Massive-Scale Neural Network Training

Weijie Zhao

11/08/2022

Sponsored Online Advertising

Query palo alto hotel

Ad 

User portrait 

… …

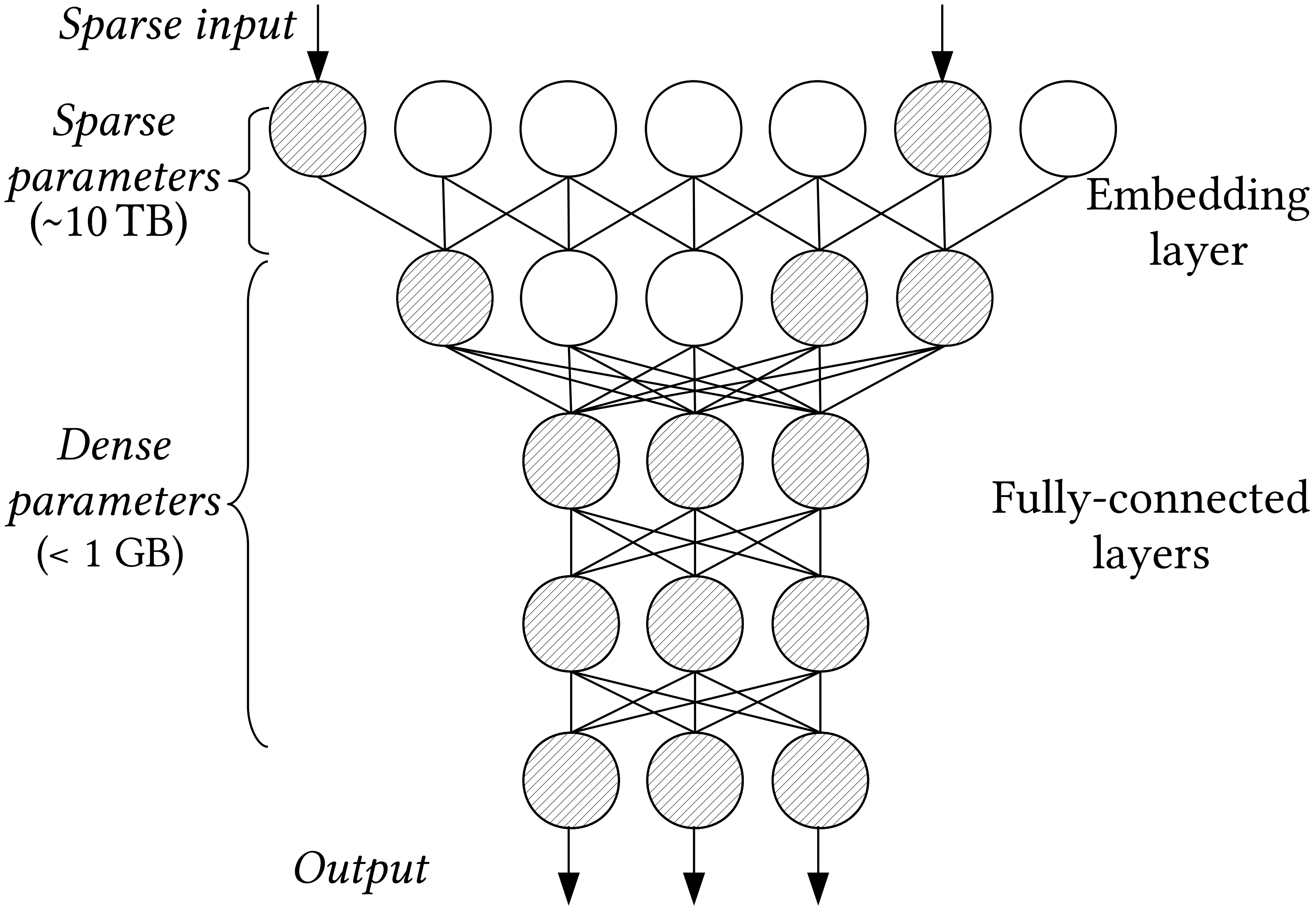
Neural Network

Click-through Rate

$$\textbf{CTR} = \frac{\#Click\text{−}throughs}{\#Impressions} \times 100\%$$

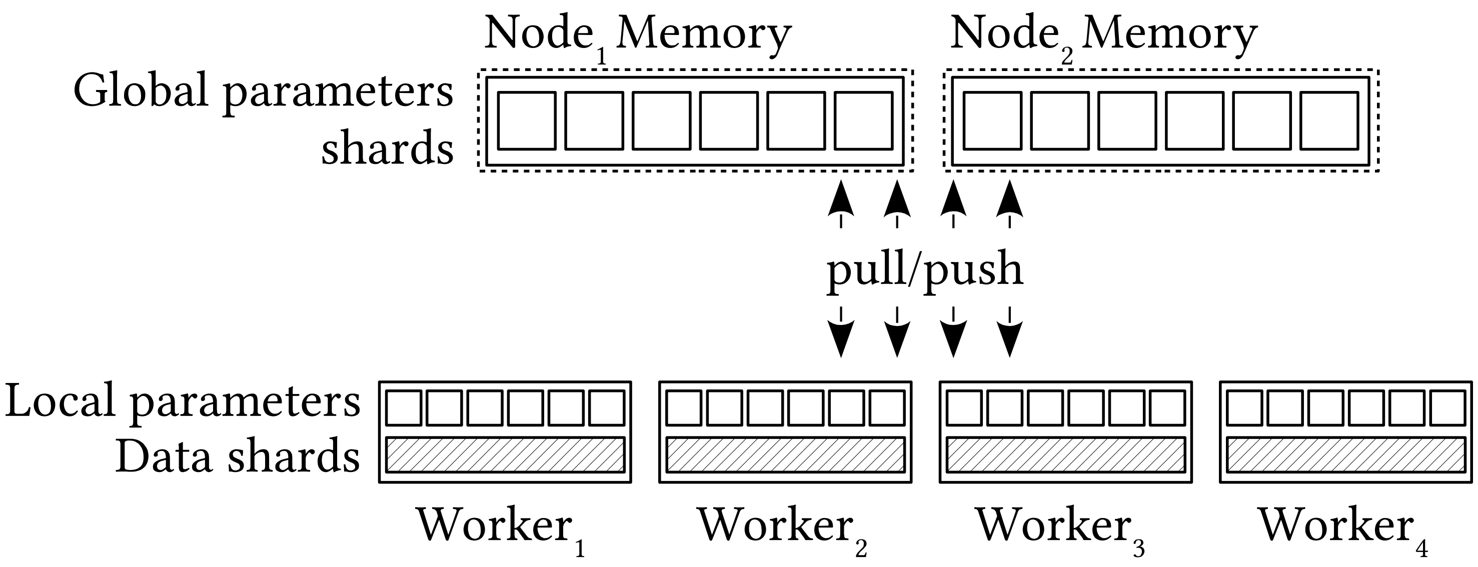
High-dimensional sparse vectors (10^{11} dimensions)

A Visual Illustration of CTR Models



MPI Cluster Solution

Distributed Parameter Server



Wait! Why do We Need Such a Massive Model?

