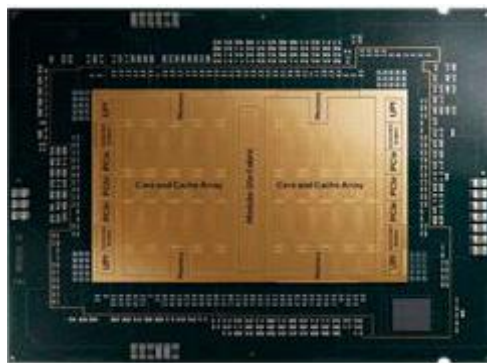
Considering moving gradients from one GPU to another  
(in data parallel)

# Computing Devices

## CPU



AMD EPYC 9754  
128cores  
256threads  
2.25GHz  
256MB L3



Intel Xeon  
8593Q  
64cores  
128threads  
2.2GHz  
320MB L3

## GPU




Nvidia A6000  
10,752 cores  
48GB  
38.7 TFLOPS

## FPGA



More powerful than the #1  
supercomputer in 2001

# Outline

- Neural Network Layer and low-level operators
- Components of A GPU Server
-  • GPU Architecture
- Program Execution on GPU



# GPU Lineup

Architecture	Blackwell	Hopper	Ampere
GPU Name	NVIDIA B200	NVIDIA H100	NVIDIA A100
FP64	37 teraFLOPS	34 teraFLOPS	9.7 teraFLOPS
FP64 Tensor Core	37 teraFLOPS	67 teraFLOPS	19.5 teraFLOPS
FP32	75 teraFLOPS	67 teraFLOPS	19.5 teraFLOPS
FP32 Tensor Core	2.2 petaFLOPS	989 teraFLOPS	312 teraFLOPS
FP16/BF16 Tensor Core	4.5 petaFLOPS	1979 teraFLOPS	624 teraFLOPS
INT8 Tensor Core	9 petaOPs	3958 teraOPs	1248 teraOPs
FP8 Tensor Core	9 petaFLOPS	3958 teraFLOPS	-
FP4 Tensor Core	18 petaFLOPS	-	-
GPU Memory	192GB HBM3e	80GB HBM3	80GB HBM2e
Memory Bandwidth	Up to 7.7TB/s	3.2TB/s	2TB/s

# GPU Architecture

Nvidia RTX A6000  
(GA102)  
84 SMs  
(Streaming  
Multiprocessors)  
12 memory  
controller (32bit  
ea.) total 384bit.  
6MB L2 cache.





# H100 Architecture

132/114 SMs, 80G HBM3



# Streaming Multiprocessor

4 partitions per SM, each with 32 cores → 32 threads each

128 cores per SM

64KB register per partition (fastest to access)

128KB shared L1 cache per SM

128 FP32 operations in one cycle



# H100 SM

4 partitions per SM, each with 32 cores → 32 threads each

128 cores per SM

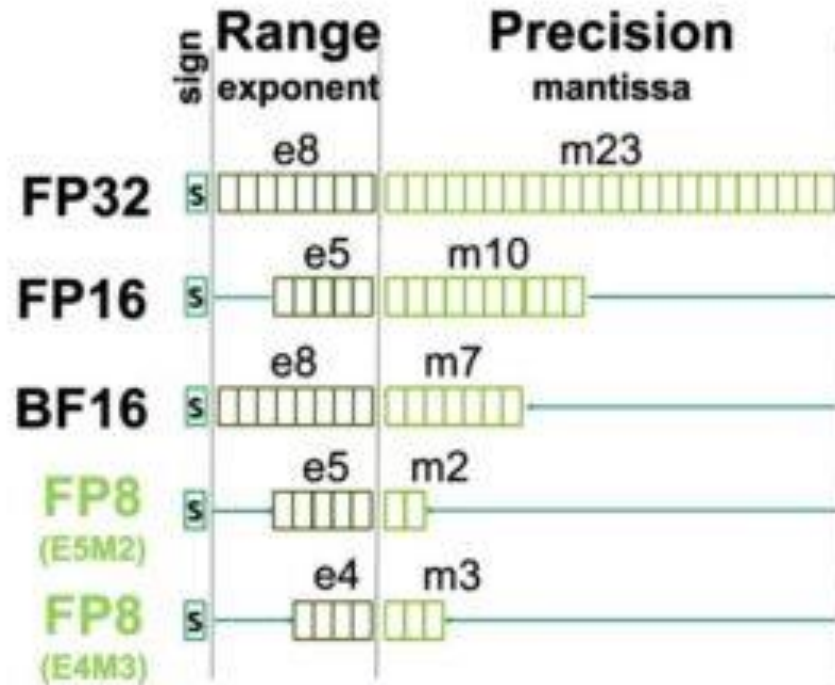
64KB register per partition (fastest to access)

