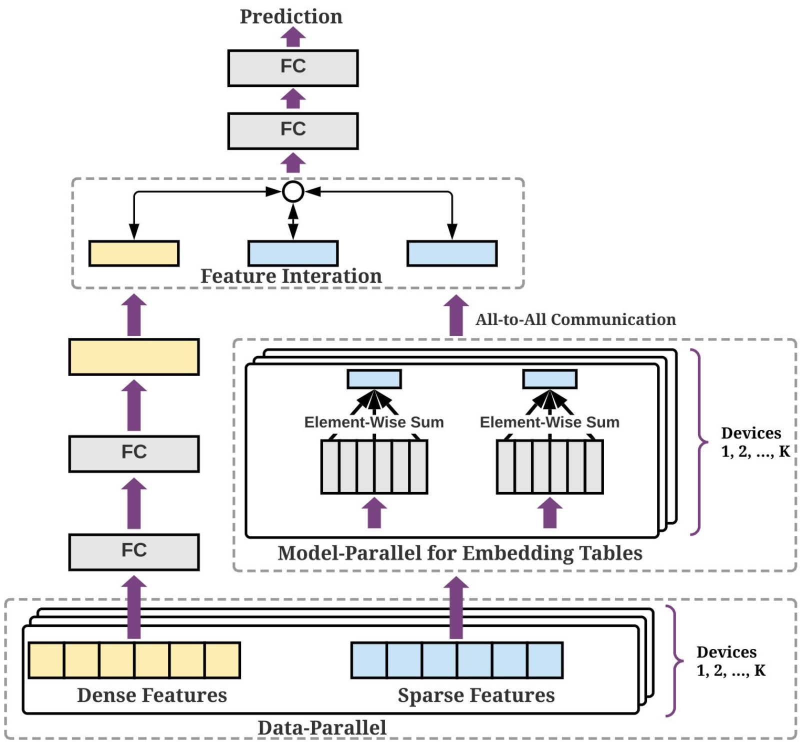
Pre-train and Search: Efficient Embedding Table Sharding with Pre-trained Neural Cost Models