Hashing For Reducing CTR Models

One permutation + one sign random projection (work done in 2015)

Image search ads is typically a small source of revenue

Table 1. OP+OSRP for Image Search Sponsored Ads Data

	# Nonzero Weights	Test AUC
Baseline LR	31,949,213,205	0.7112
Baseline DNN		0.7470
Hash+DNN ($k = 2^{34}$)	6,439,972,994	0.7407
Hash+DNN ($k = 2^{23}$)	3,903,844,565	0.7388
Hash+DNN ($k = 2^{22}$)	2,275,442,496	0.7370
Hash+DNN ($k = 2^{31}$)	1,284,025,453	0.7339
Hash+DNN ($k = 2^{30}$)	707,983,366	0.7310
Hash+DNN ($k = 2^{29}$)	383,499,175	0.7278
Hash+DNN ($k = 2^{28}$)	203,864,439	0.7245
Hash+DNN ($k = 2^{27}$)	106,824,123	0.7208
Hash+DNN ($k = 2^{26}$)	55,363,771	0.7175
Hash+DNN ($k = 2^{25}$)	28,479,330	0.7132
Hash+DNN ($k = 2^{24}$)	14,617,324	0.7113

1. Hashing + DNN significantly improves over LR (logistic regression)!
2. A fine solution if the goal is to use **single-machine** to achieve good accuracy!

Hashing For Reducing CTR Models

One permutation + one sign random projection (work done in 2015)

Web search ads
use more features
and larger models

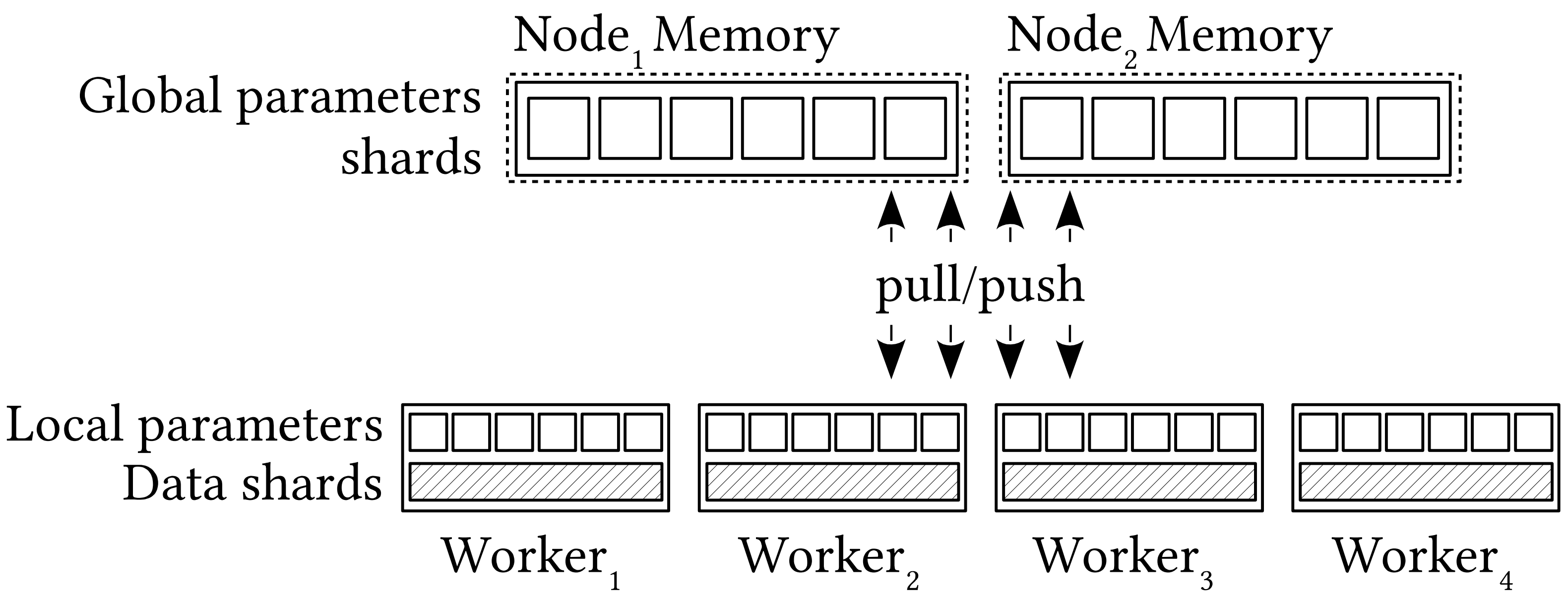
Table 2. OP+OSRP for Web Search Sponsored Ads Data

	# Nonzero Weights	Test AUC
Baseline LR	199,359,034,971	0.7458
Baseline DNN		0.7670
Hash+DNN ($k = 2^{32}$)	3,005,012,154	0.7556
Hash+DNN ($k = 2^{31}$)	1,599,247,184	0.7547
Hash+DNN ($k = 2^{30}$)	838,120,432	0.7538
Hash+DNN ($k = 2^{29}$)	433,267,303	0.7528
Hash+DNN ($k = 2^{28}$)	222,780,993	0.7515
Hash+DNN ($k = 2^{27}$)	114,222,607	0.7501
Hash+DNN ($k = 2^{26}$)	58,517,936	0.7487
Hash+DNN ($k = 2^{24}$)	15,410,799	0.7453
Hash+DNN ($k = 2^{22}$)	4,125,016	0.7408

1. Even a 0.1% decrease in AUC would result in a noticeable decrease in revenue
2. Solution of using hashing + DNN + single machine is typically not acceptable