256KB shared L1 cache per SM

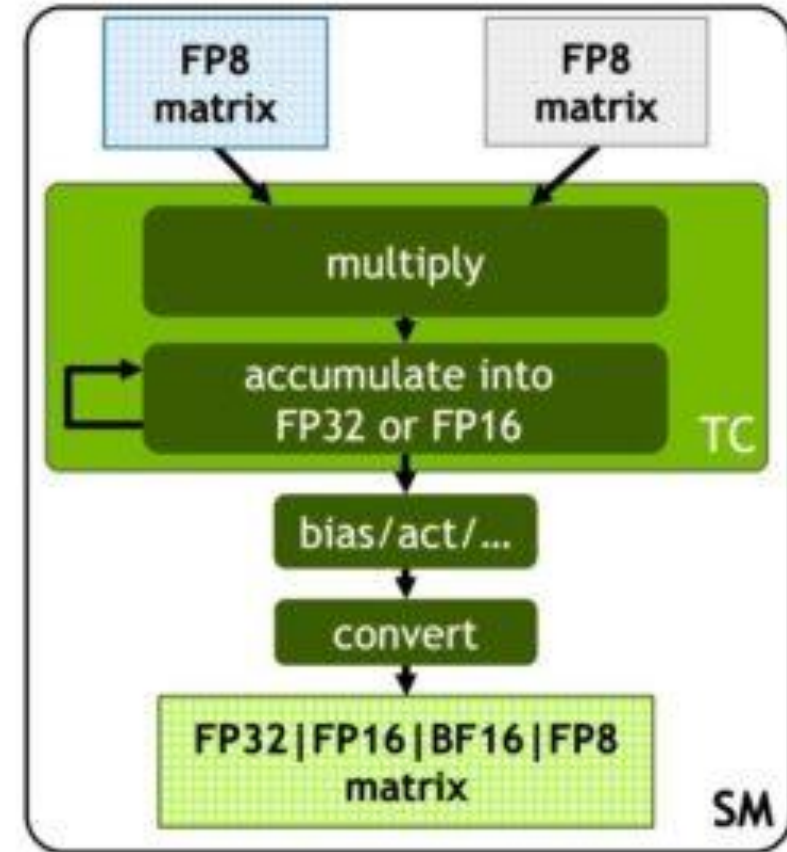




# New in H100, FP8 operations



Allocate 1 bit to either range or precision



Support for multiple accumulator and output types

# CPU vs. GPU

	CPU (AMD EPYC 9754)	GPU (A6000)
num. threads	256	10752
clock	2.25 GHz	1.8 GHz
compute	576 GFlops	38.7 TFlops
Power	360W	300W

# Outline

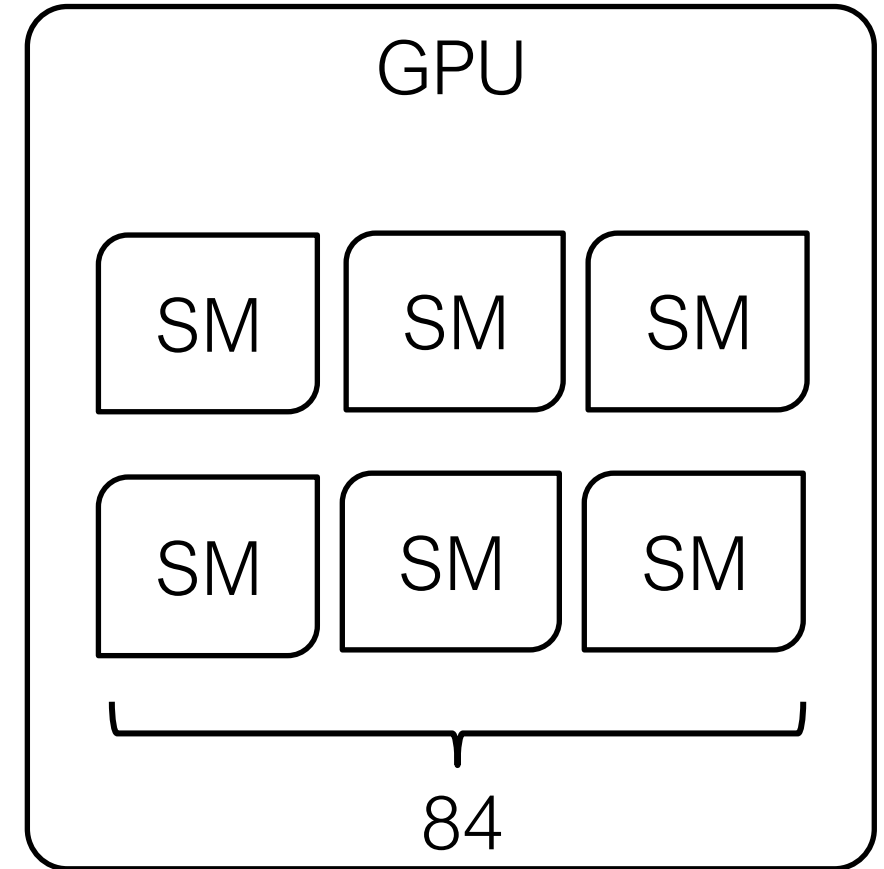
- Neural Network Layer and low-level operators
- Components of A GPU Server
- GPU Architecture
- ➔ • Program Execution on GPU

