AutoShard: Automated Embedding Table Sharding for Recommender Systems

Daochen Zha†, Louis Feng‡, Bhargav Bhushanam‡, Dhruv Choudhary‡, Jade Nie‡,
Yuandong Tian‡, Jay Chae‡, Yinbin Ma‡, Xia Hu†

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

Background of Embedding Tables

- **What is embedding table?** Embedding learning is a core technique for modeling categorical features in deep recommendation models. It maps sparse categorical features into dense vectors.
- **What is the 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.

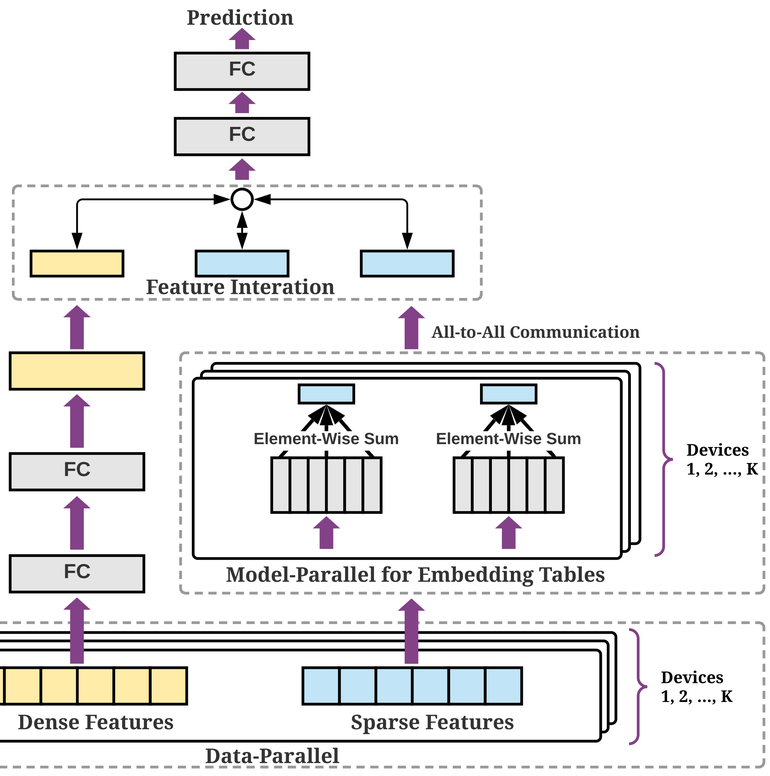


Figure: A typical recommendation model with dense and sparse features. The system exploits a combination of model parallelism (i.e., the embedding tables are partitioned into different devices) and data parallelism (i.e., replicating MLPs on each device and partitioning training data into different devices). The embedding vectors obtained from embedding lookup are appropriately sliced and transferred to the target devices through an all-to-all communication.

Embedding Table Sharding Problem

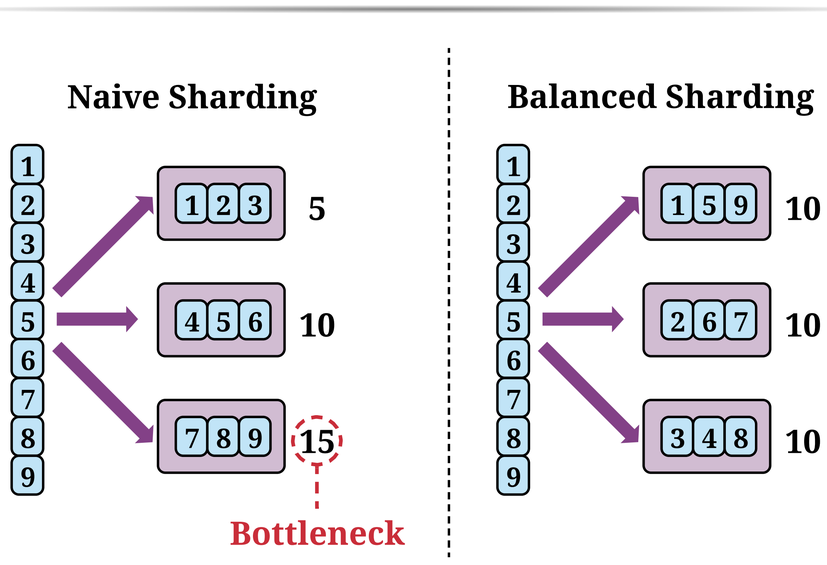


Figure: An illustrative sharding problem of partitioning 9 embedding tables across 3 devices with naive sharding and balanced sharding. Optimizing the naive sharding in this task can achieve 1.5X speedup (i.e., $15/10 = 1.5$).

