Hardware-Software Co-Design

Abhinav Bhatele, Daniel Nichols

UNIVERSITY OF MARYLAND

Announcements

- For those that haven't presented, submit videos by May 1
- Extra credit due May 7
- Exam grades out; submit regrade requests by Friday 4/25

What is HW/SW Co-Design?

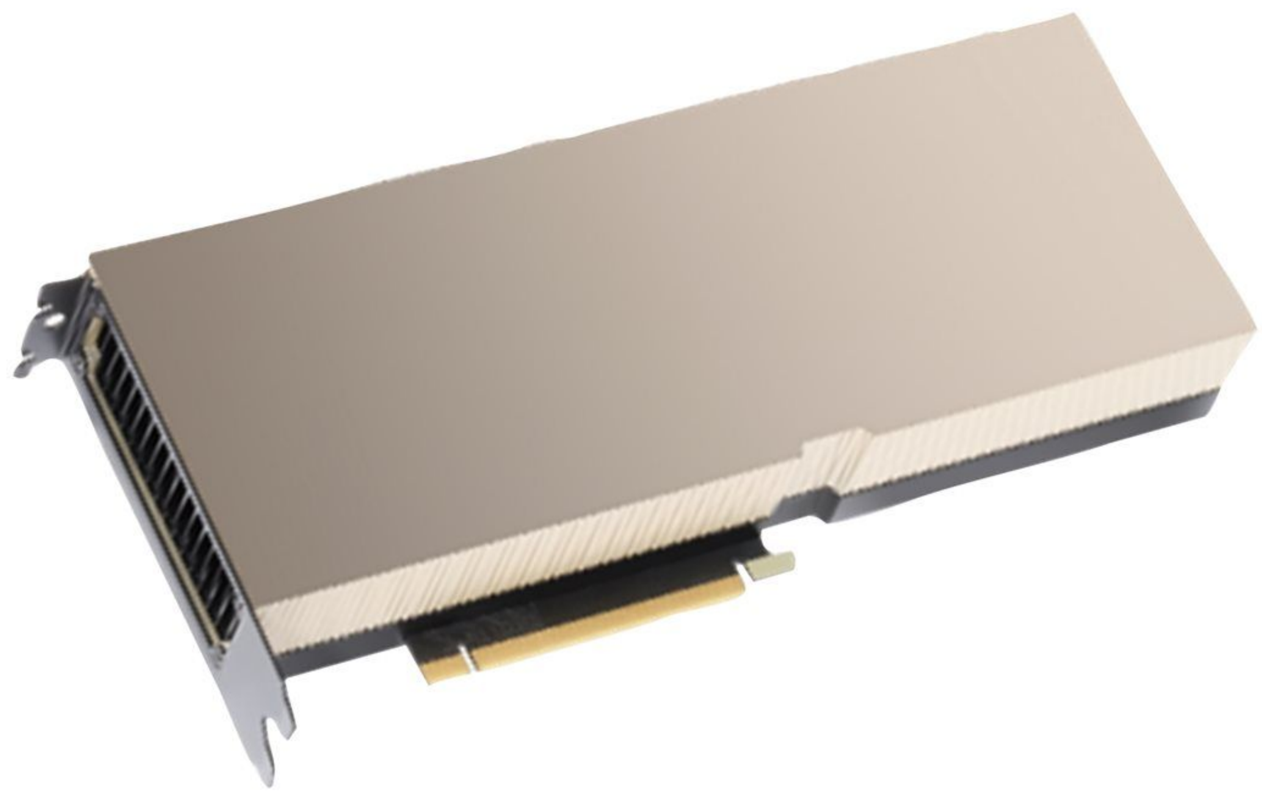
- So far we have been changing our algorithms to optimally match hardware
- But what if we changed both?

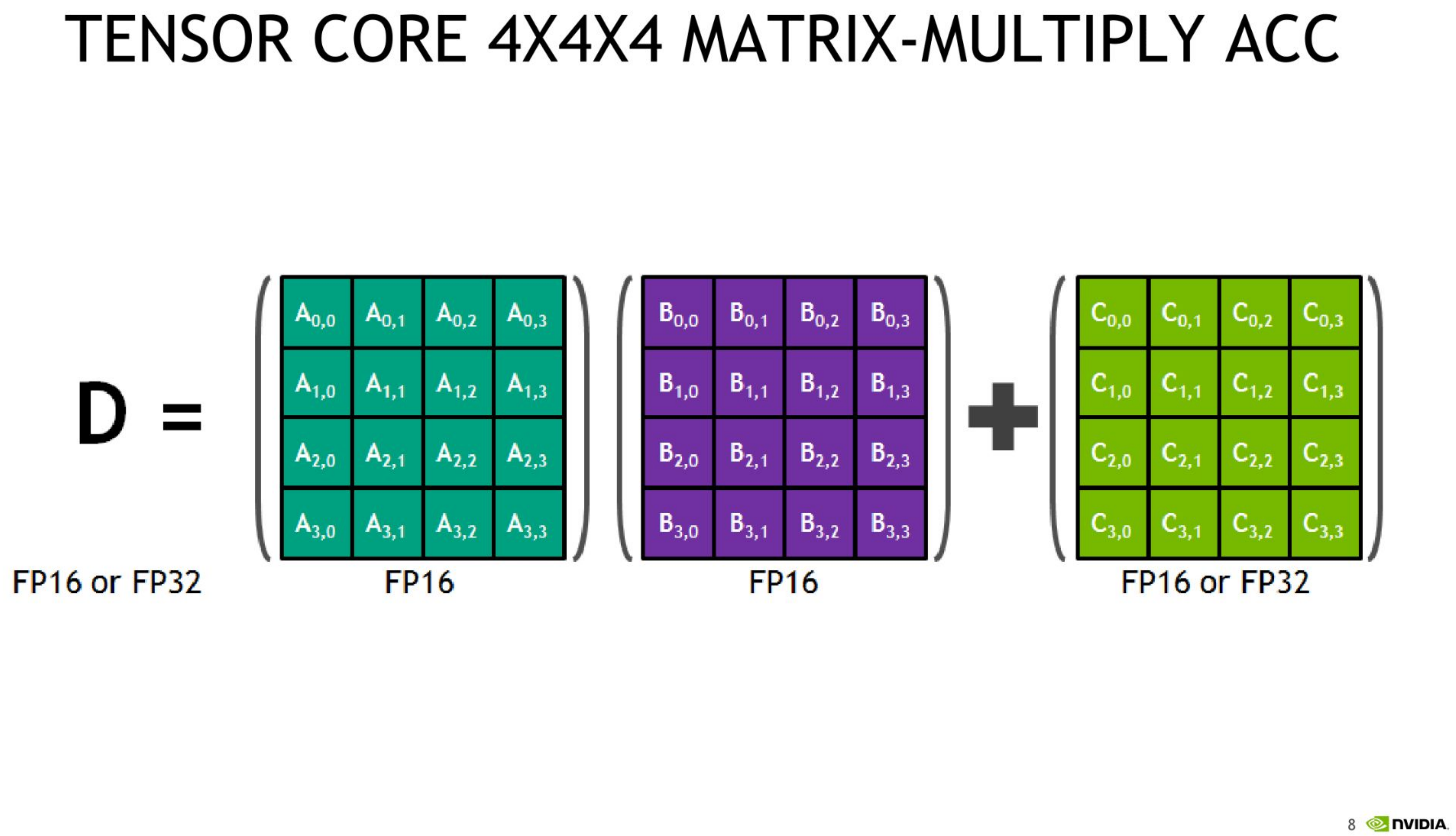
What are some HW inefficiencies we've seen often in this class?

Types of HW/SW Co-Design

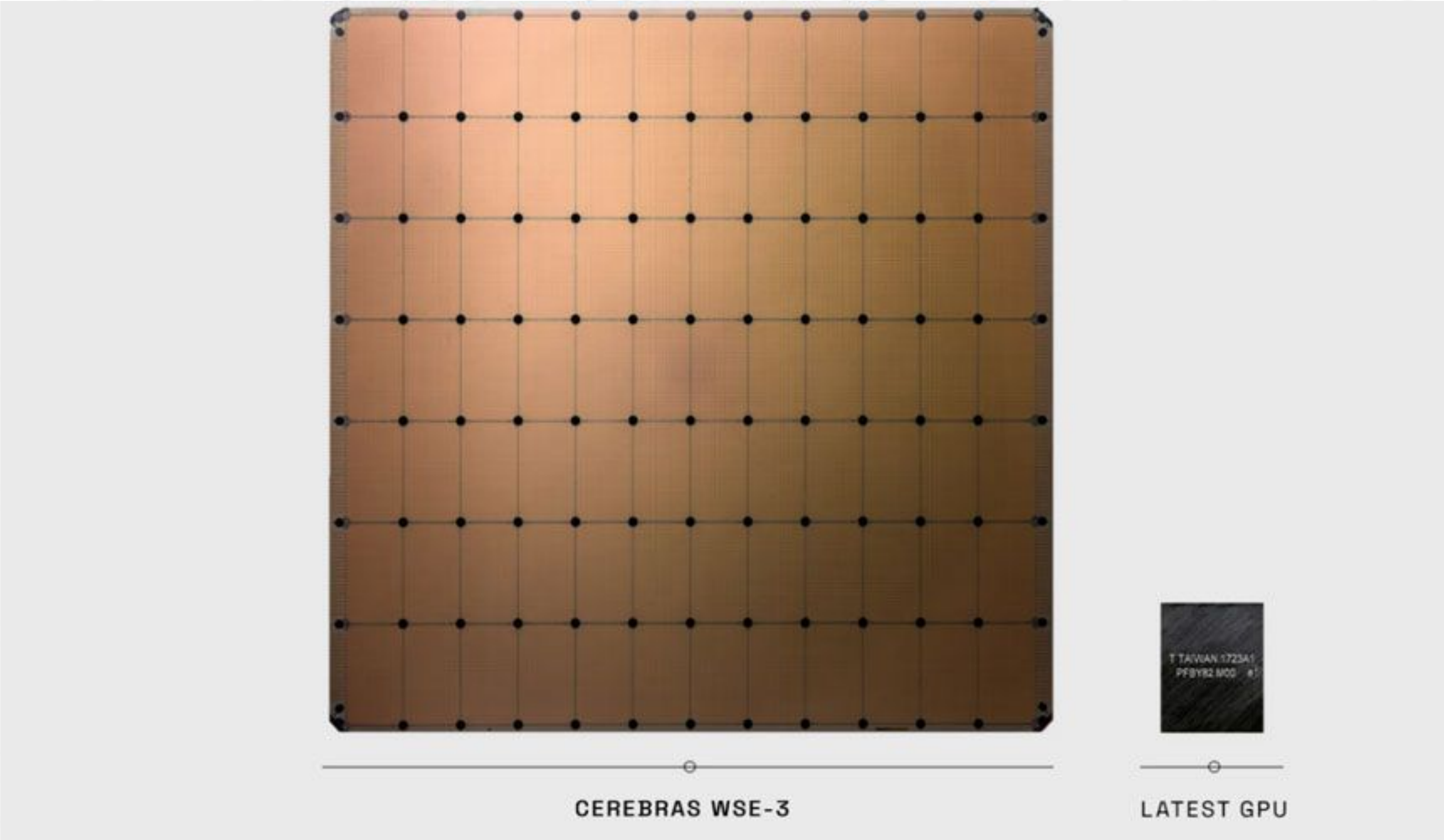
- Standalone accelerators for specific domains
  - great efficiency but not very general
- Extend existing hardware with task specific components
  - great efficiency but can mess with original performance
- Improve existing hardware