# GPU Programming Model

- CPU – host
  - Run normal program (C++)
- GPU – device
  - Run cuda kernel code
- CUDA: one part runs on CPU, one part runs on GPU
- Needs to move data between system memory and GPU memory

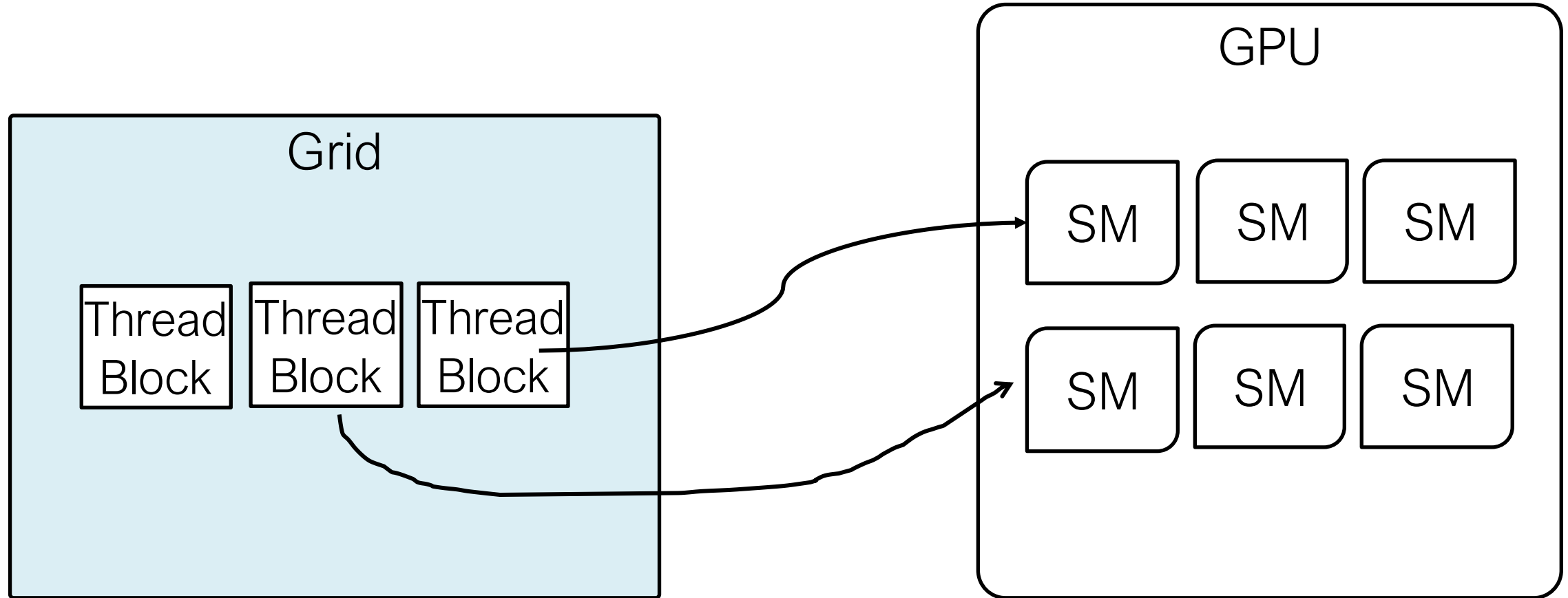
# SIMT Execution on GPU

- Single Instruction Multiple Threads
- Threads are grouped into Thread Blocks
- Thread Blocks are grouped into Grid
- Kernel executed as Grid of Blocks of Threads