MPI Cluster Solution

Distributed Parameter Server

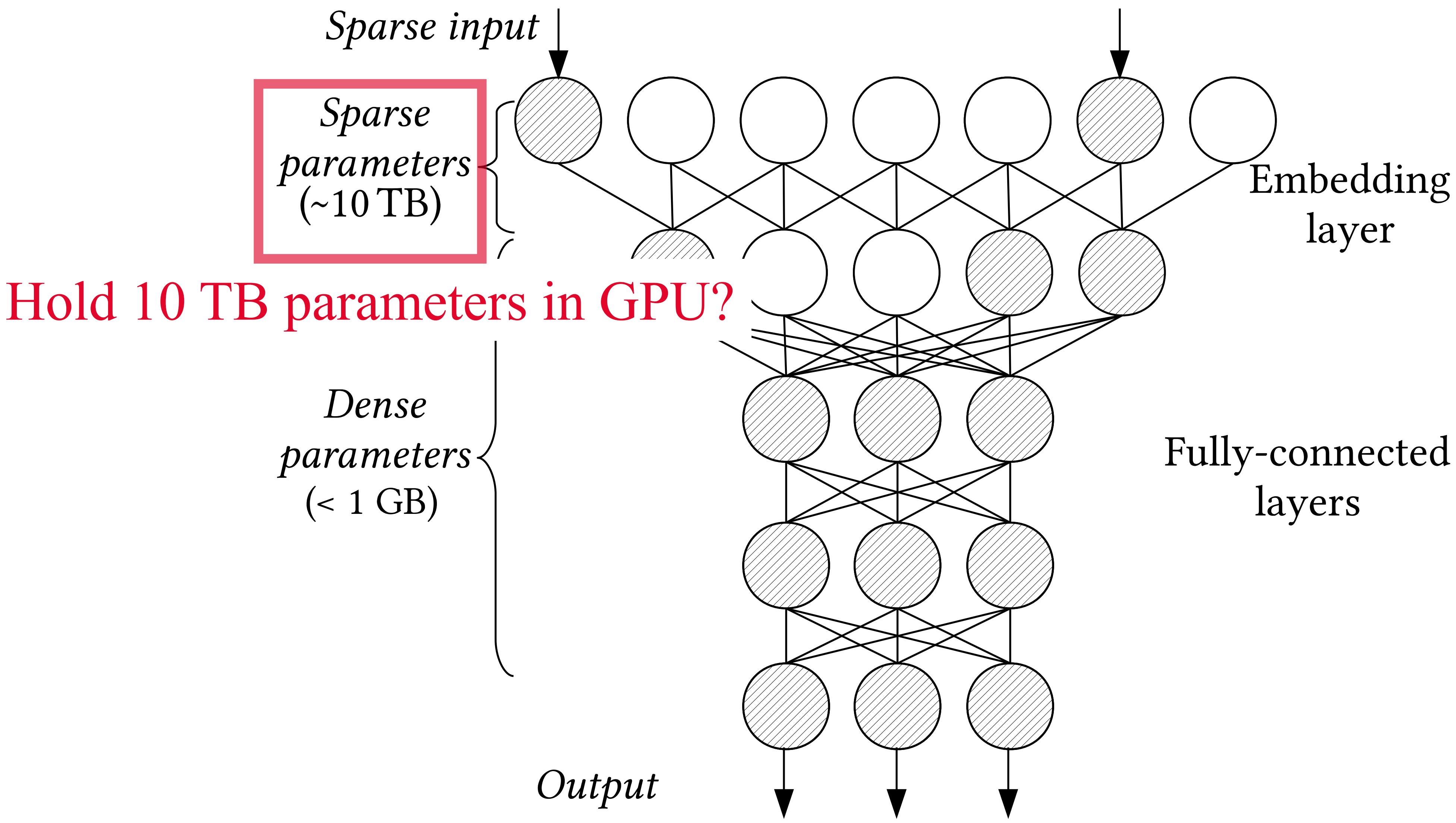


10-TB model parameters → Hundreds of computing nodes

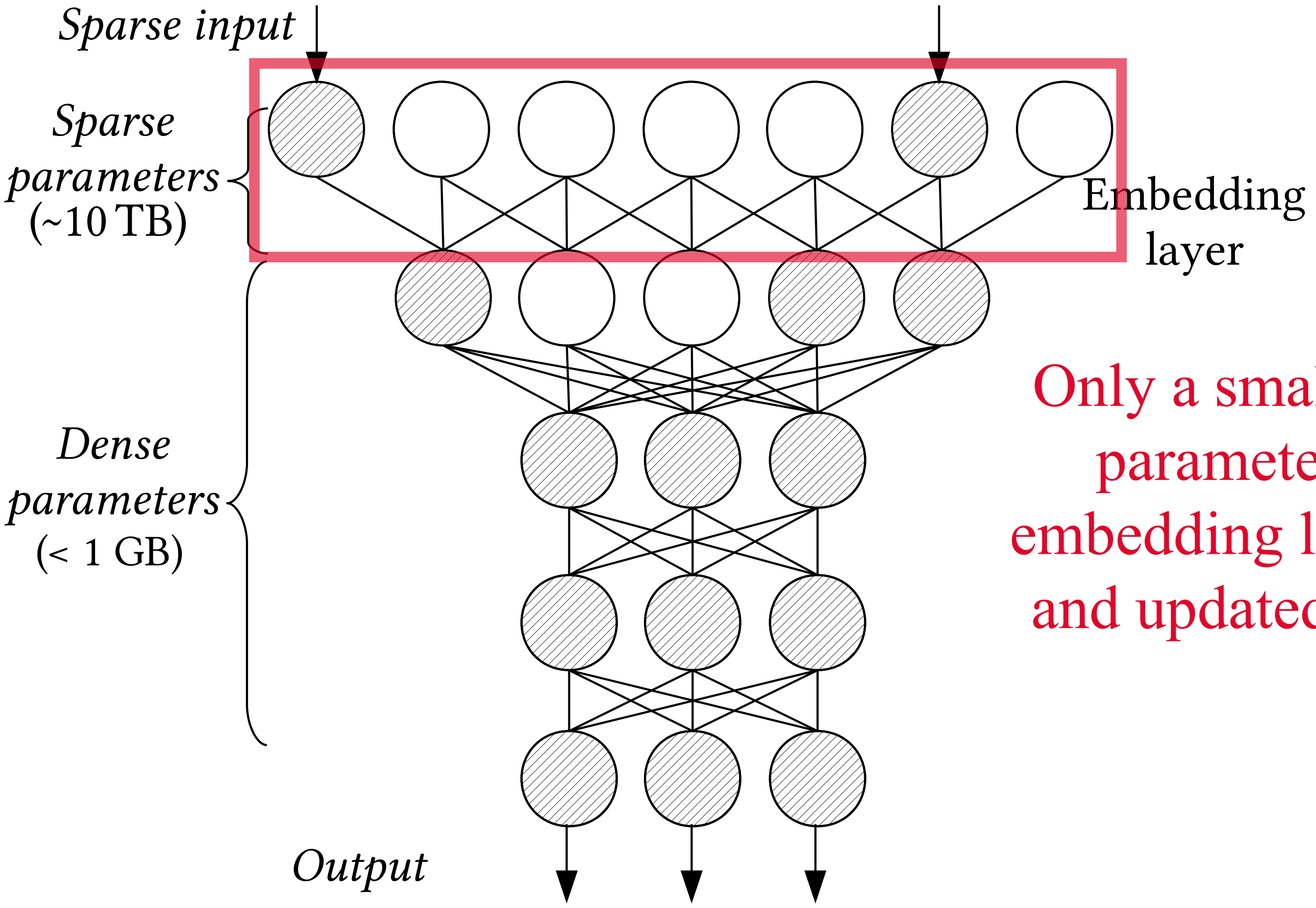
- Hardware and maintenance cost
- Communication cost

But all the cool kids use GPUs!

Let’s train the 10-TB Model with GPUs!