Examples

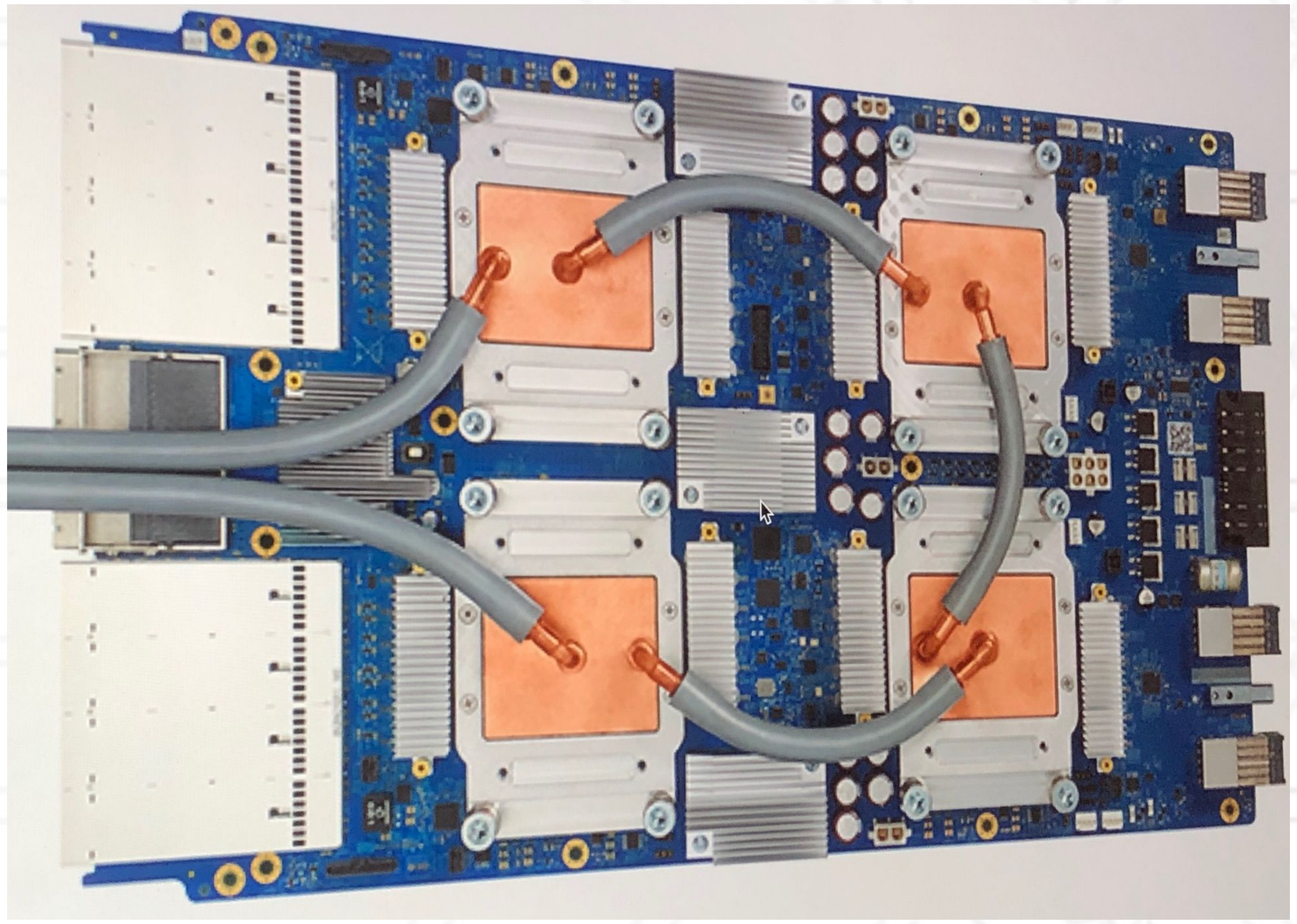


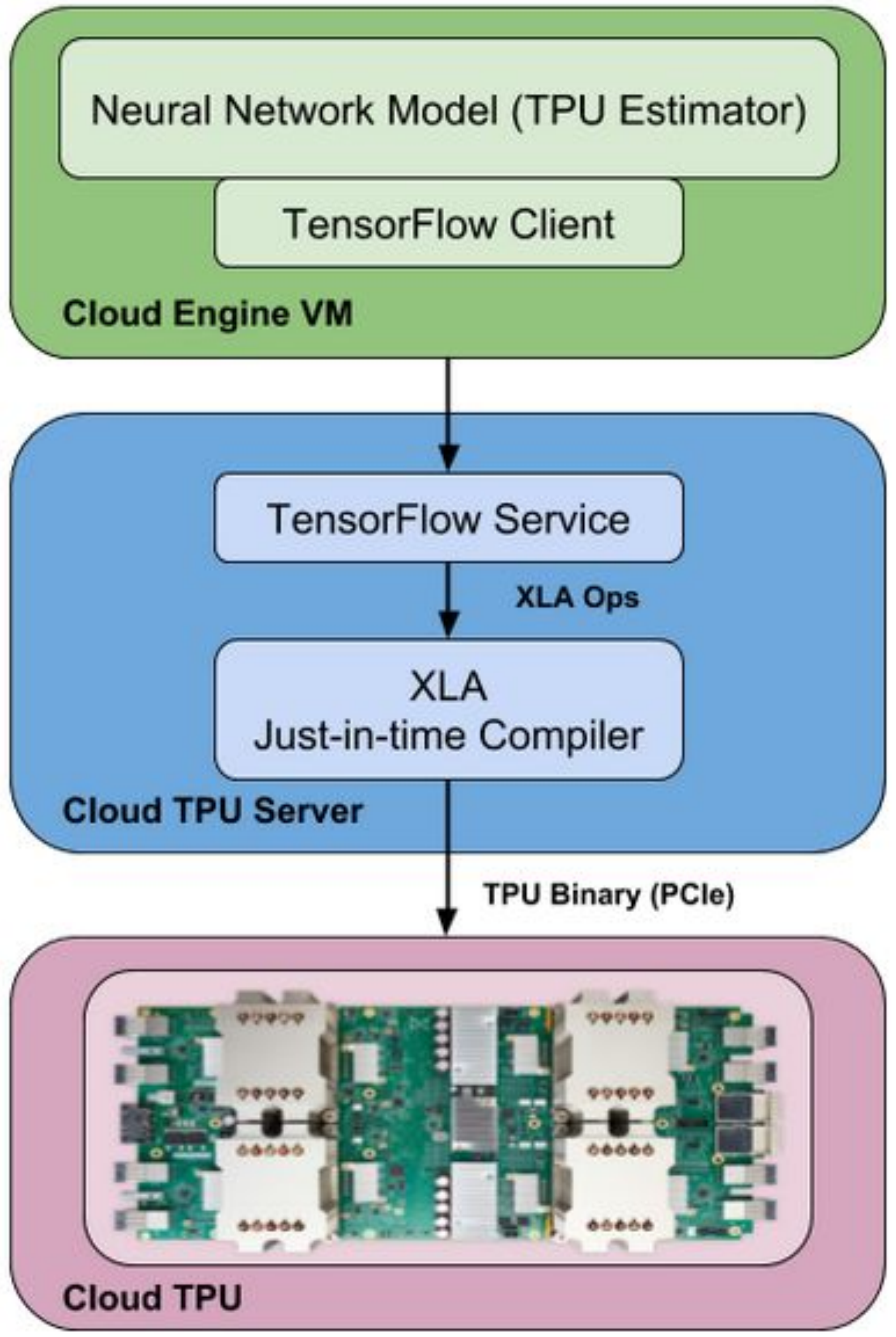

TENSOR CORE 4X4X4 MATRIX-MULTIPLY ACC
D =
FP16 or FP32
FP16
FP16
FP16 or FP32
8
NVIDIA

Examples



Examples



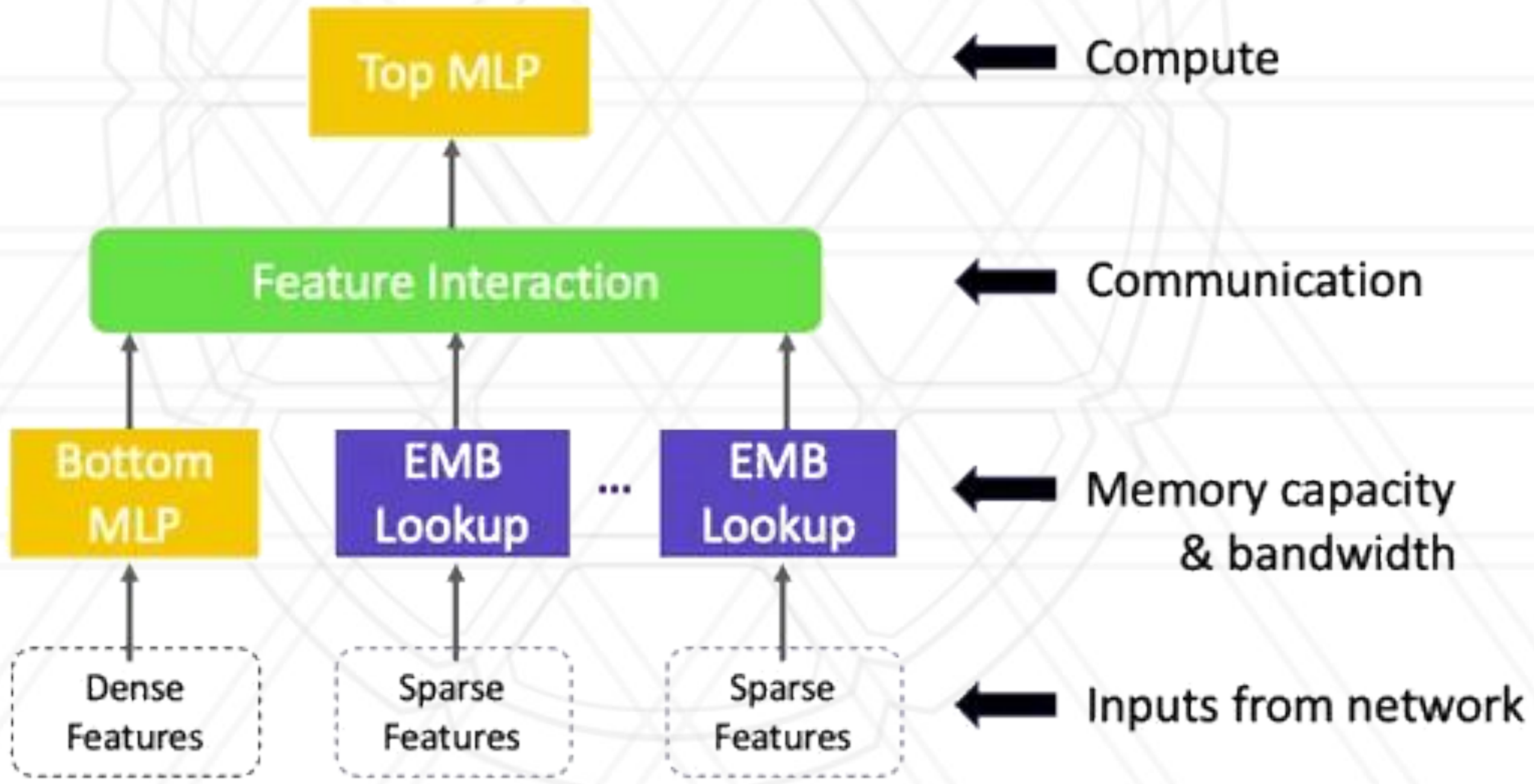

Neural Network Model (TPU Estimator)
TensorFlow Client
Cloud Engine VM
TensorFlow Service
XLA Ops
XLA
Just-in-time Compiler
Cloud TPU Server
TPU Binary (PCIe)
Cloud TPU