# How instructions are executed on GPU



# How Kernel Threads are Executed

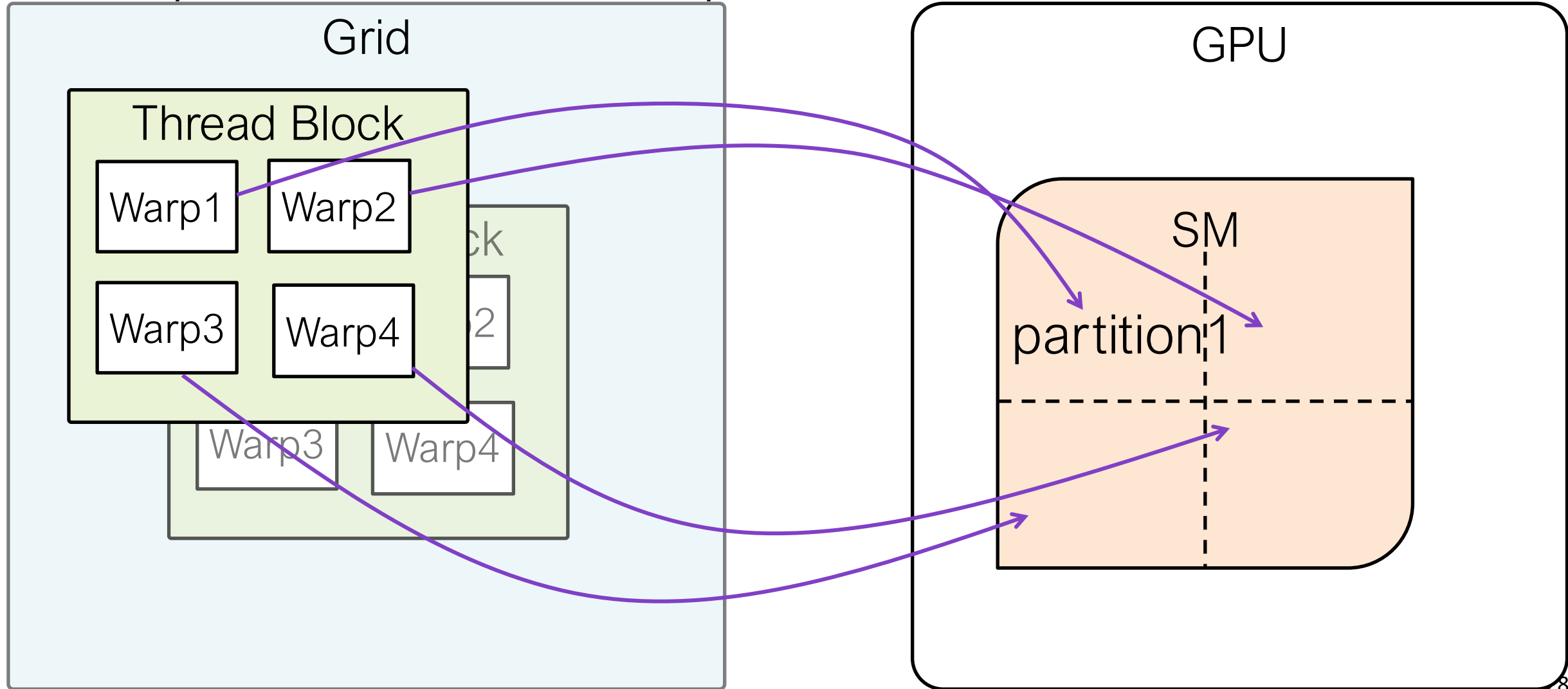
- SM partition a thread block into warps
- Warp is the unit of GPU creating, managing, scheduling and executing threads
- Each warp contains **32** threads (why 32?)
  - Start at same program address
  - Have own program counter and registers
  - Execute one common instruction at a cycle
  - Can branch and execute independently

