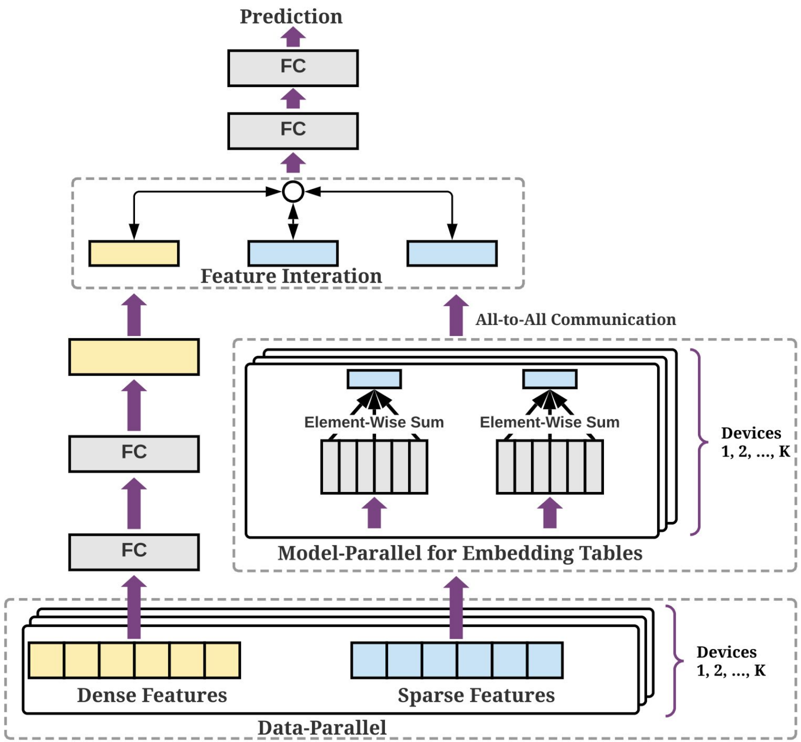
AutoShard: Automated Embedding Table Sharding for Recommender Systems

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

Rice University
Meta Platforms

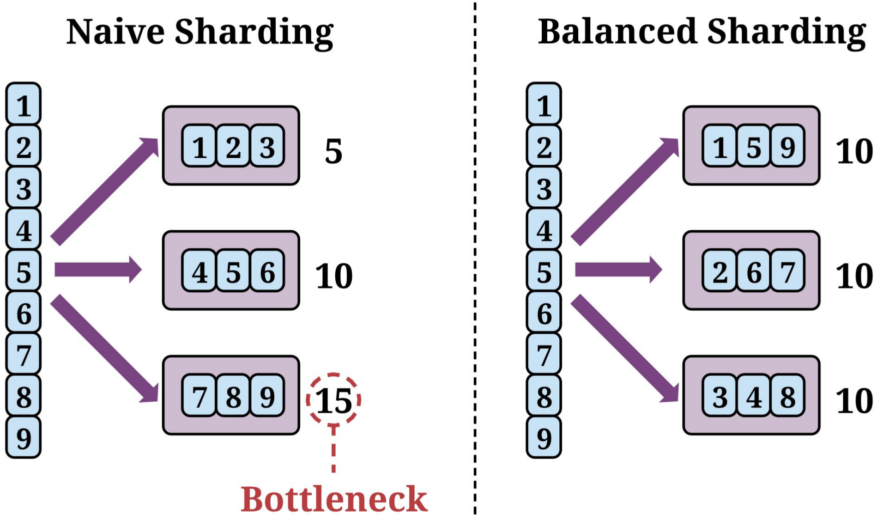
Background

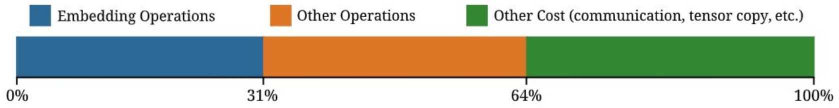


Prediction
FC
FC
Feature Interation
All-to-All Communication
Element-Wise Sum
Element-Wise Sum
Devices
1, 2, ..., K
Model-Parallel for Embedding Tables
FC
FC
Devices
1, 2, ..., K
Dense Features
Sparse Features
Data-Parallel

Embedding Table Sharding Problem

- **Problem Setting**
  - We consider embedding table sharding among GPU devices.
  - We do not consider communication cost.