Goals of Co-Design

- Data movement and locality optimizations
- Specialized computation components
  - higher throughput, lower latency
- Reduced power consumption
- Software development ease
- Reduce costs

DLRMs Overview

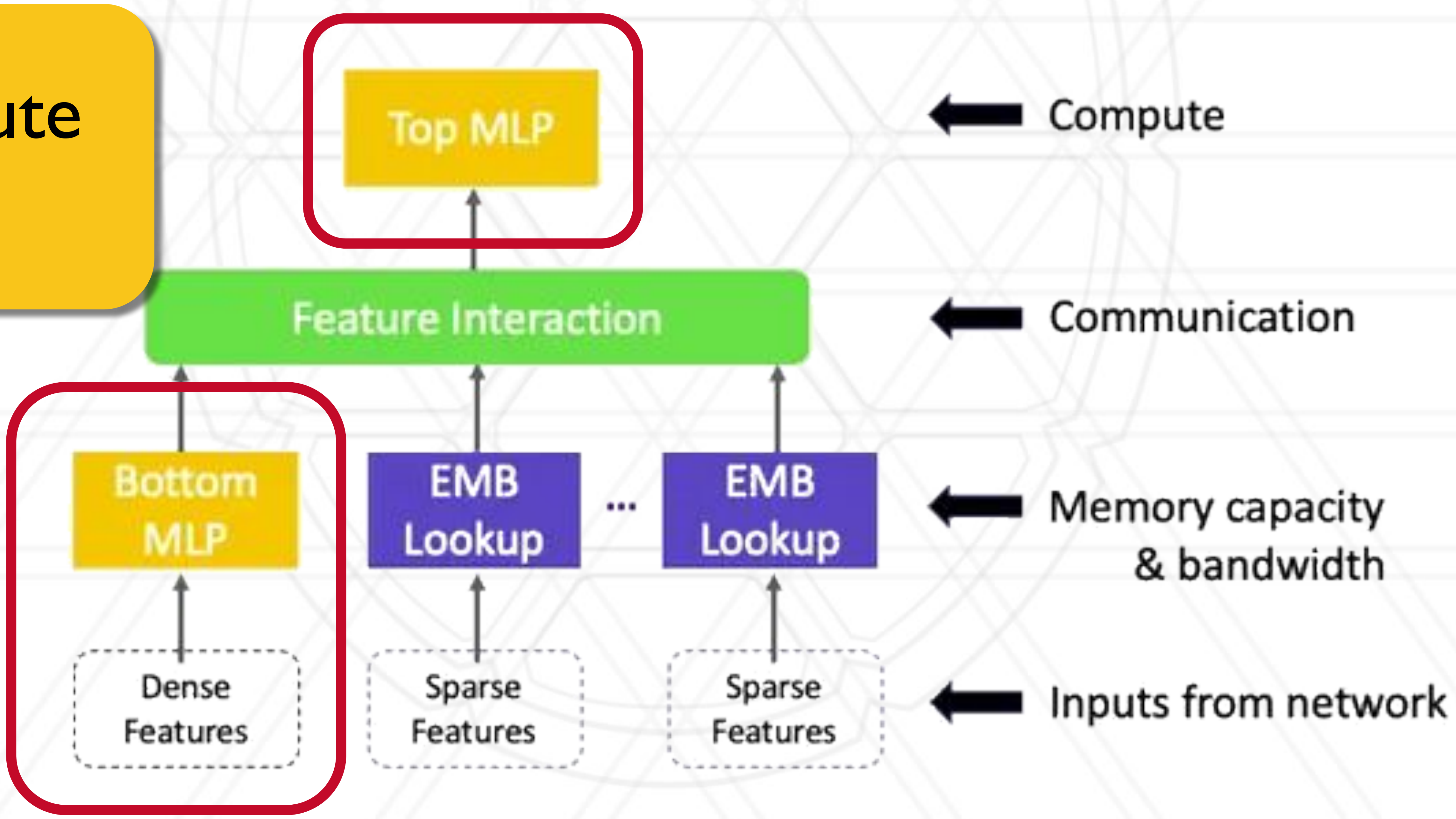
- Deep Learning Recommendation Models
- "Deep Learning Recommendation Model for Personalization and Recommendation Systems", M. Naumov et al
- Online and offline training



DLRMs Overview

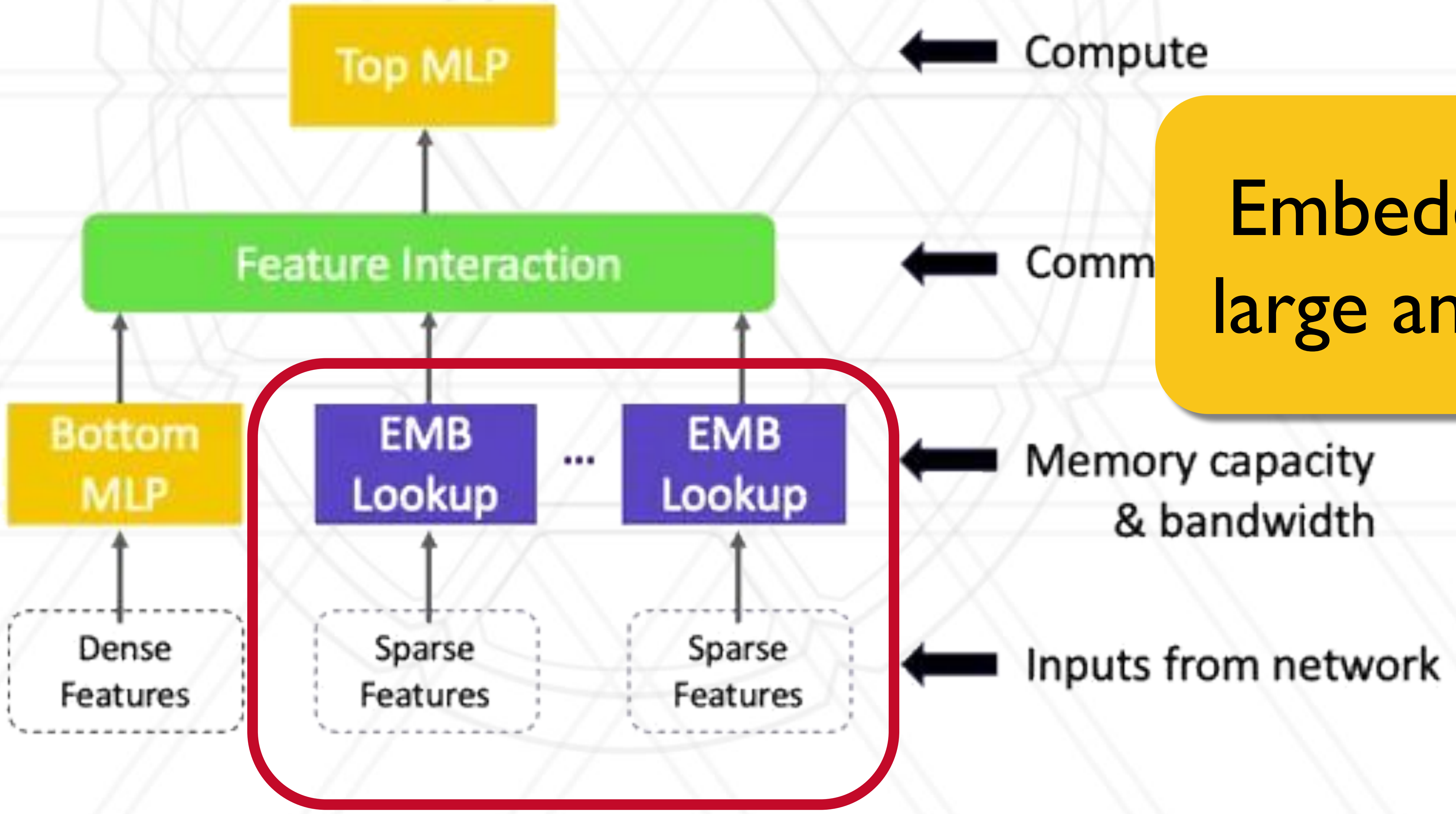
- Deep Learning Recommendation Models
- "Deep Learning Recommendation Model for Personalization and Recommendation Systems", M. Naumov et al
- Online and offline training

MLP layers are compute intensive



DLRMs Overview

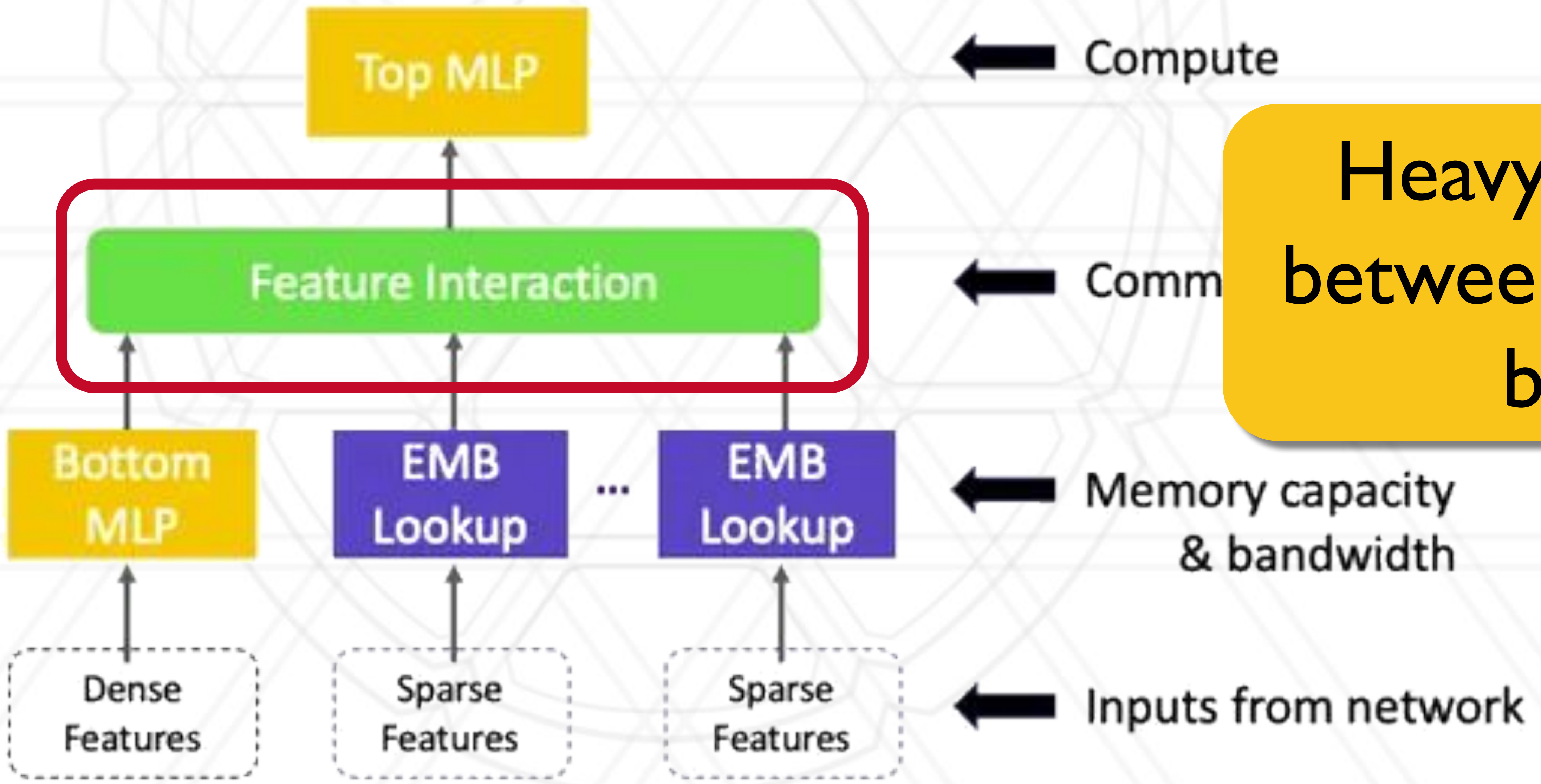
- Deep Learning Recommendation Models
- "Deep Learning Recommendation Model for Personalization and Recommendation Systems", M. Naumov et al
- Online and offline training