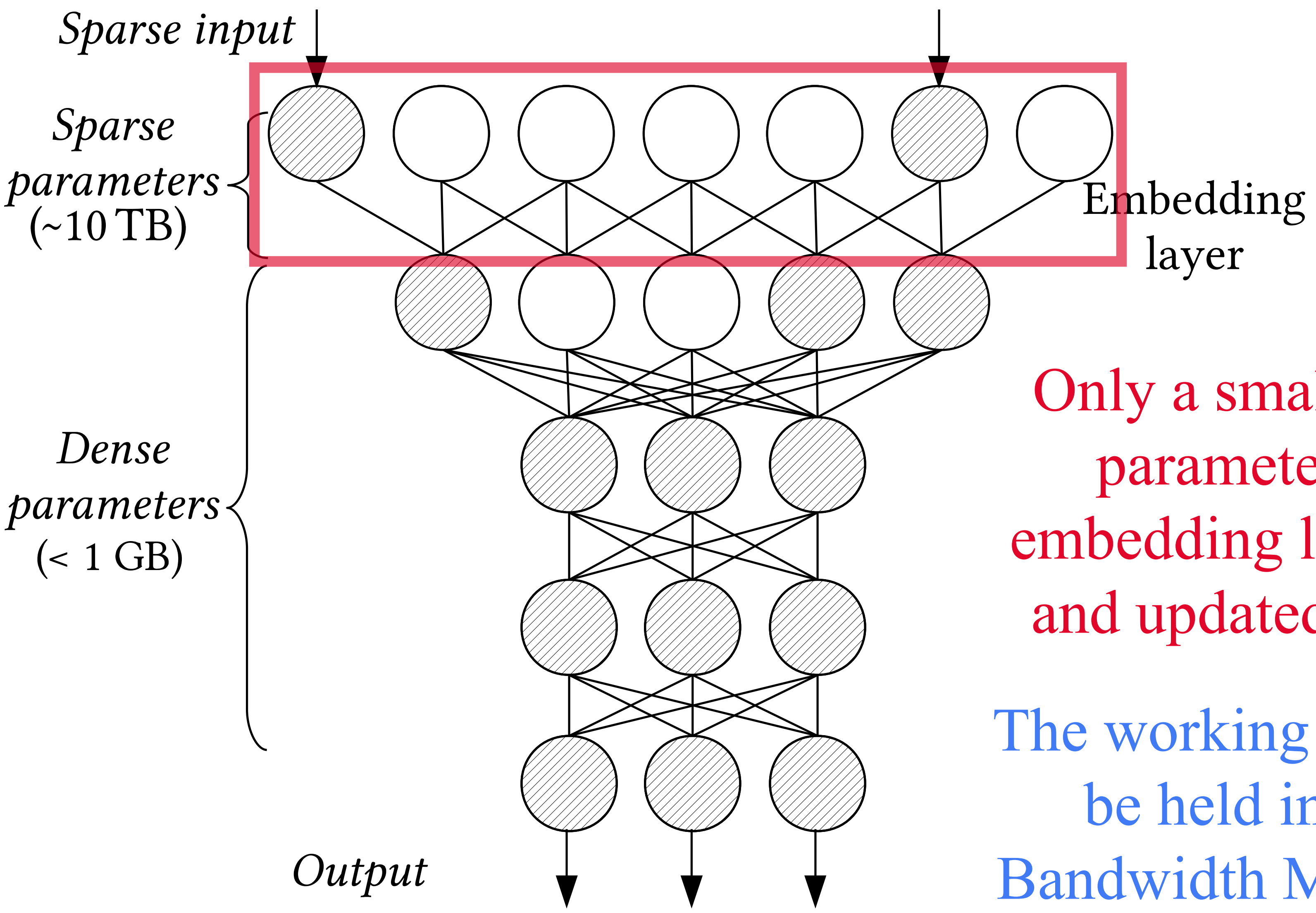

Sparse input
Sparse parameters (~10 TB)
Hold 10 TB parameters in GPU?
Embedding layer
Dense parameters (< 1 GB)
Fully-connected layers
Output

A few hundreds of non-zeros



Only a small subset of parameters in the embedding layer is used and updated in a batch