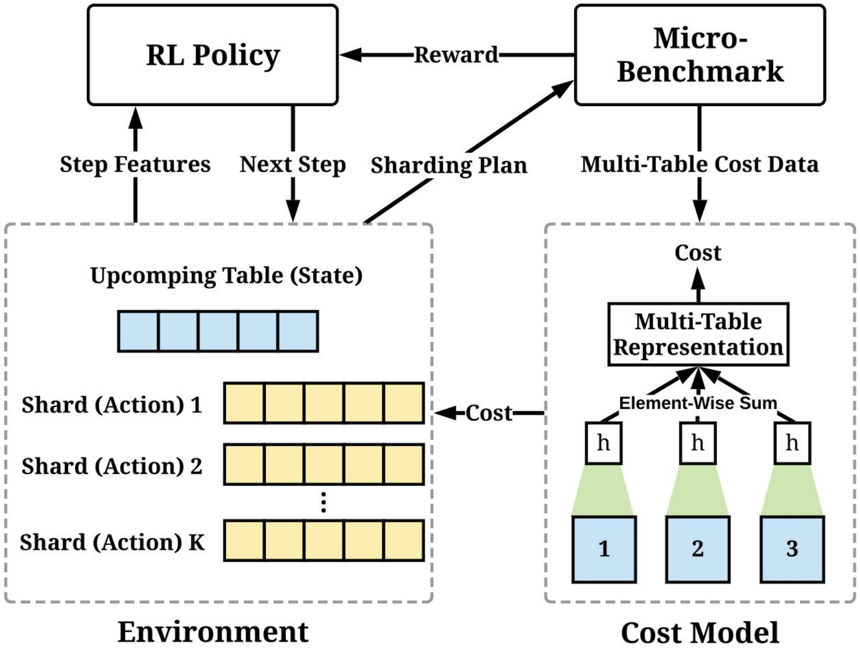




Key Challenges

- **Challenges**
  - How to efficiently estimate the cost?
  - How to partition (NP-hard problem).
- **Solution**
  - Neural cost model
  - Reinforcement learning (RL)

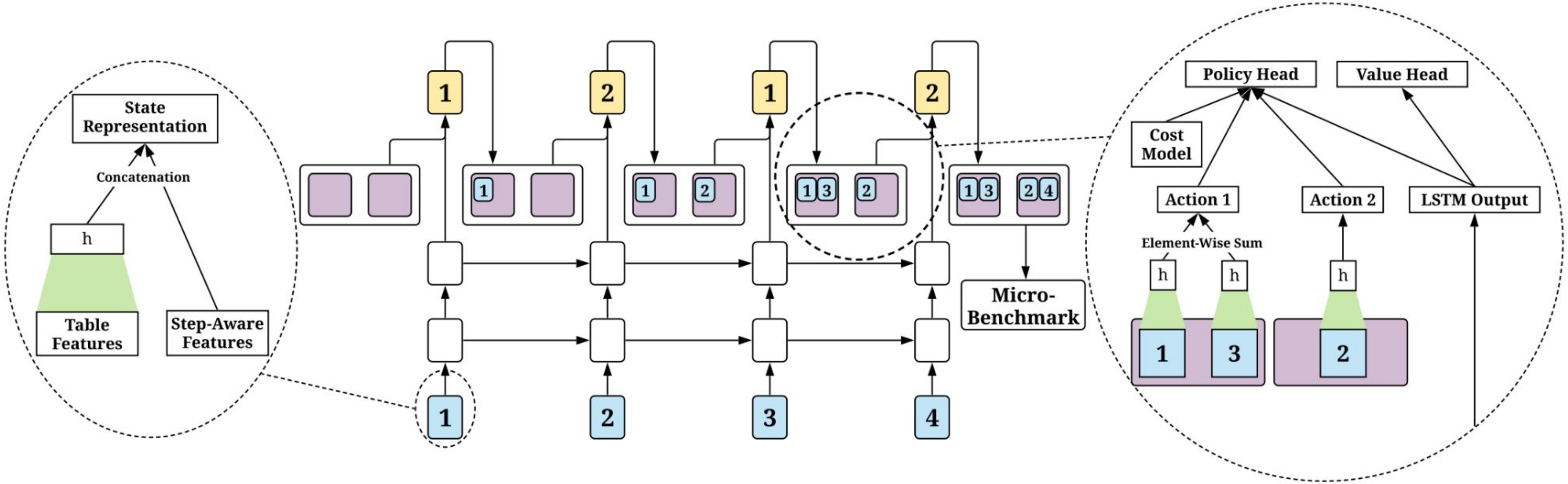
AutoShard



RL Policy
Reward
Micro-Benchmark
Step Features
Next Step
Sharding Plan
Multi-Table Cost Data
Upcomping Table (State)
Shard (Action) 1
Shard (Action) 2
Shard (Action) K
Cost
Cost
Multi-Table Representation
Element-Wise Sum
h
h
h
1
2
3
Environment
Cost Model

How Does AutoShard Shard?