- **Objective.** Given a number of embedding tables, we aim to find the optimal strategy to place them to minimize the max cost.

Challenges

- It is challenging to estimate the costs. The total cost of multiple tables in a shard is not the sum of the single table costs within the shard due to parallelism and operator fusion.
- The partitioning problem is known to be NP-hard.

AutoShard Framework

We present our novel practice in Meta, namely AutoShard, based on cost modeling and deep reinforcement learning (RL).

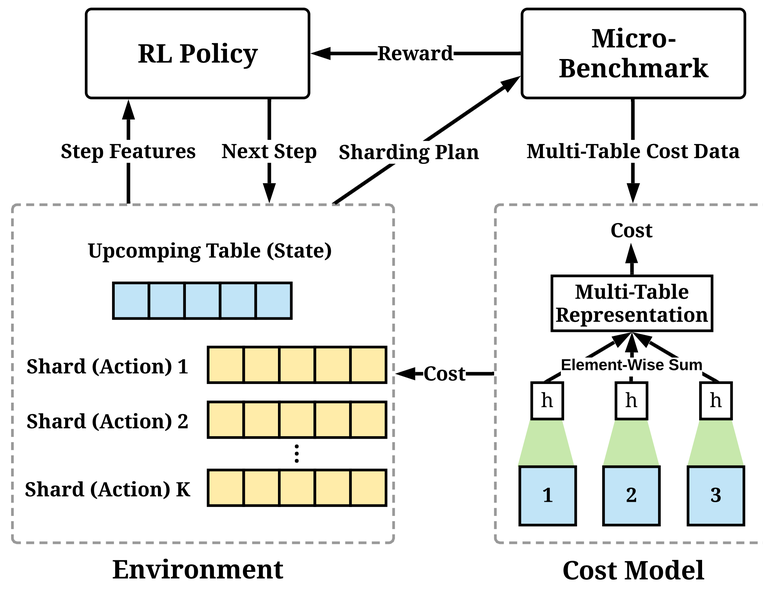
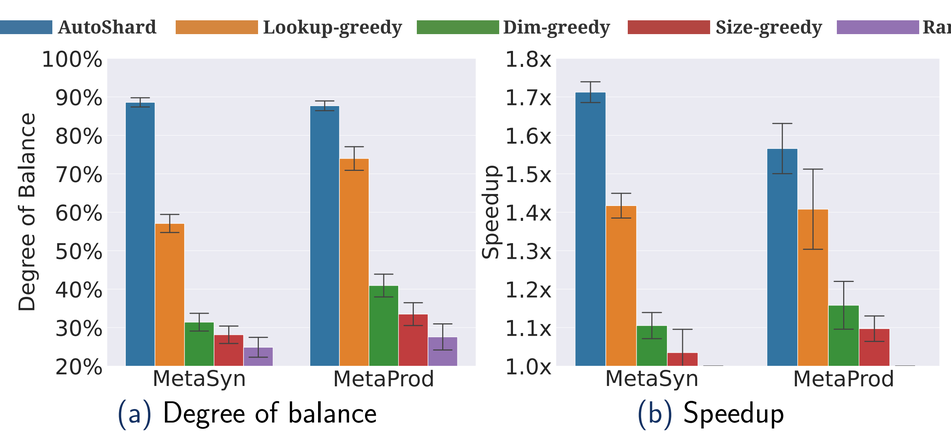


Figure: An overview of AutoShard. The RL policy interacts with the environment and allocates embedding tables one by one. Based on the generated shards, the micro-benchmark measures the actual latencies of embedding table operators and produces a reward to update the RL policy. The cost model approximates multi-table costs based on the data collected from the micro-benchmark. The estimated costs will be used to enhance the action representations in RL training.

Results



(a) Degree of balance (b) Speedup

Figure: AutoShard significantly outperforms all the baselines.

Paper

Code