Pre-train and Search: Efficient Embedding Table Sharding with Pre-trained Neural Cost Models

Daochen Zha†, Louis Feng‡, Liang Luo‡, Bhargav Bhushanam‡, Zirui Liu†, Yusuo Hu‡, Jade Nie‡, Yuzhen Huang‡, Yuandong Tian‡, Arun Kejariwal‡, Xia Hu†

†*Department of Computer Science, Rice University*
‡*Meta Platforms, Inc.*

Background of Embedding Table Sharding

- **What is the embedding table?** Embedding learning is used for modeling categorical features in deep recommendation models (DLRMs). It maps sparse categorical features into dense vectors.
- **What is the embedding table sharding problem?** Industrial recommendation models demand an extremely large number of parameters for embedding tables, requiring multi-terabyte memory. We have to partition the tables and put them in multiple GPUs. However, improper partitioning can lead to imbalance.
- **What is the limitation of the existing methods?** They rely on heuristics with oversimplified cost functions, so they often have unsatisfactory sharding performance.

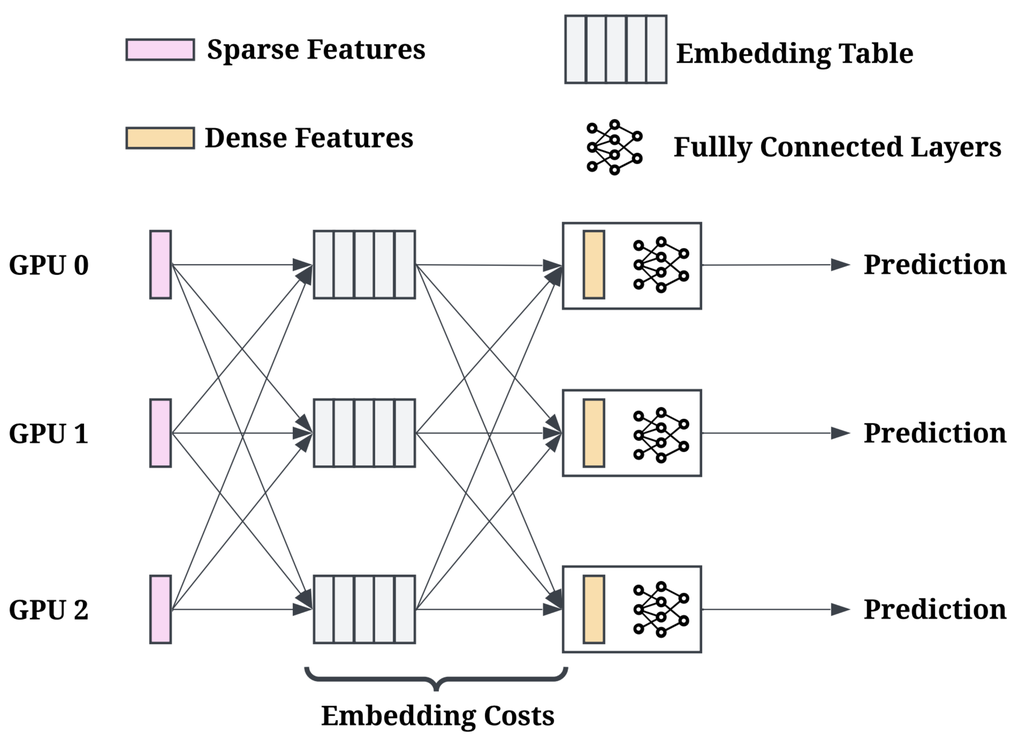

Figure: An illustrative distributed training workflow of DLRMs on three GPUs.

Our Proposal: "Pre-train and Search"