- **Key Points**
  - Shard sequentially with an LSTM policy.
  - Once trained, it can transfer.



Experiments

- **Datasets**
  - MetaSyn: https://github.com/facebookresearch/dlrm_datasets
  - MetaProd: around 600 production tables

Attribute	Value
Number of Tables	856
Batch Size	65,536
Max/Mean/Min Hash Sizes	12,543,670 / 4,107,458 / 1
Max/Mean/Min Pooling Factors	193 / 15 / 0

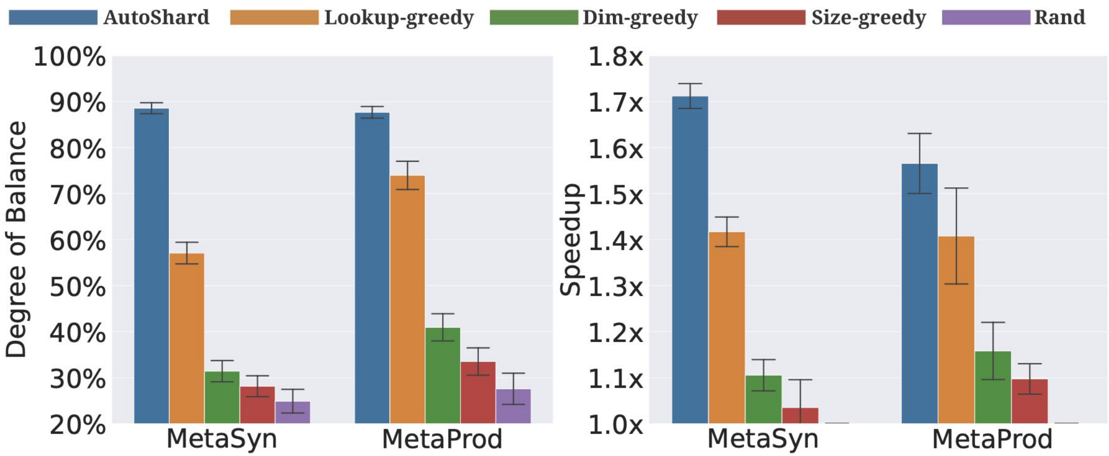
MetaSyn statistics

- **Metrics**
  - Degree of Balance: min latency / max latency
  - Speedup: max latency speedup over random sharding
- **Baselines**
  - Lookup-greedy, dim-greedy, size-greedy

Daochen Zha (daochen.zha@rice.edu)

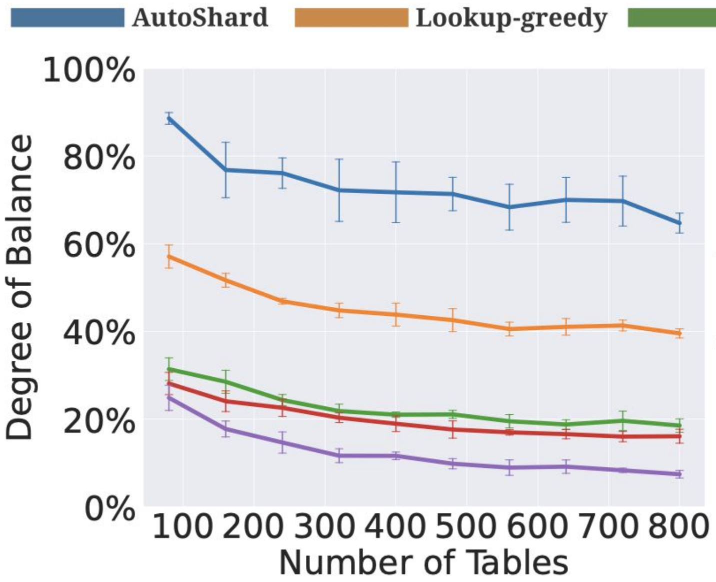
Effectiveness

- **How is it evaluated?**
  - Randomly sample 90 subsets of 80 tables from all the tables as training tasks.
  - Evaluate on another 10 subsets of 80 tables.
  - Shard to 8 GPUs