EmbeddingBags are very large and memory bound

DLRMs Overview

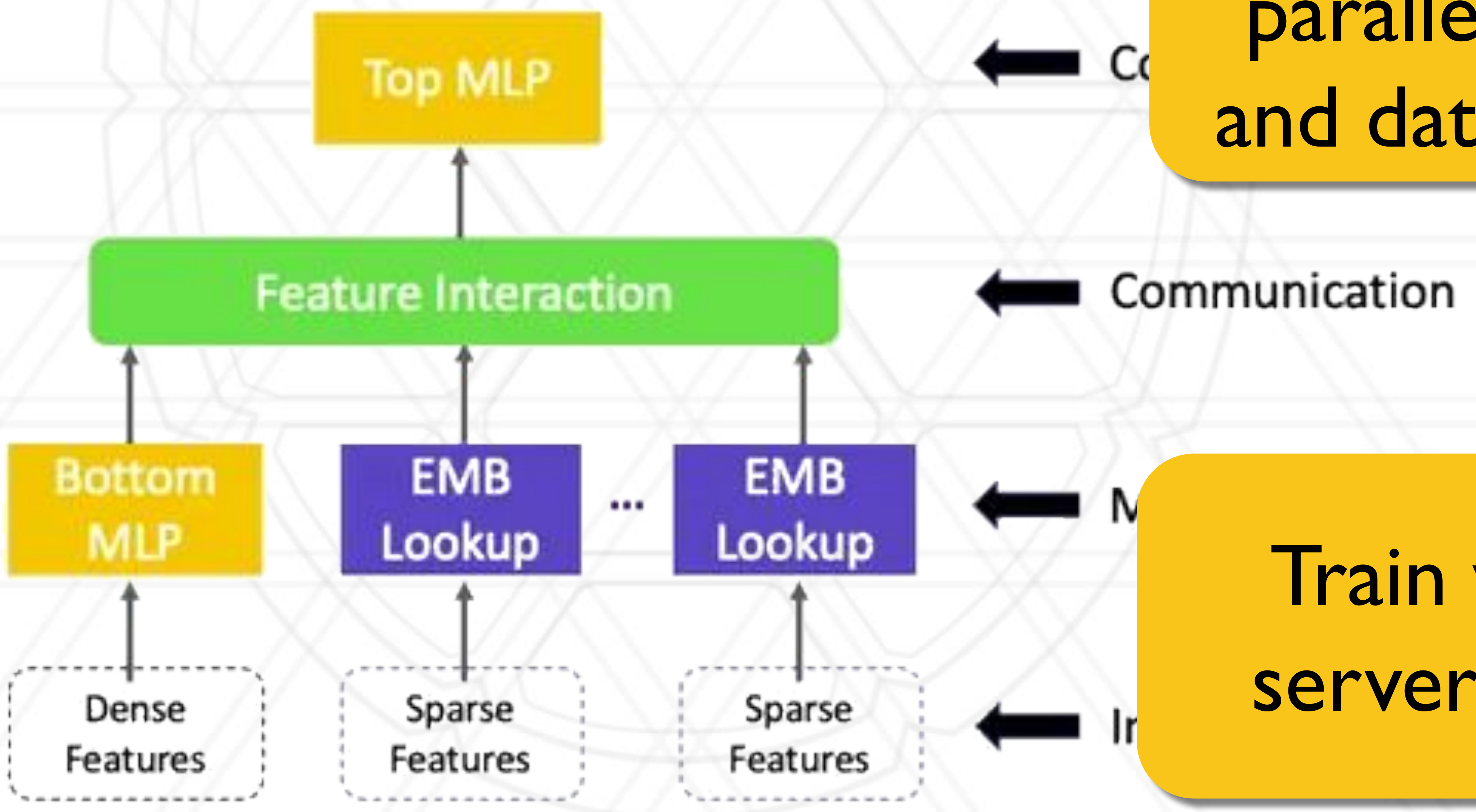
- Deep Learning Recommendation Models
- "Deep Learning Recommendation Model for Personalization and Recommendation Systems", M. Naumov et al
- Online and offline training



Heavy communication between embeddings and bottom MLP

DLRMs Overview

- Deep Learning Recommendation Models
- "Deep Learning Recommendation Model for Personalization and Recommendation Systems", M. Naumov et al
- Online and offline training

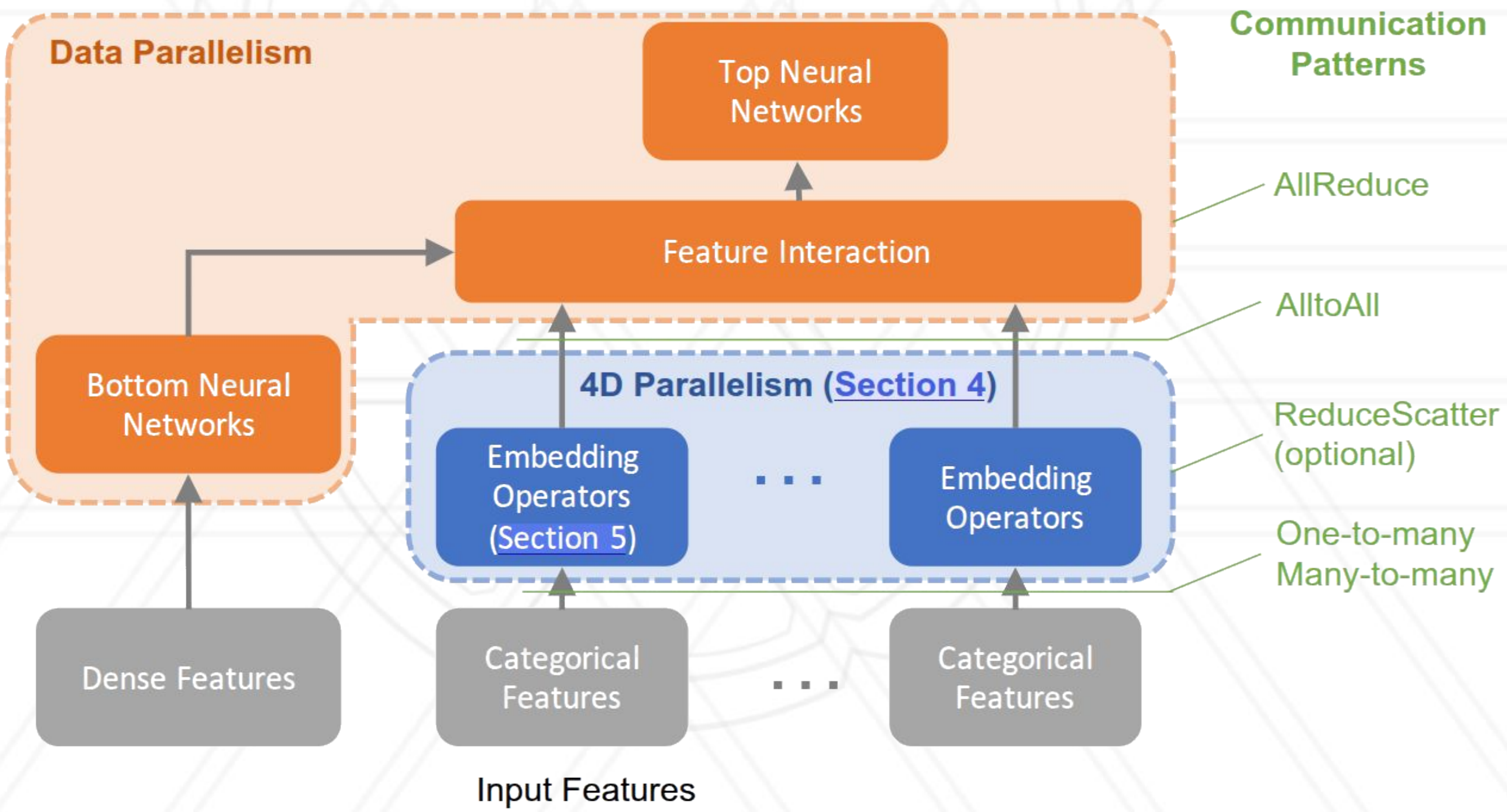


Traditionally use model parallel for embeddings and data parallel for MLP

Train with parameter server and async SGD

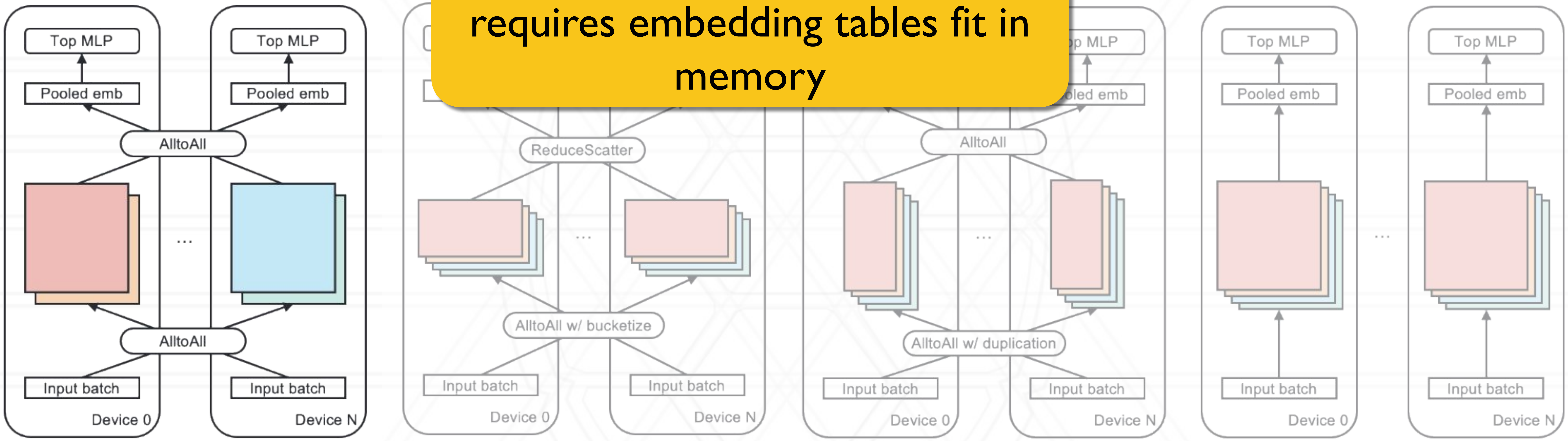
Neo Overview

- A DLRM training system
  - Neo software with 4D parallelism for embedding operators
  - Optimized sequential embedding implementations
  - ZionEX: a hardware system designed to efficiently run Neo