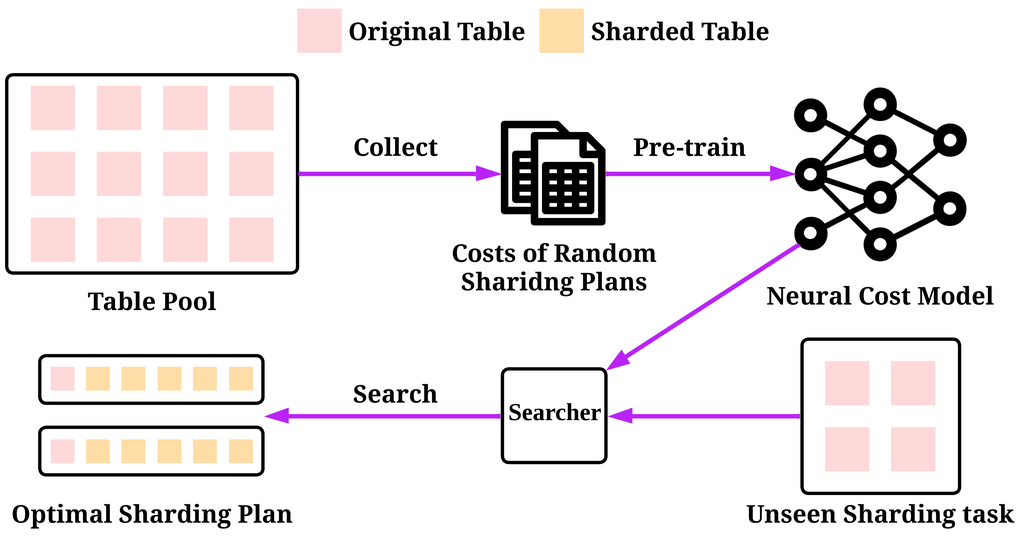

Figure: NeuroShard identifies the optimal sharding plan with *"pre-train, and search"*.

- **High-level idea.** Pre-train a universal neural cost model and then perform a search with the cost model.

Implementation

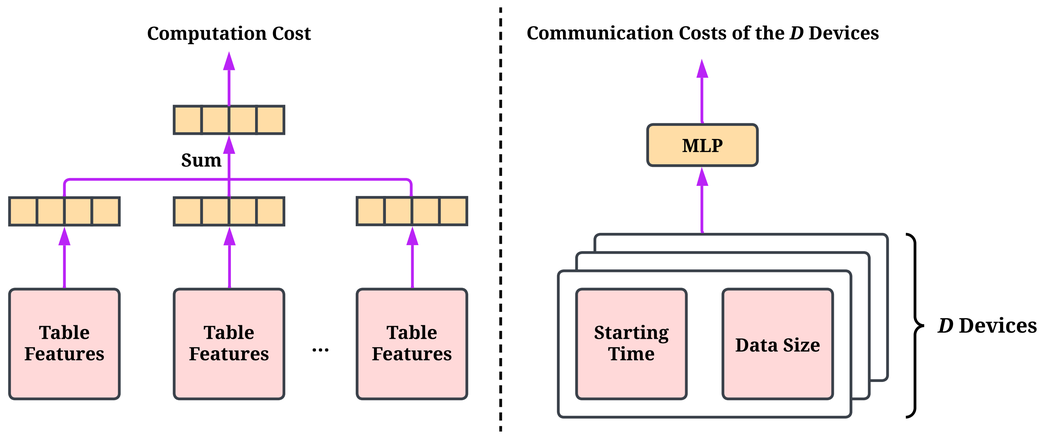

Figure: Computation (left) and communication (right) cost models. We train separate models to predict computation and communication costs.

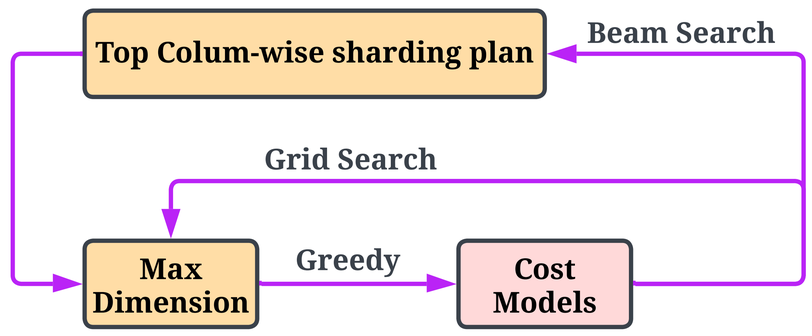

Figure: The search process. The outer loop finds the best column-wise sharding plan with beam search. The inner loop focuses on the max dimension constraint of greedy allocation with grid search.

Results

NeuroShard outperforms the existing sharding strategies on benchmark sharding datasets (results available in full text). Additionally, it also boosts the training throughput of a production-scale model.

Sharding Algorithm	Embedding Cost (Milliseconds)	Training Throughput Improvement
Random	118.3	-
Size-based	107.6	+4.0%
Dim-based	90.8	+13.9%
Lookup-based	102.4	+11.9%
Size-lookup-based	109.2	+12.8%
AutoShard	86.6	+32.4%
DreamShard	61.6	+45.3%
TorchRec	86.4	+34.6%
NeuroShard	55.2	+54.9%

Figure: Embedding cost and overall training throughput improvement of NeuroShard and baselines on a production model. The results are collected from a training cluster with 128 GPUs.

Paper

Code