# Warp Execution on GPU

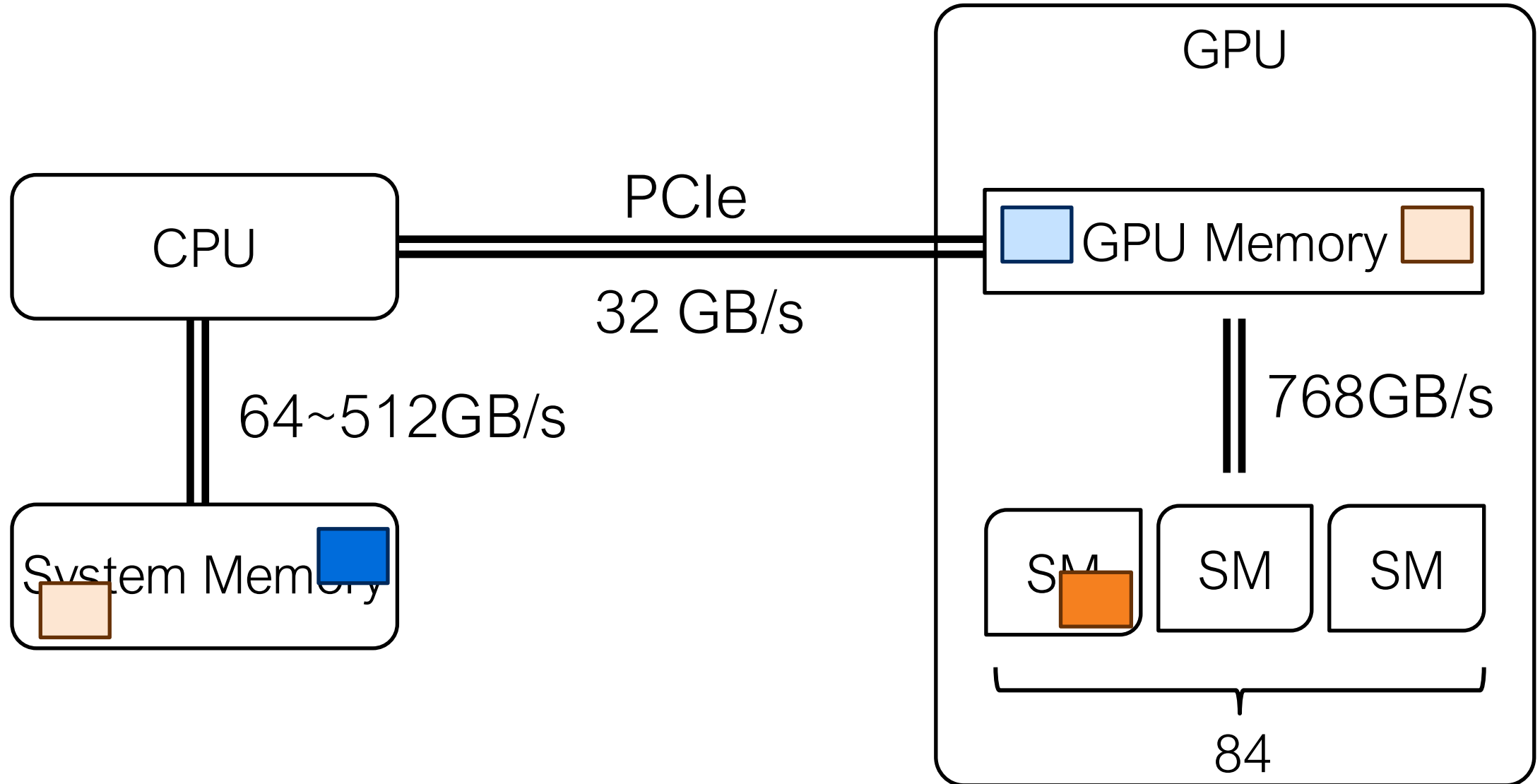
- Execution context stays on SM for lifetime of warp (Program counter, Registers, Shared memory)
- Warp-to-warp context switch is instant
- At running time, warp scheduler
  - Chooses warp with active threads
  - Issues instruction to warp's threads
- Number of warps on SM depends on mem requested and available