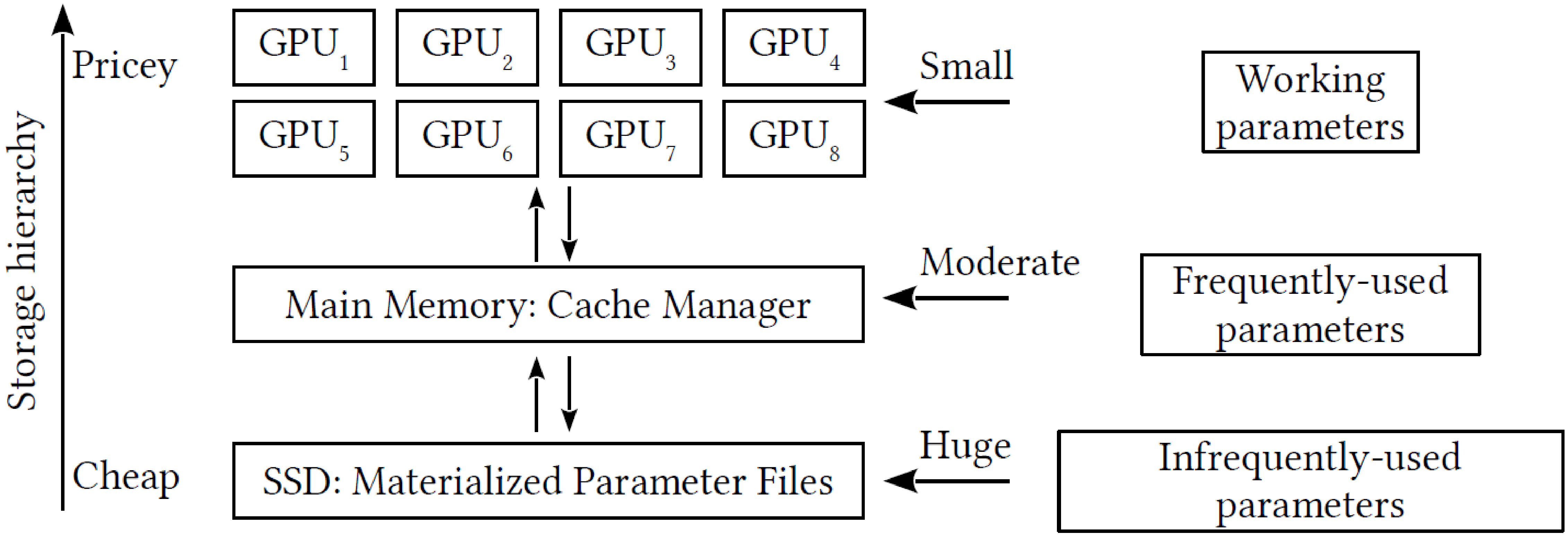
A few hundreds of non-zeros



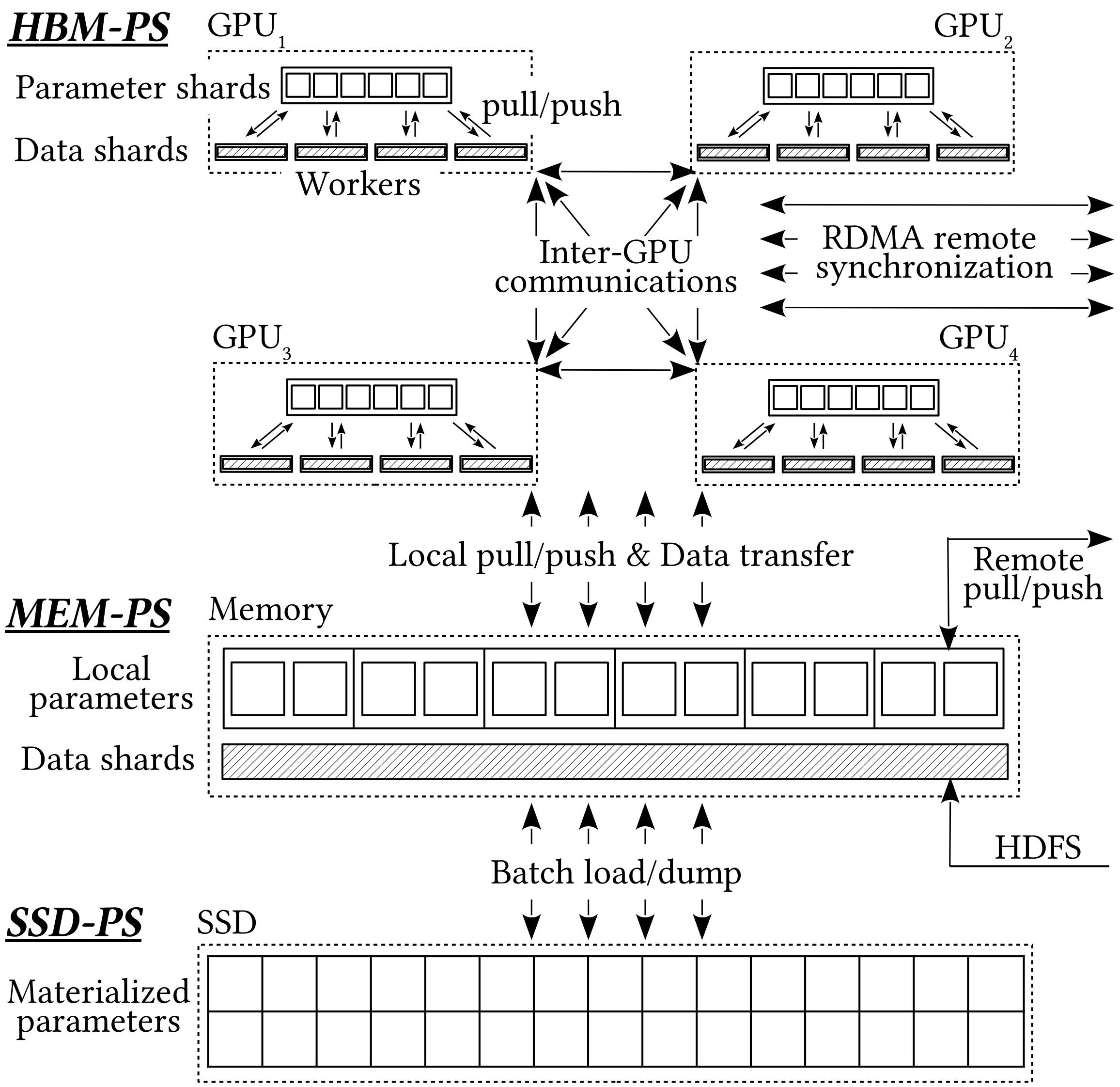
Only a small subset of parameters in the embedding layer is used and updated in a batch

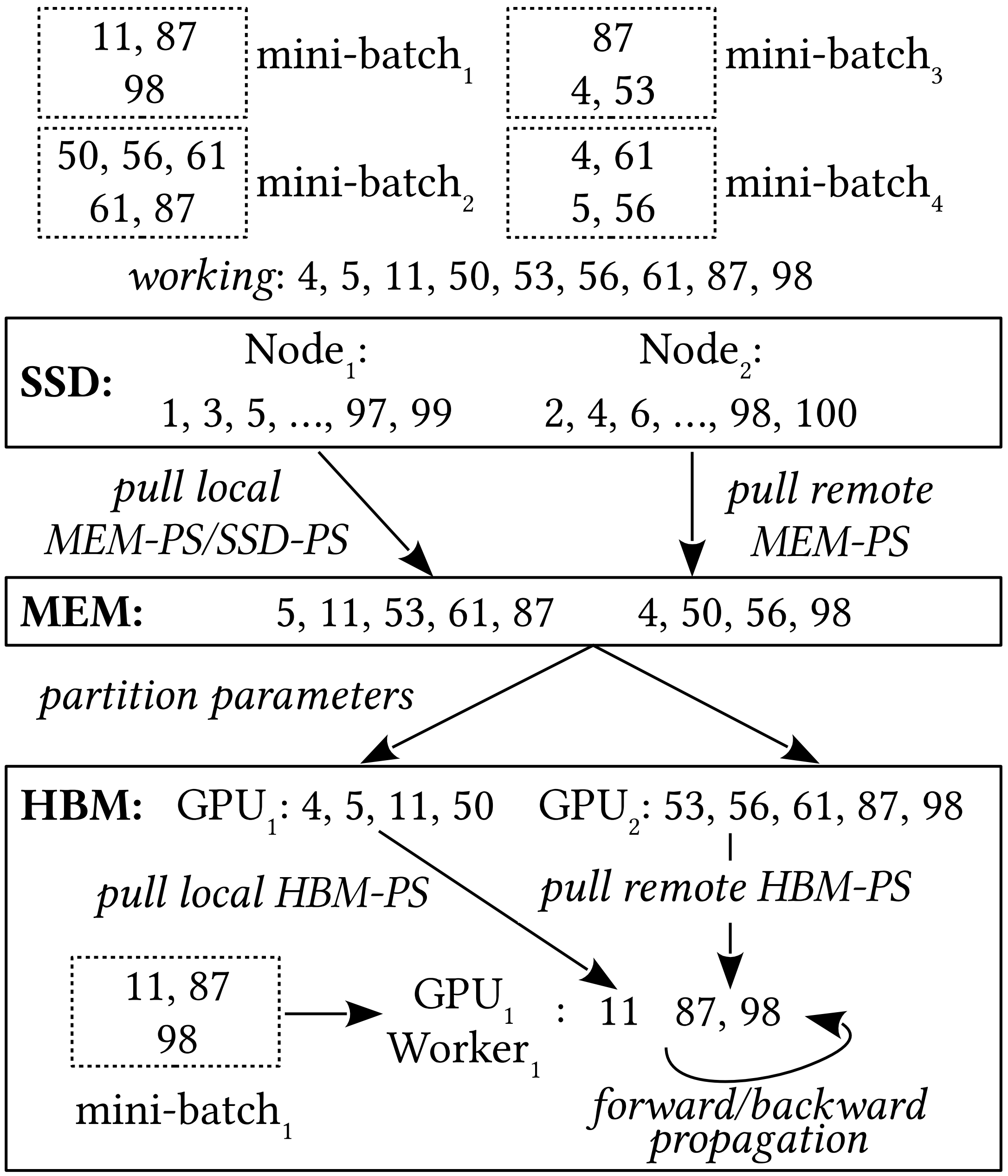
The working parameters can be held in GPU High Bandwidth Memory (HBM)