4D Embedding Parallelism

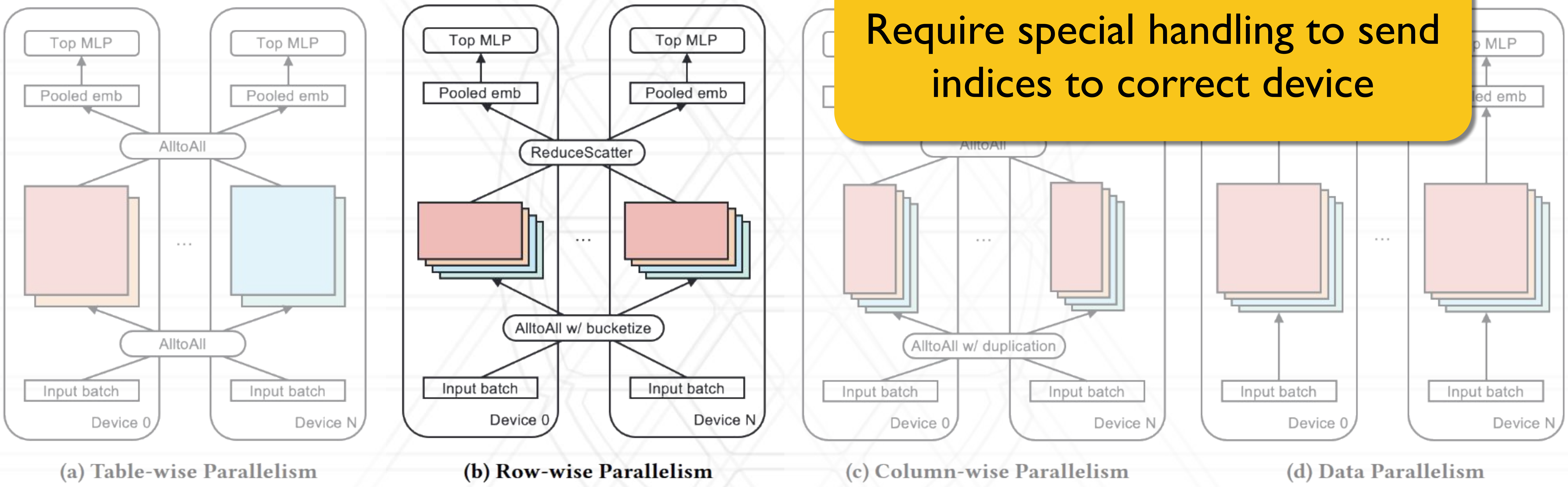
Most communication optimal, but requires embedding tables fit in memory



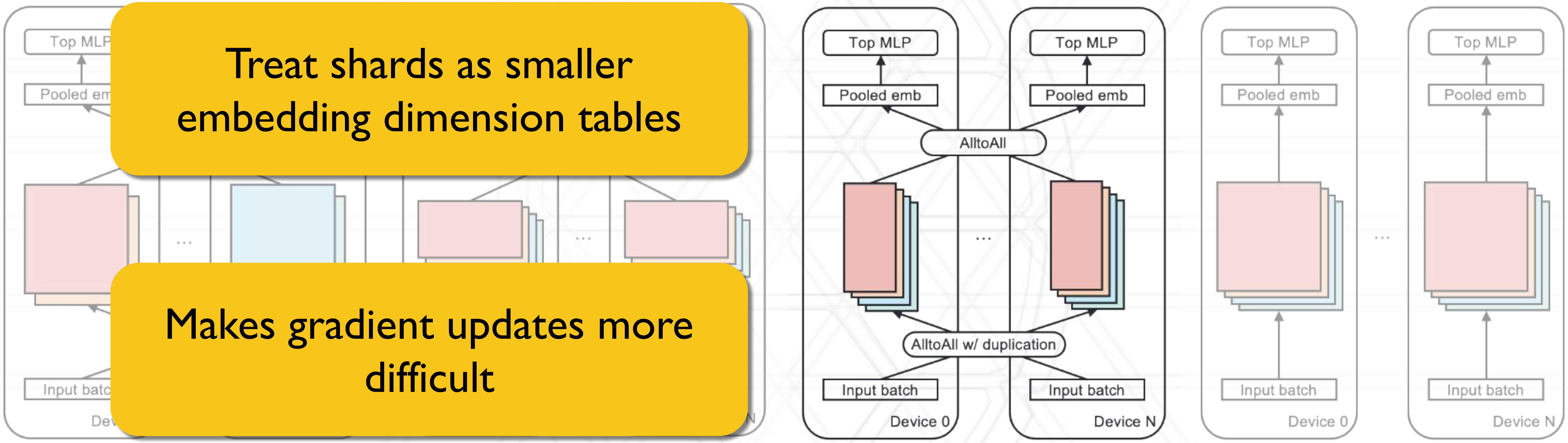
(a) Table-wise Parallelism (b) Row-wise Parallelism (c) Column-wise Parallelism (d) Data Parallelism

4D Embedding Parallelism

Require special handling to send indices to correct device



4D Embedding Parallelism

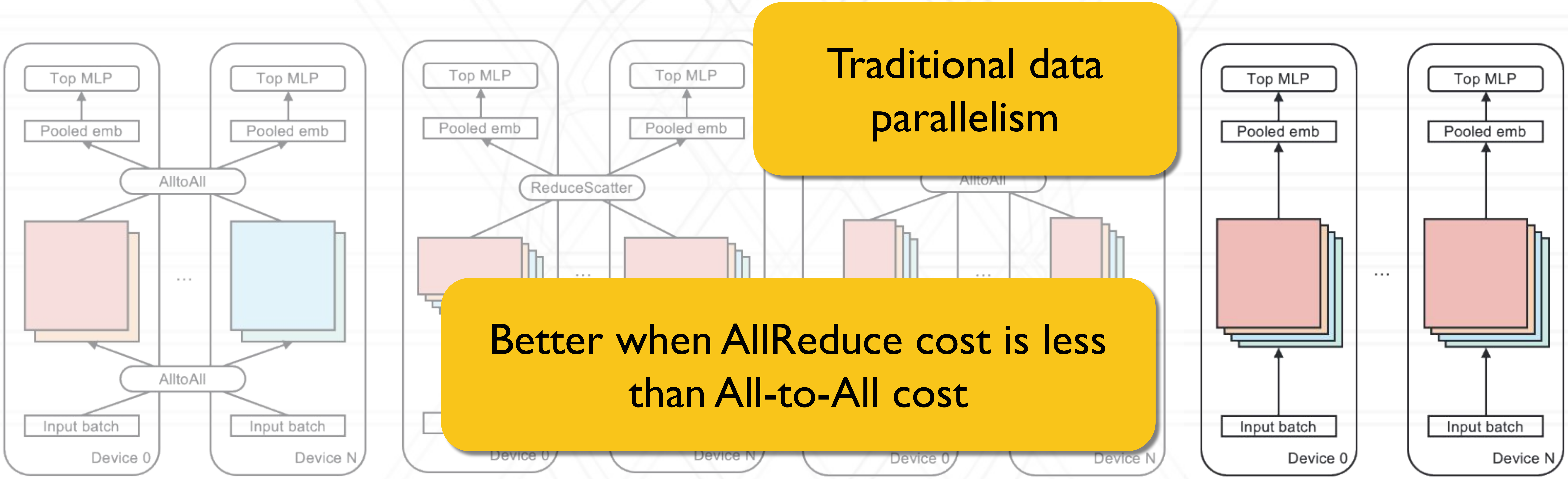


Treat shards as smaller embedding dimension tables

Makes gradient updates more difficult

(a) Table-wise Parallelism (b) Row-wise Parallelism **(c) Column-wise Parallelism** (d) Data Parallelism

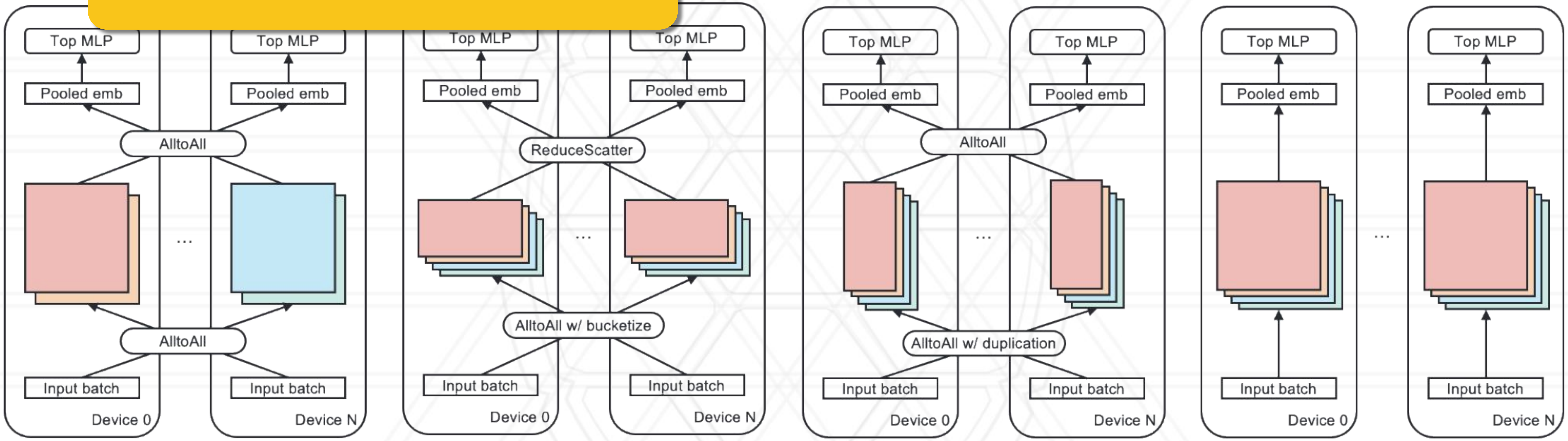
4D Embedding Parallelism



(a) Table-wise Parallelism (b) Row-wise Parallelism (c) Column-wise Parallelism **(d) Data Parallelism**

4D Embedding Parallelism

Use all 4 for ideal parallelism!