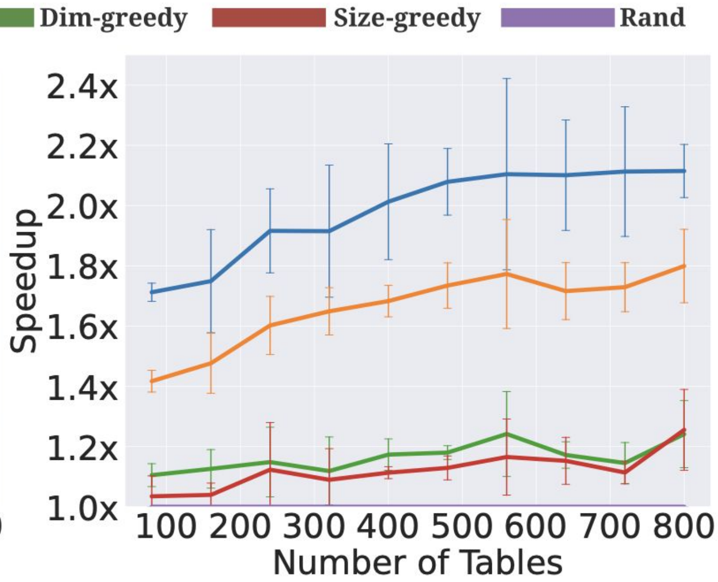


Performance of AutoShard against baselines

AutoShard Scales to Hundreds of Tables

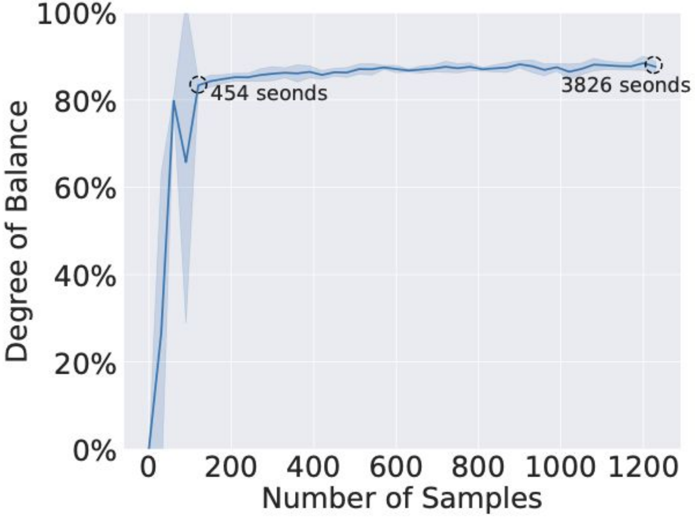


Degree of Balance

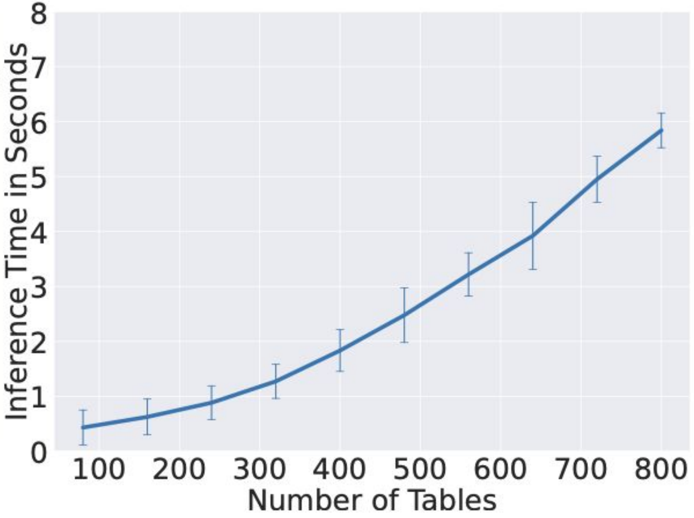


Speedup

Efficiency



Training curve on
four 2080 Ti GPUs



Inference time with
a single CPU core

Summary and Takeaways

- **Embedding table sharding problem**
  - Placing a large number of embedding tables on hundreds of (GPU) devices.
  - Challenges: cost estimation, NP-hardness.
- **Our contributions**
  - AutoShard with neural cost model and RL for sharding.
  - Validated its effectiveness on both open-sourced and production data.

Paper

Code

Daochen Zha (daochen.zha@rice.edu)