# Executing one Thread block on one SM

4 warps can be executed in parallel at one time on each SM



# CPU-GPU Data Movement

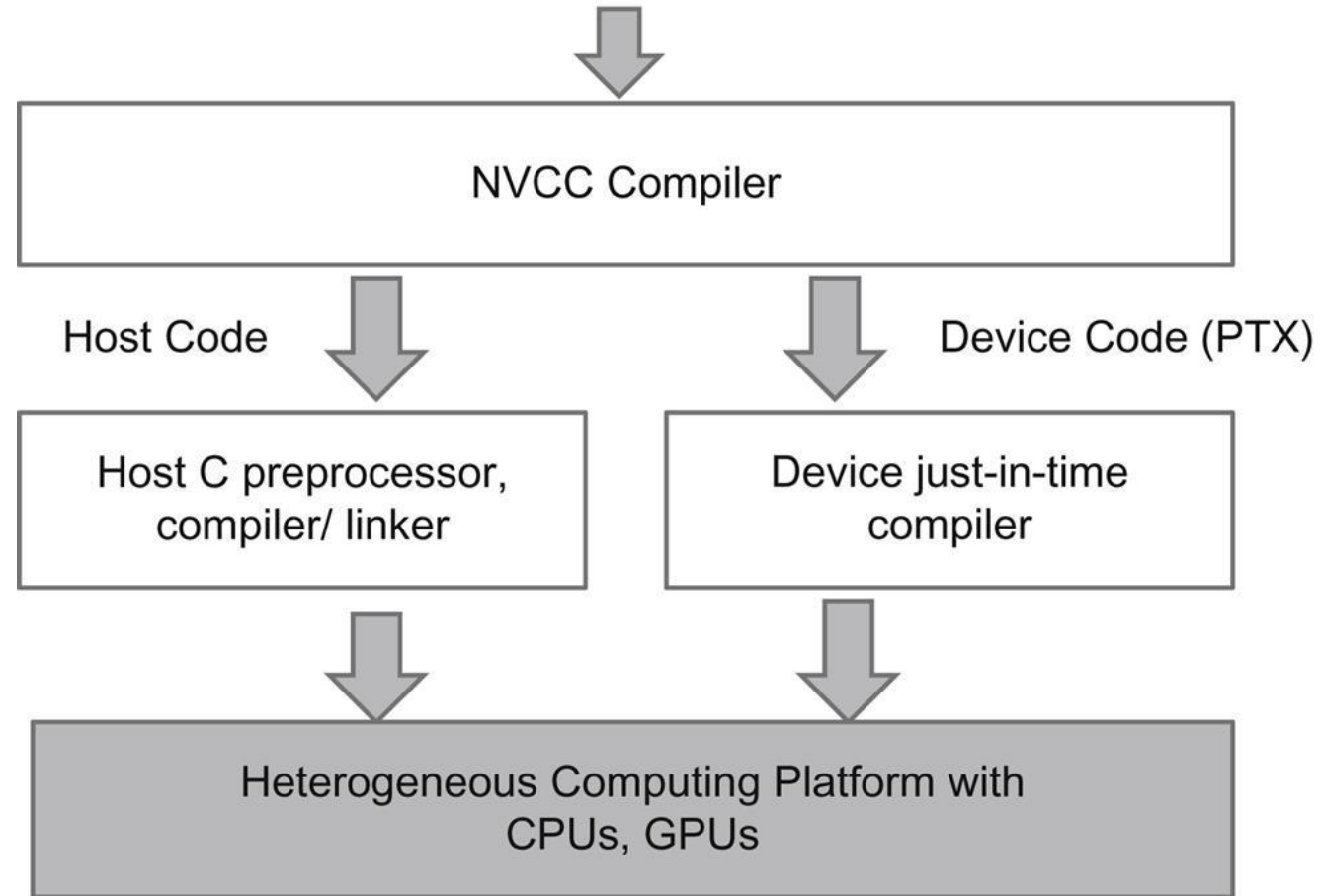


# CUDA Kernel

- Each kernel is a function (program) that runs on GPU
- Program itself is serial
- Can simultaneously run many (10k) threads at the same time
- Using thread index to compute on right portion of data

# Compiling CUDA Code

Integrated C programs with CUDA extensions



```
nvcc -o output.so --shared src.cu -Xcompiler -fPIC
```

# Summary

- Neural network
  - is composed of layers, each layer defines input and output vectors/embeddings
  - Each layer's computation consists of low-level operators, which is executed on a computing device (GPU, CPU, FPGA, etc)



# Summary

- GPU is composed of
  - streaming processing units (SMs)
    - each with four partitions of 32 cores
    - shared L1 cache
  - memory
  - L2 cache: share with all SMs
- Threads organized in
  - grid of thread blocks
  - each block is divided into warps running in parallel on one SM.

# Next

- Write GPU Programs

# Assignment 1

<https://llmsystem.github.io/llmsystemhomework/>

# Quiz 1.2 on canvas

- Due today