(a) Table-wise Parallelism

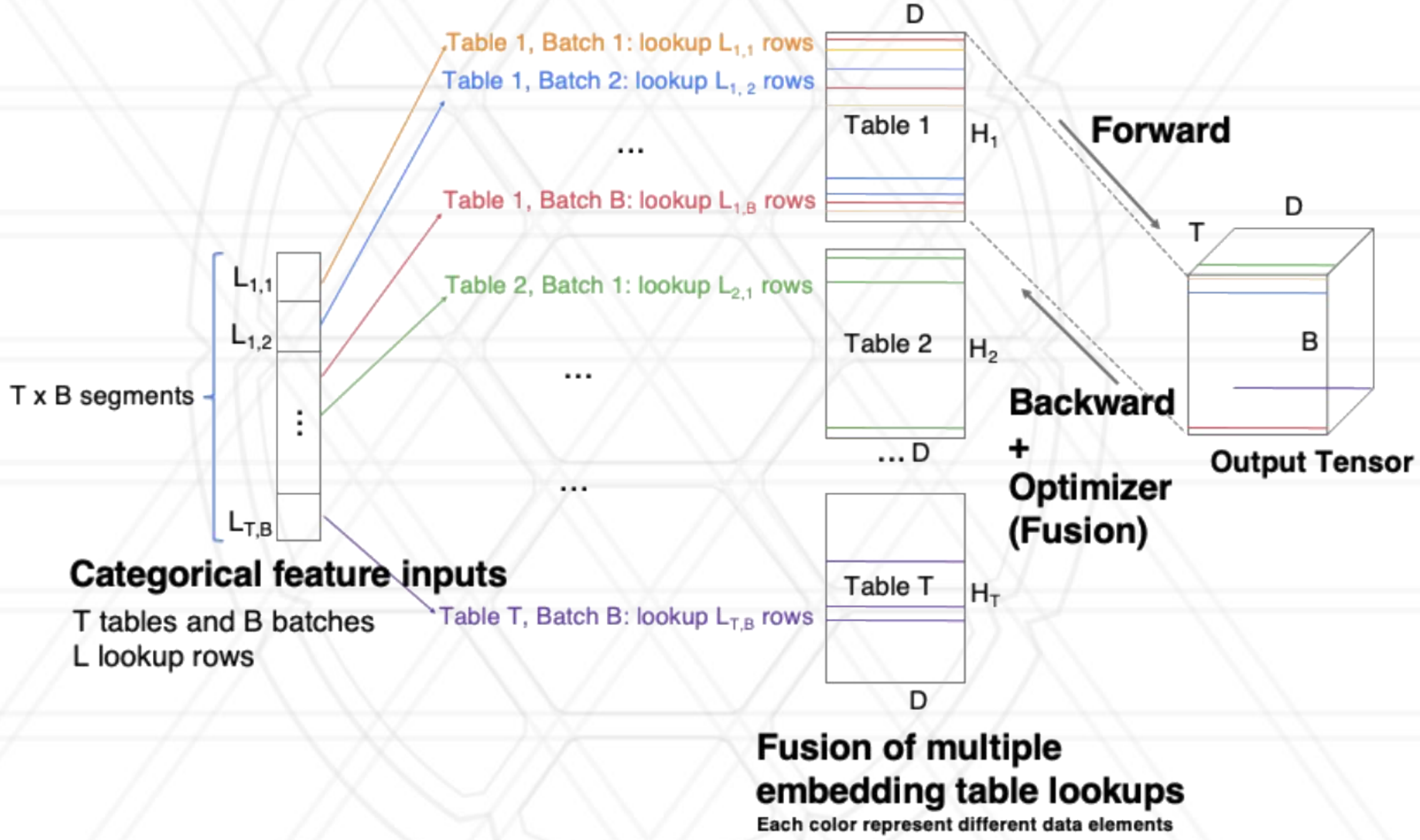
(b) Row-wise Parallelism

Use performance models and pipelining to find optimal configurations

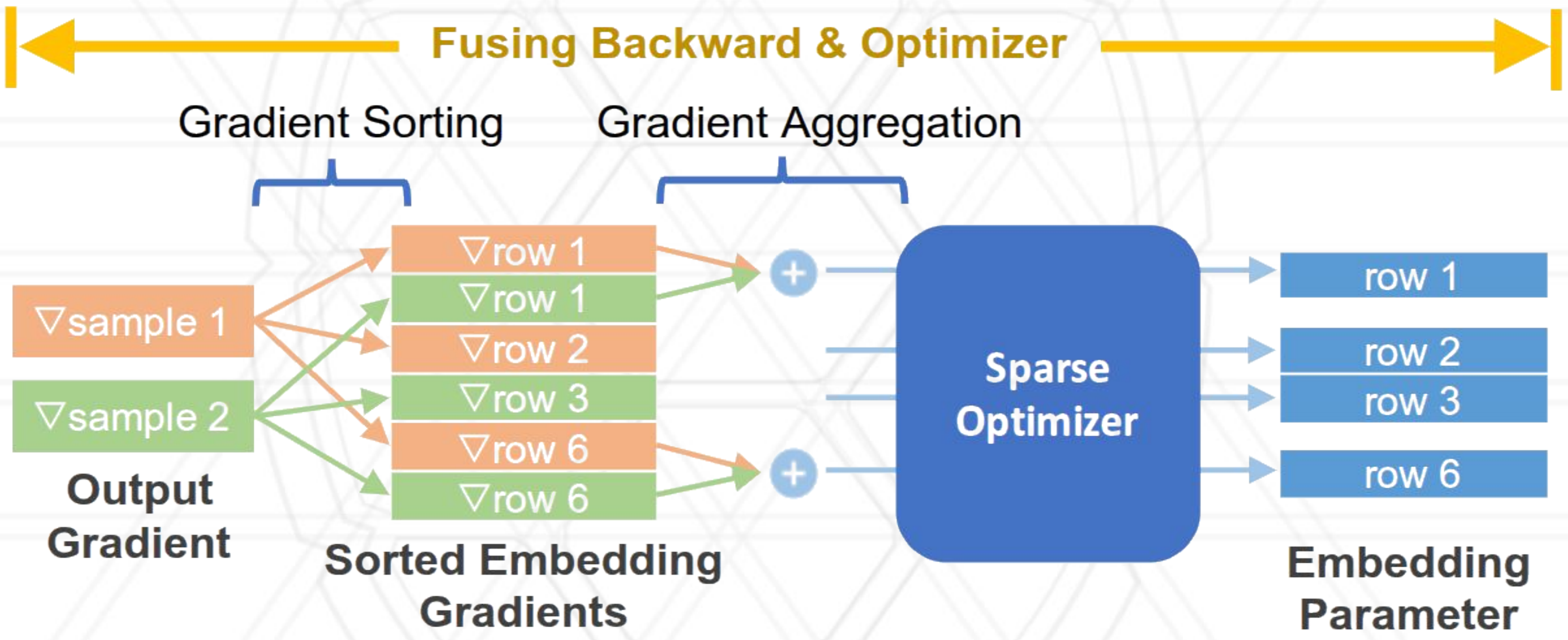
Optimizing the Embedding Operators

- Two key inefficiencies
  - Each lookup is a single GPU kernel; incurs high overheads
  - Large tables need multi-GPU implementations
- Operator optimizations
  - Fused embeddings into single kernels; sort embedding gradients
  - Multi-GPU implementations
  - Co-Design with ZionEX to save memory

Fused Embedding Operators



Sort Gradients of Embedding Operators



Memory Optimizations

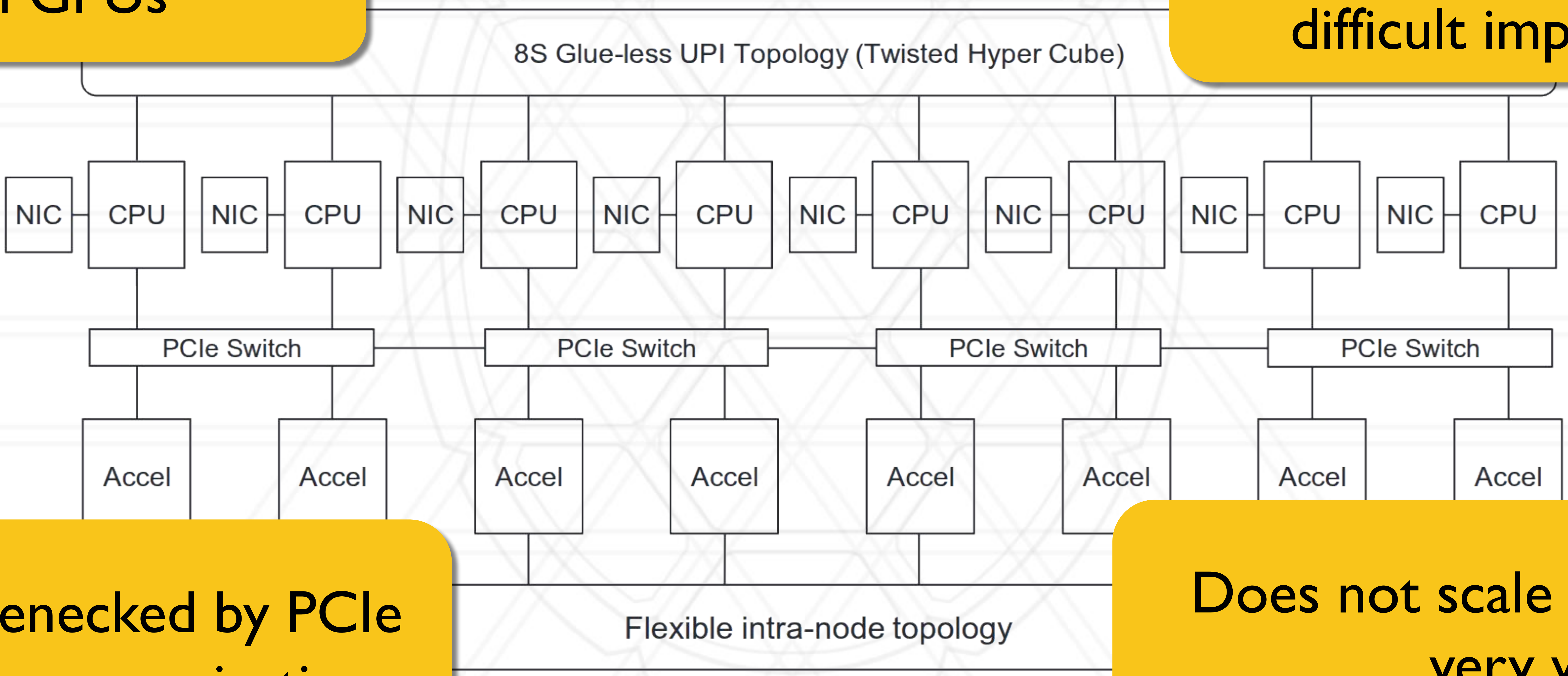
- Make use of device memory, host memory, and disk
- Access behavior is irregular
  - Default managed memory (like CUDA unified) experiences very poor performance
  - Implement custom cache in software
- Embedding compression
  - low precision (high precision cache and low precision embeddings)
  - sparse optimizers

ZionEX: Co-Designing with Neo

- Previous system, Zion, had many limitations

Embedding on CPUs and MLP on GPUs

Balancing work requires careful design considerations and difficult implementation



GPUs are bottlenecked by PCIe for internode communication

Does not scale to multi-node very well!