GPU Computing Node Architecture



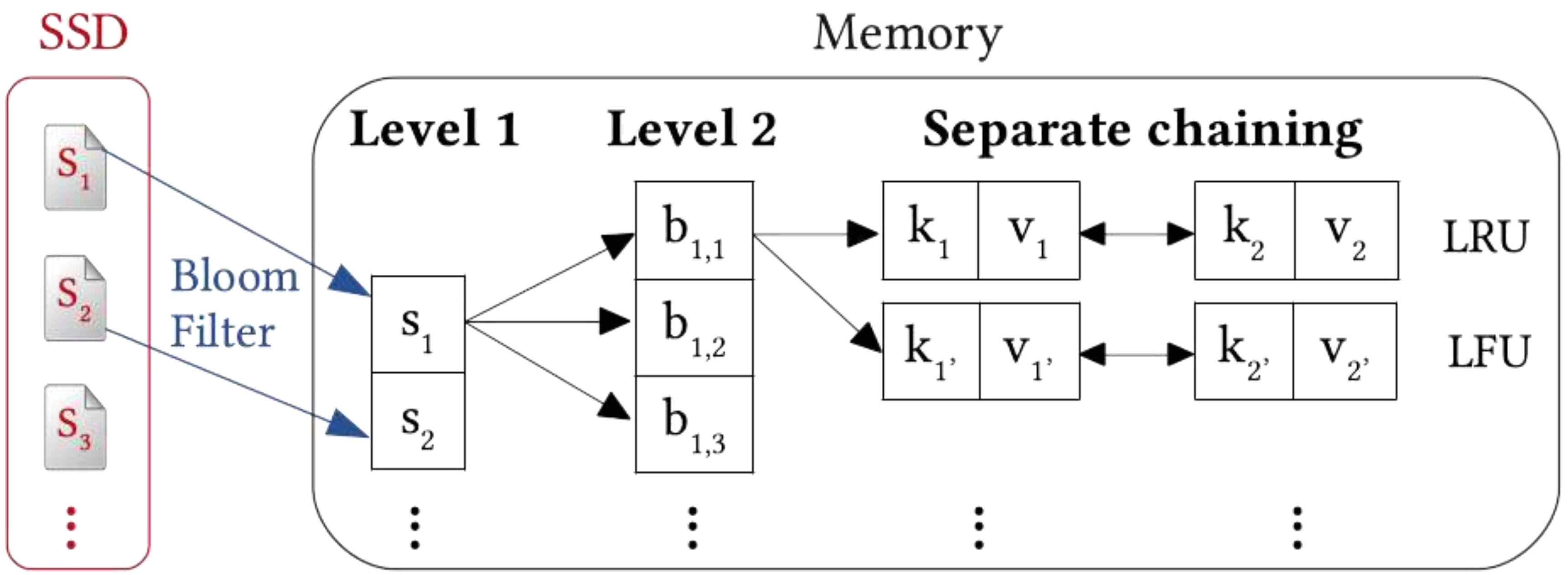
Solve the Machine Learning Problem in a System Way!





11, 87
98
mini-batch1
87
4, 53
mini-batch3
50, 56, 61
61, 87
mini-batch2
4, 61
5, 56
mini-batch4
working: 4, 5, 11, 50, 53, 56, 61, 87, 98
SSD:
Node1:
1, 3, 5, ..., 97, 99
Node2:
2, 4, 6, ..., 98, 100
pull local
MEM-PS/SSD-PS
pull remote
MEM-PS
MEM:
5, 11, 53, 61, 87
4, 50, 56, 98
partition parameters
HBM:
GPU1: 4, 5, 11, 50
GPU2: 53, 56, 61, 87, 98
pull local HBM-PS
pull remote HBM-PS
11, 87
98
mini-batch1
GPU1
Worker1
: 11
87, 98
forward/backward
propagation

MEM-PS and SSD-PS



[1] Weijie Zhao, Deping Xie, Ronglai Jia, Yulei Qian, Ruiquan Ding, Mingming Sun, and Ping Li. 2020. "Distributed Hierarchical GPU Parameter Server for Massive Scale Deep Learning Ads Systems". MLSys '20.

[2] Weijie Zhao, Jingyuan Zhang, Deping Xie, Yulei Qian, Ronglai Jia, and Ping Li. 2019. "AIBox: CTR Prediction Model Training on a Single Node". CIKM '19.

Stochastic Quantization

- A quantization range $[-w, w]$
- Divide it into 2^b bins of equal length $\triangle$. b is the bit number
- $\triangle = 2w/(2^b - 1)$
- Fixed quantization:

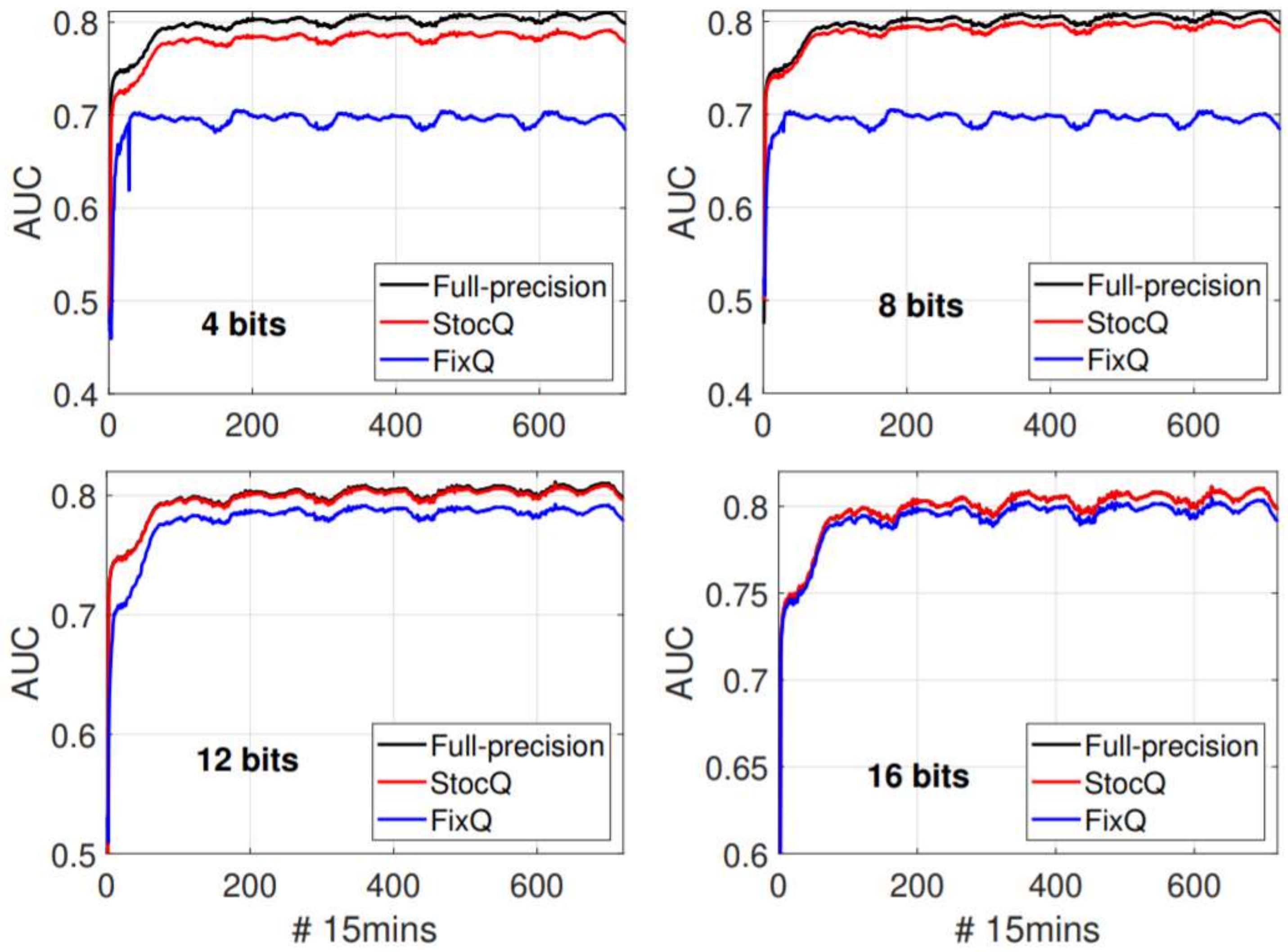
$$Q_f(x) = i^* \Delta, \text{ where } i^* = \lfloor \frac{x}{\Delta} + 0.5 \rfloor$$