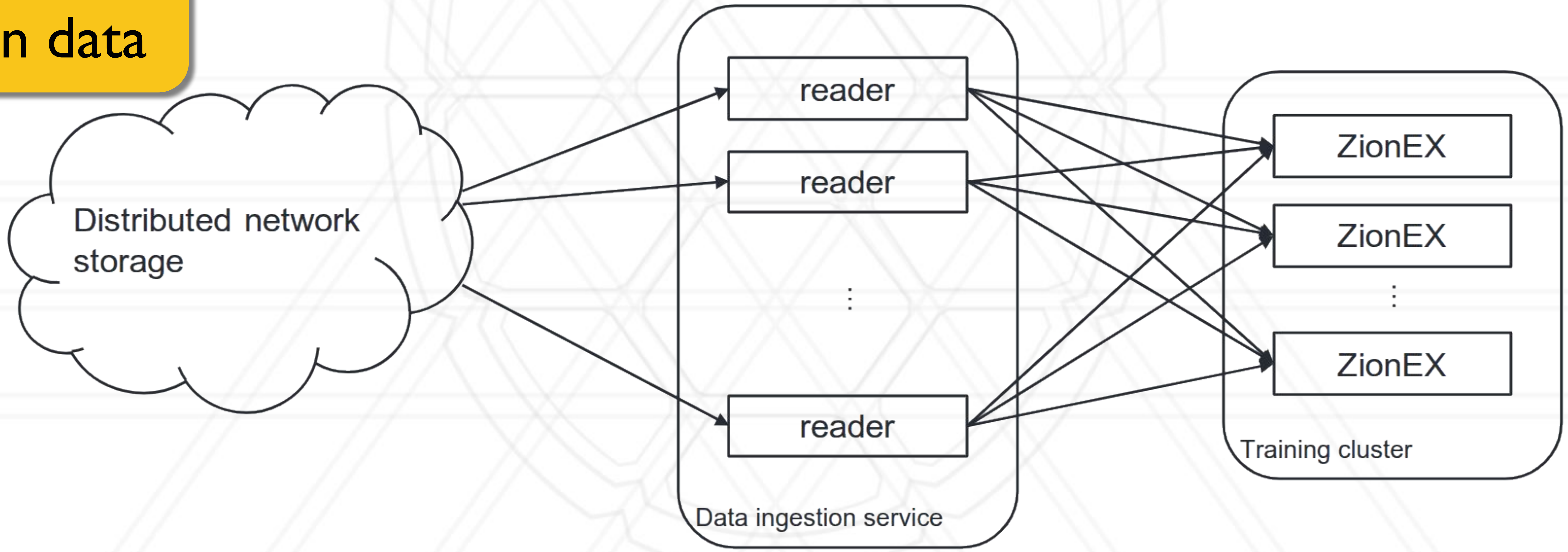
ZionEX: Co-Designing with Neo

- ZionEx addresses these shortcomings
- Designed to support 4D parallelism and data requirements of DLRM

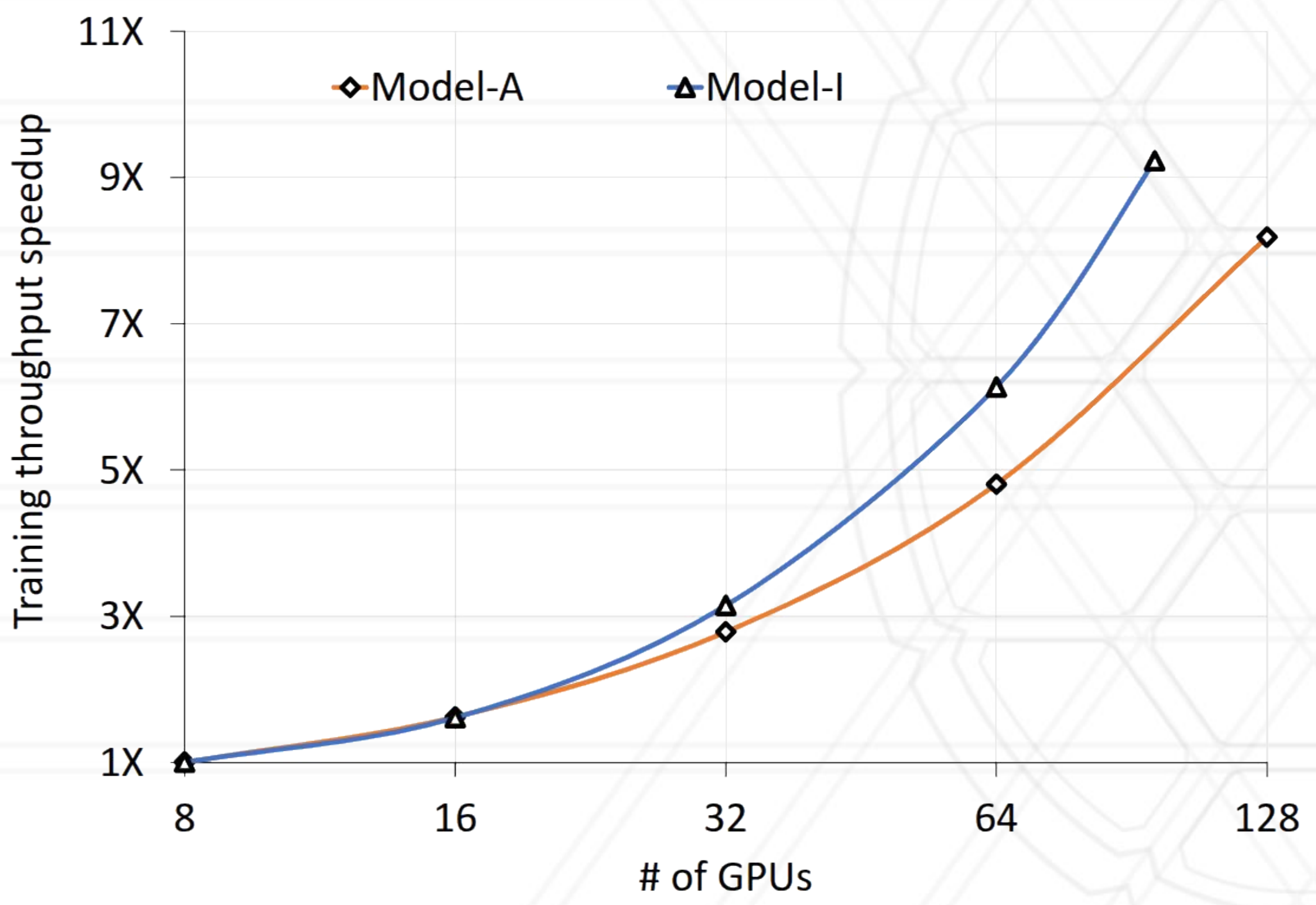
ZionEX: Co-Designing with Neo

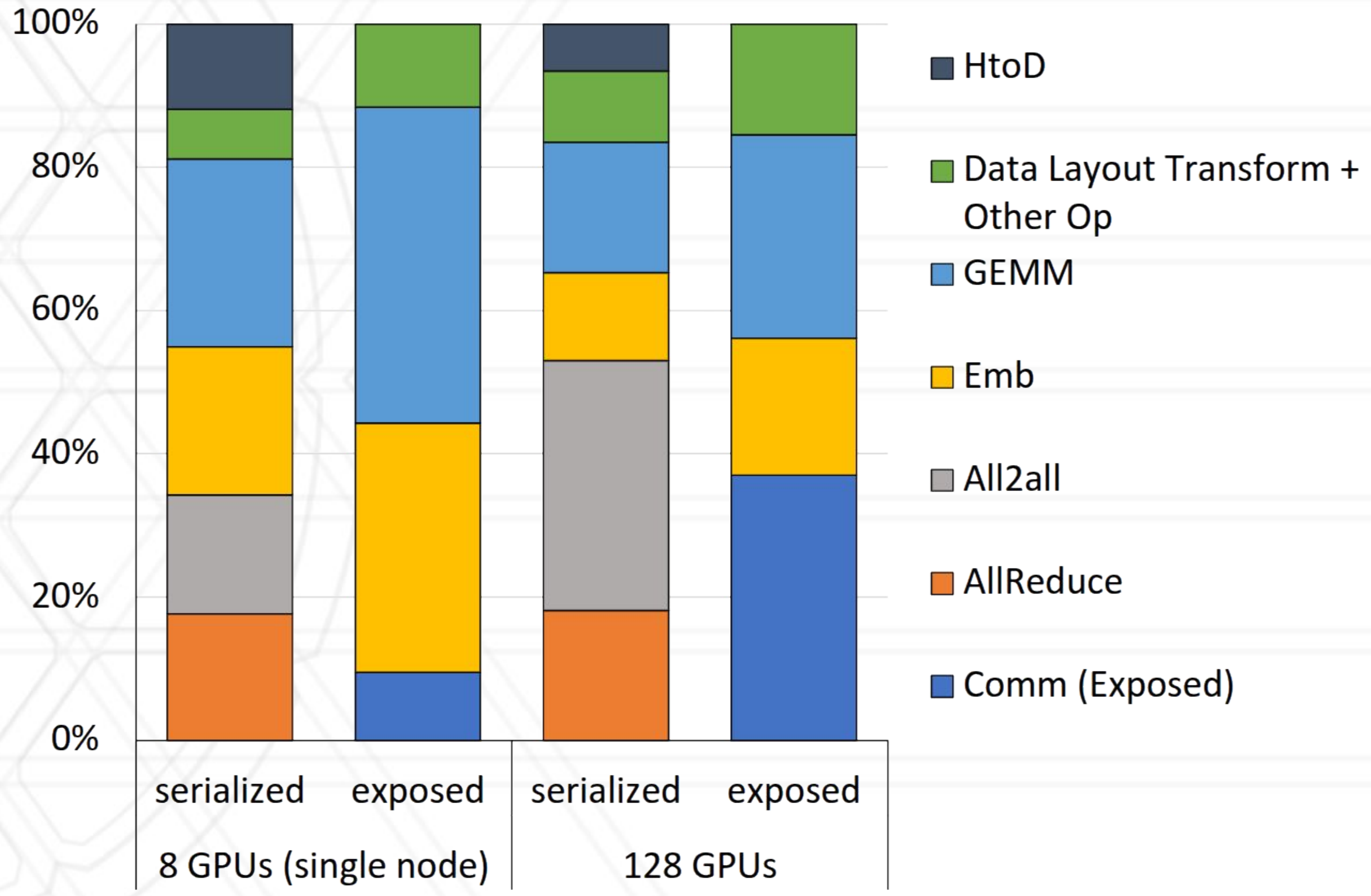
- ZionEx addresses these shortcomings
- Designed to support 4D parallelism and data requirements of DLRM
- Custom data ingestion servers to overcome latencies