- Stochastic quantization:

$$Q_s(x) = i^* \Delta, \text{ where } i^* = \lfloor \frac{x}{\Delta} + \text{rand}() \rfloor$$

- $x = 1.8$, $\triangle = 1$
- $Q_f(x) = 2$
- $Q_s(x) = \begin{cases} 2 & 80\% \\ 1 & 20\% \end{cases}$

Prediction Performance



Experimental Evaluation

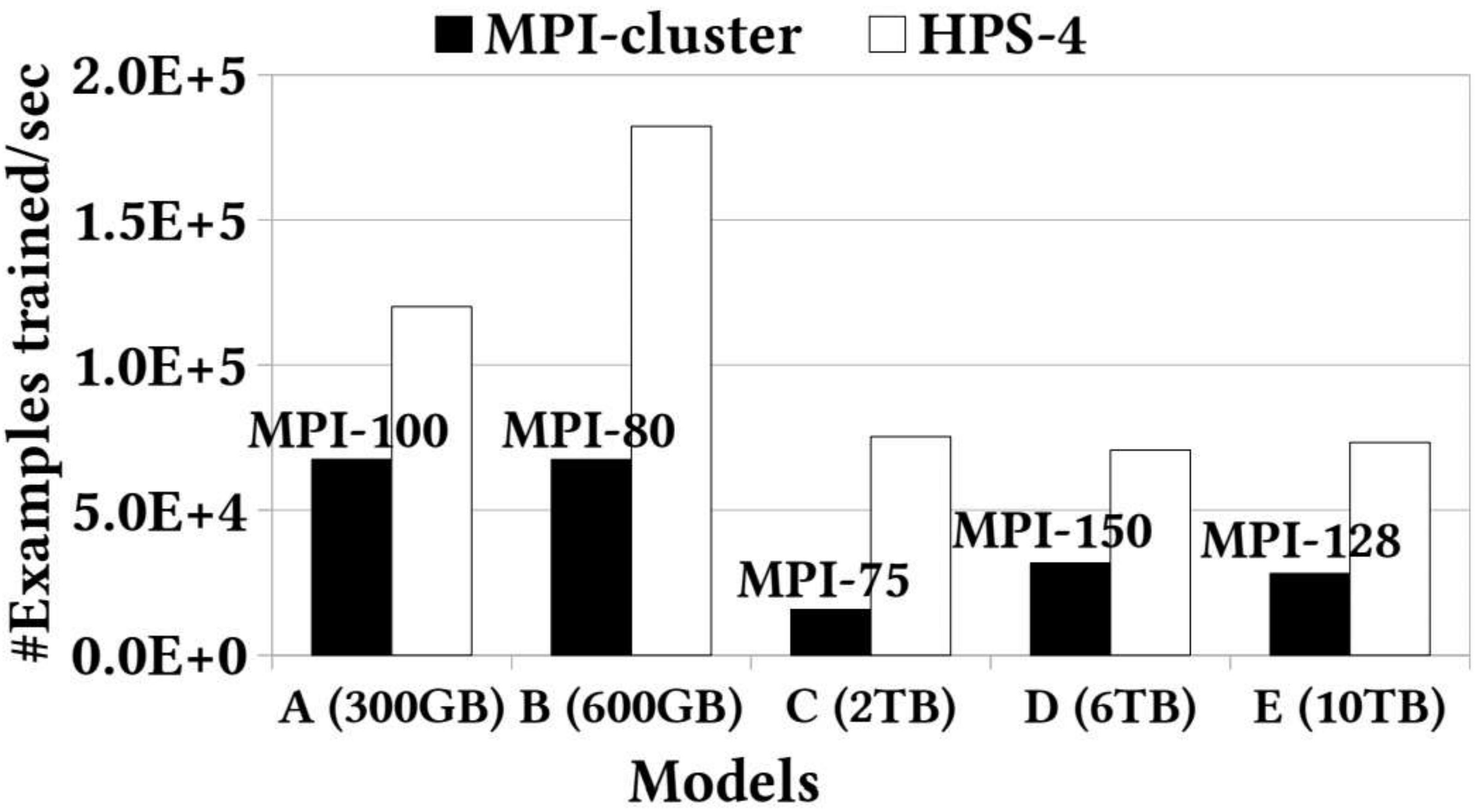
- 4 GPU computing nodes
- 8 cutting-edge 32 GB HBM GPUs
- Server-grade CPUs with 48 cores (96 threads)
- ~1 TB of memory
- ~20 TB RAID-0 NVMe SSDs
- 100 Gb RDMA network adaptor

Experimental Evaluation

Table 3. Model specifications.

	#Non-zeros	#Sparse	#Dense	Size (GB)	MPI
A	100	8×10^9	7×10^5	300	100
B	100	2×10^{10}	2×10^4	600	80
C	500	6×10^{10}	2×10^6	2,000	75
D	500	1×10^{11}	4×10^6	6,000	150
E	500	2×10^{11}	7×10^6	10,000	128