Prevent ZionEx nodes from waiting on data

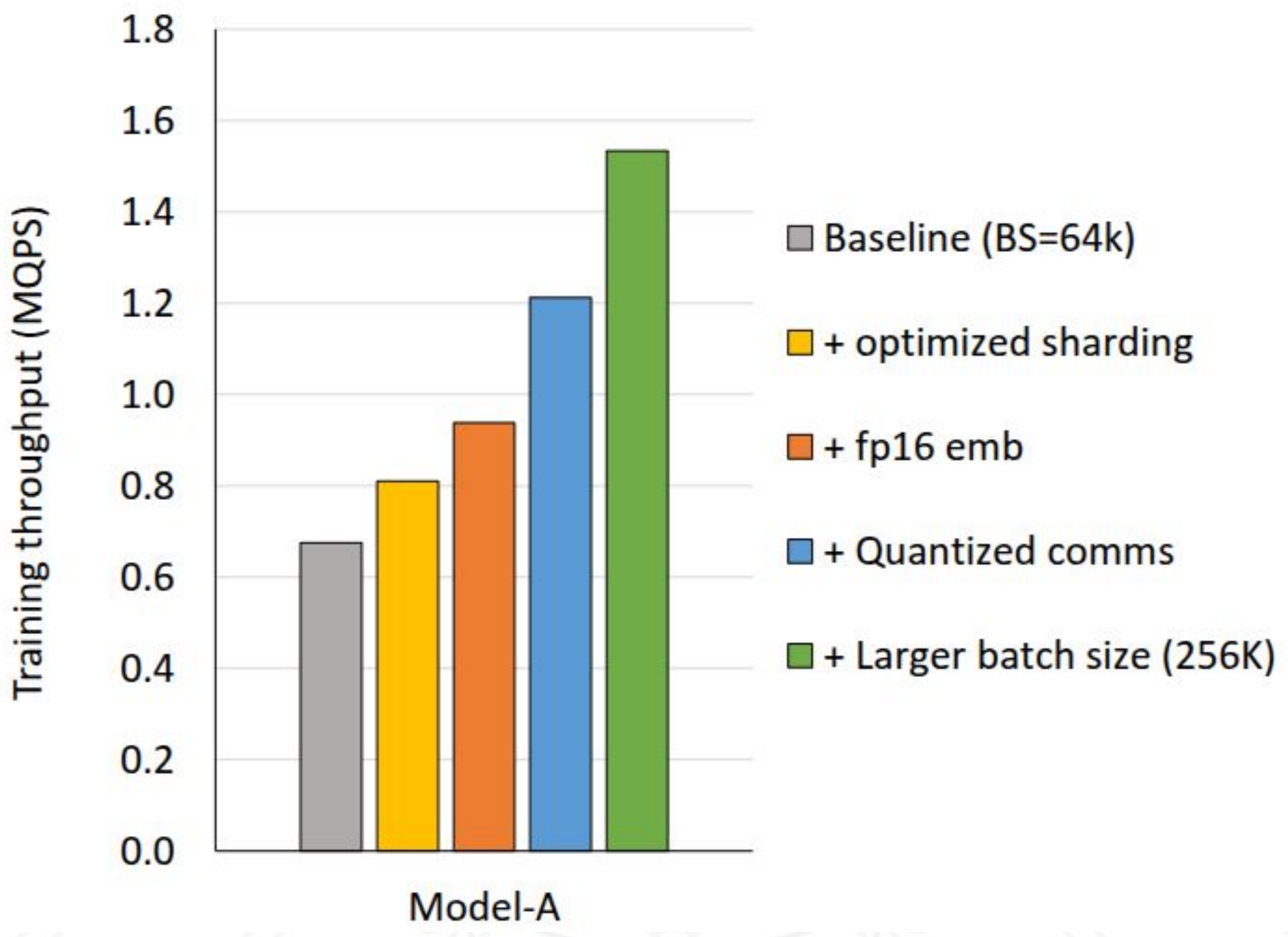


Training Performance



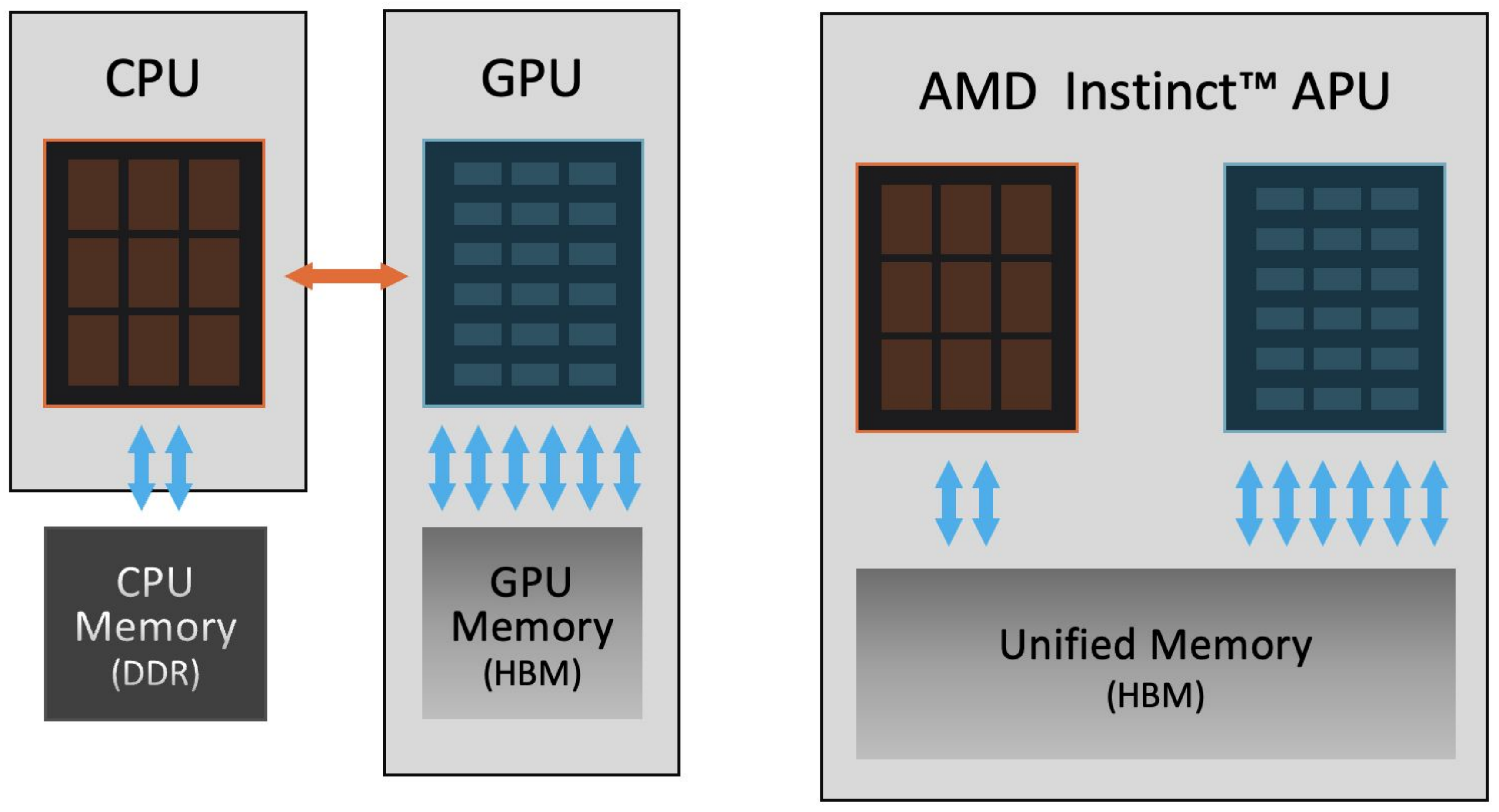


Other Training Optimizations



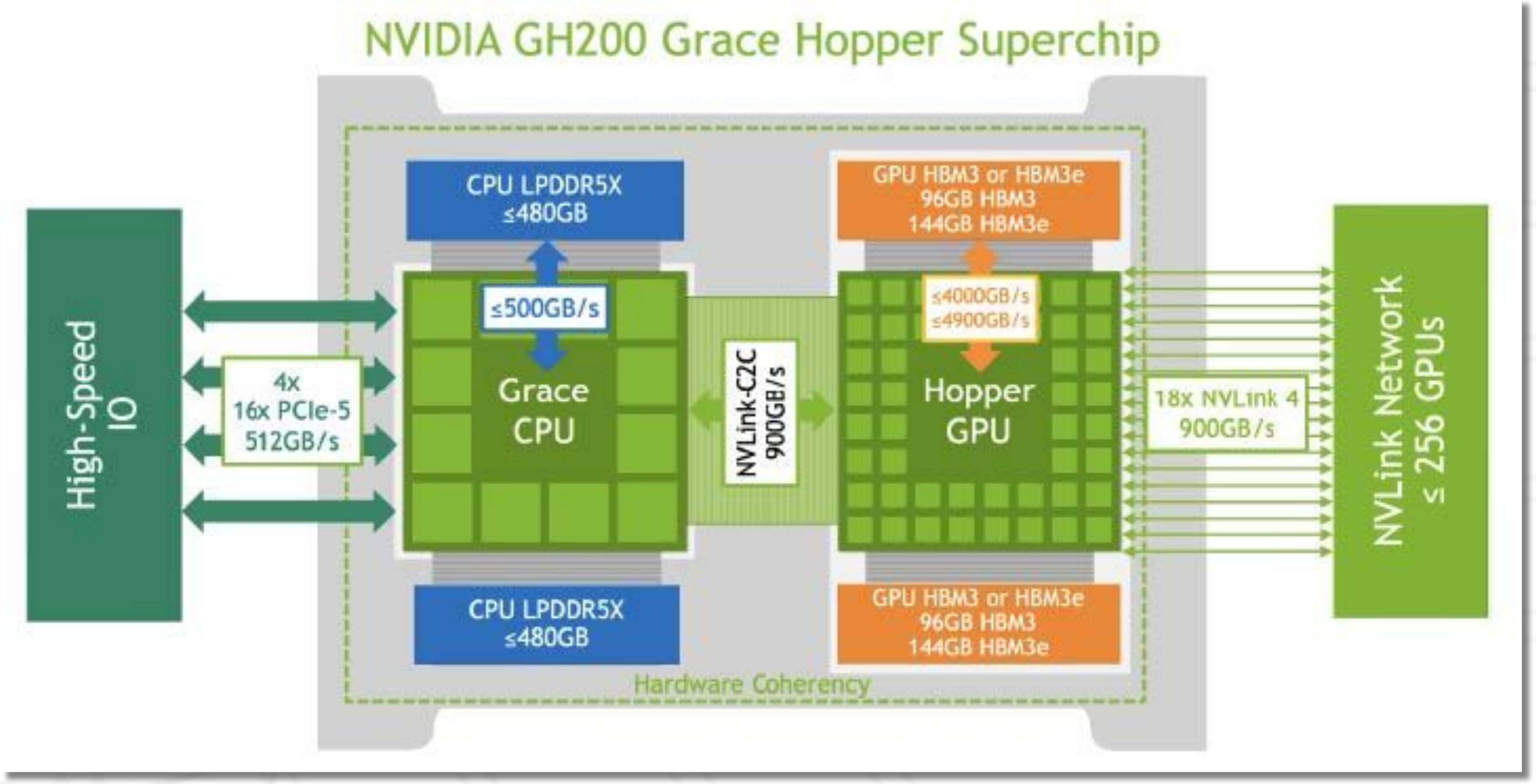
Shared Memory Nodes

MI300A and GH200 are combined CPU-GPU nodes produced by AMD and NVIDIA



(a) Discrete CPU and GPU.

(b) APU.



UNIVERSITY OF
MARYLAND