Execution Time



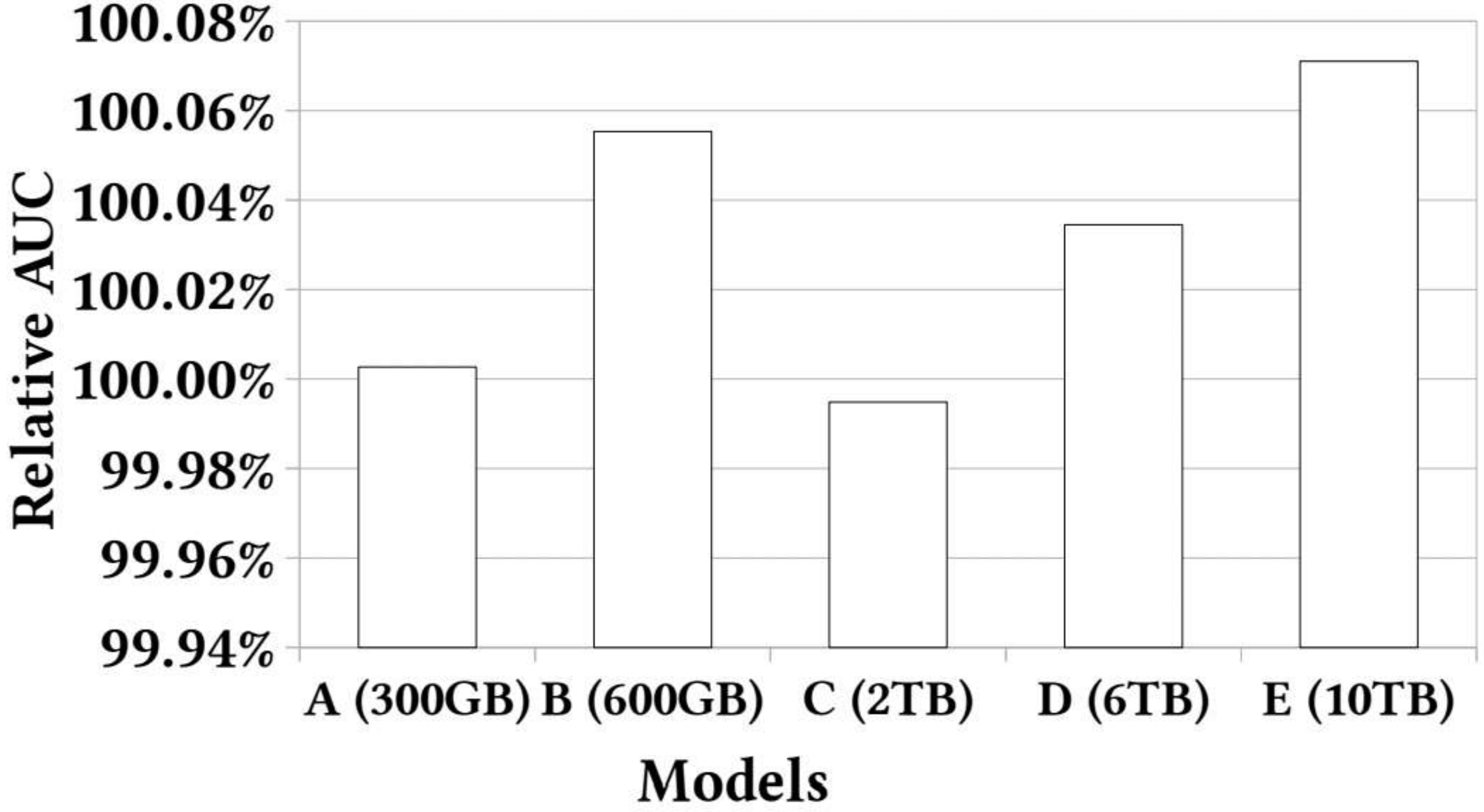
Price-Performance Ratio

- Hardware and maintenance cost: 1 GPU node ~ 10 CPU-only nodes
- 4 GPU node vs. 75-150 CPU nodes

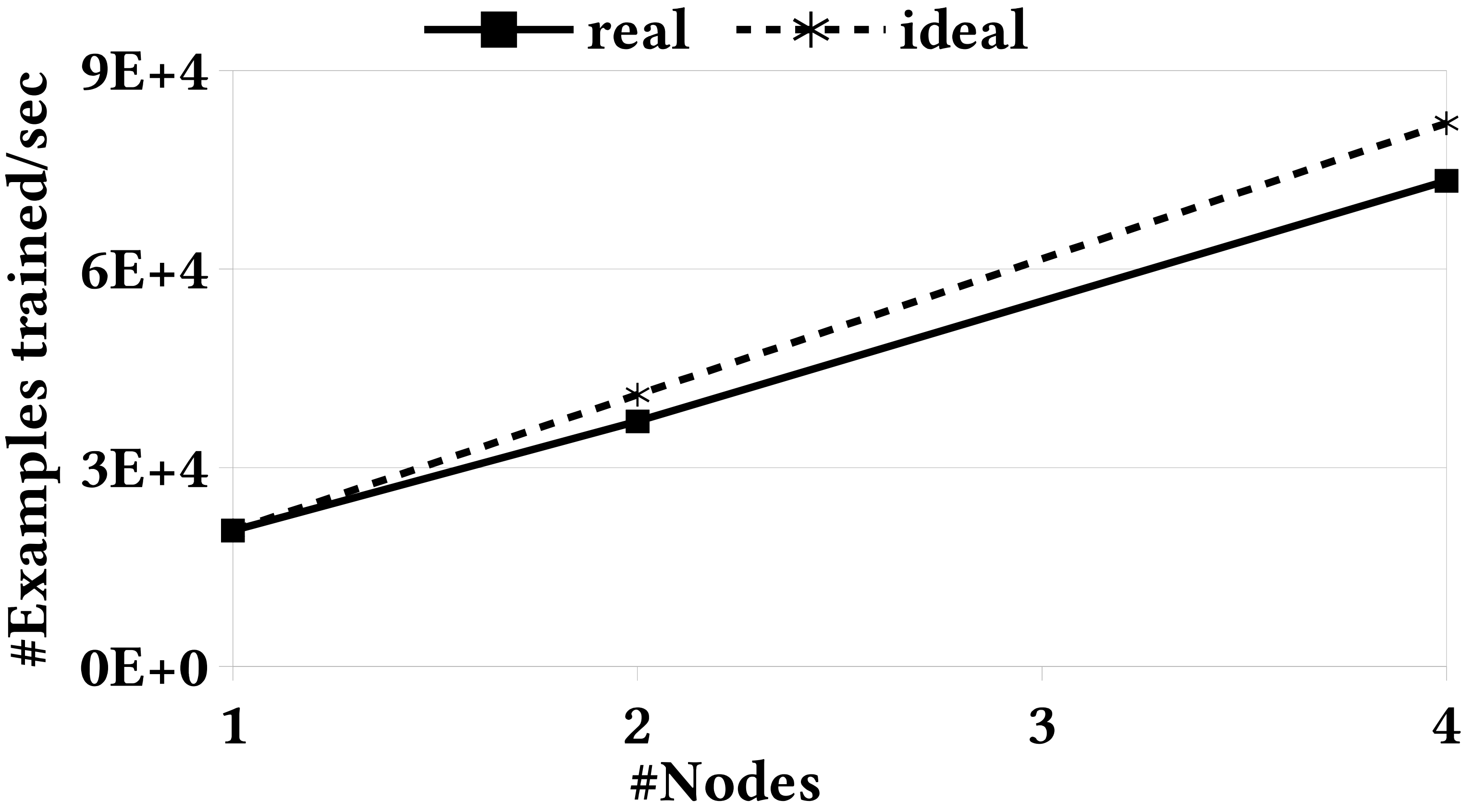
Table 4. The speedup over the MPI-cluster solution and the normalized speedup at the same hardware and maintenance cost.

	A	B	C	D	E
Speedup over MPI-cluster	1.8	2.7	4.8	2.2	2.6
Cost-normalized speedup	4.4	5.4	9.0	8.4	8.3

AUC


100.08%
100.06%
100.04%
100.02%
100.00%
99.98%
99.96%
99.94%
Relative AUC
A (300GB)
B (600GB)
C (2TB)
D (6TB)
E (10TB)
Models

Scalability



Conclusions

- We introduce the architecture of a distributed hierarchical GPU parameter server for massive deep learning ads systems.
- We perform an extensive set of experiments on 5 CTR prediction models in real-world online sponsored advertising applications.
- A 4-node hierarchical GPU parameter server can train a model more than 2X faster than a 150-node in-memory distributed parameter server in an MPI cluster.
- The cost of 4 GPU nodes is much less than the cost of maintaining an MPI cluster of 75-150 CPU nodes.
- The price-performance ratio of this proposed system is 4.4-9.0X better than the previous MPI solution.