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

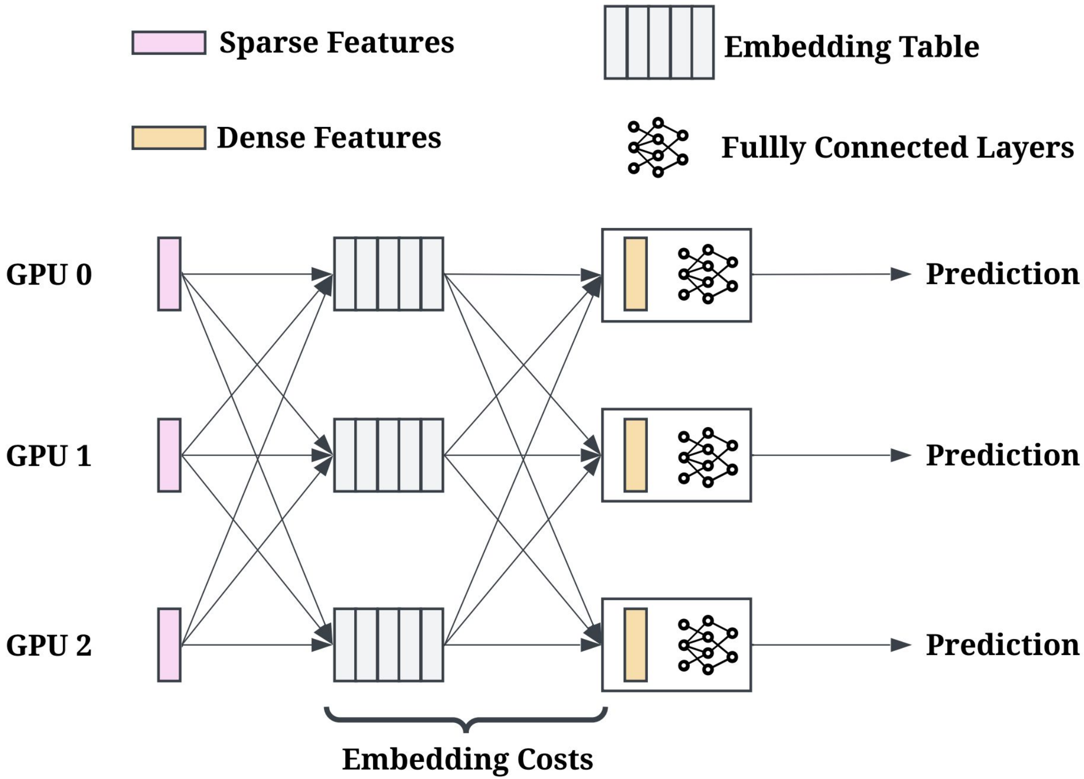
Rice University
Meta Platforms

Background



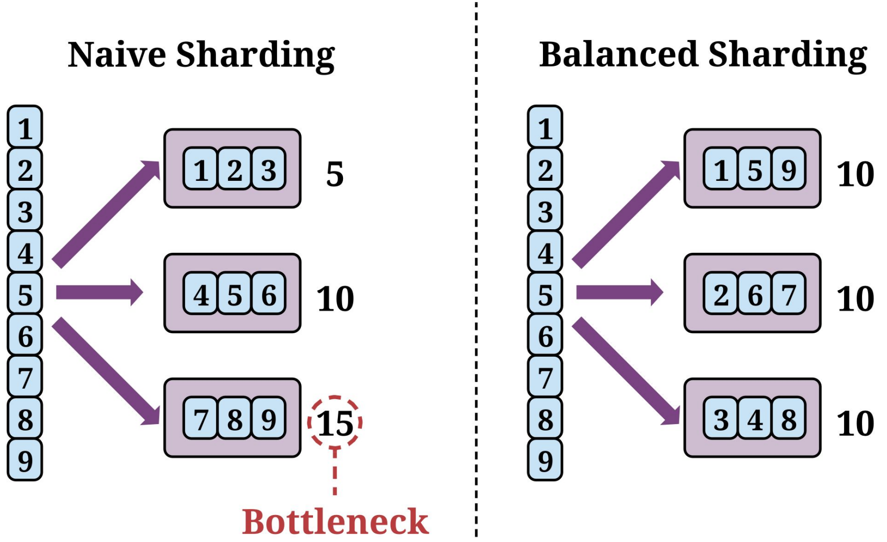
Prediction
FC
FC
Feature Interation
All-to-All Communication
Element-Wise Sum
Element-Wise Sum
Devices
1, 2, ..., K
Model-Parallel for Embedding Tables
FC
FC
Dense Features
Sparse Features
Devices
1, 2, ..., K
Data-Parallel

Background

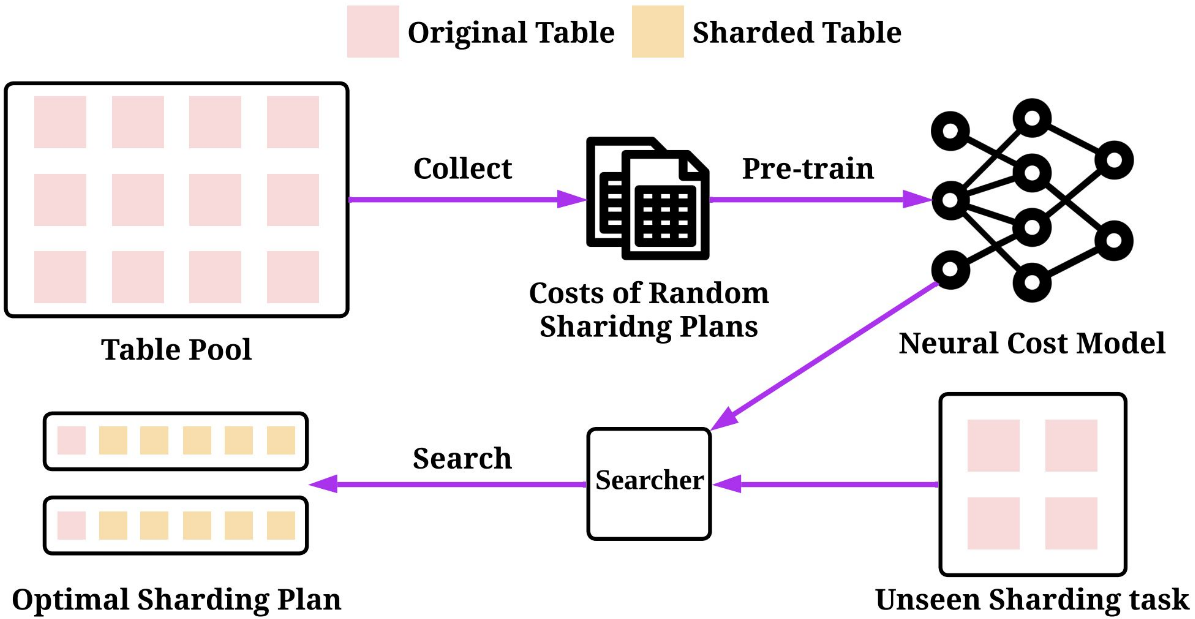

Sparse Features
Dense Features
Embedding Table
Fullly Connected Layers
GPU 0
GPU 1
GPU 2
Prediction
Prediction
Prediction
Embedding Costs

Embedding Table Sharding Problem

- **Problem Setting**
  - Given N number of embedding tables, output a sharding plan that decides 1) partitioning which tables, and 2) how to place them on GPU devices.



Our Proposal "Pre-train, and Search"

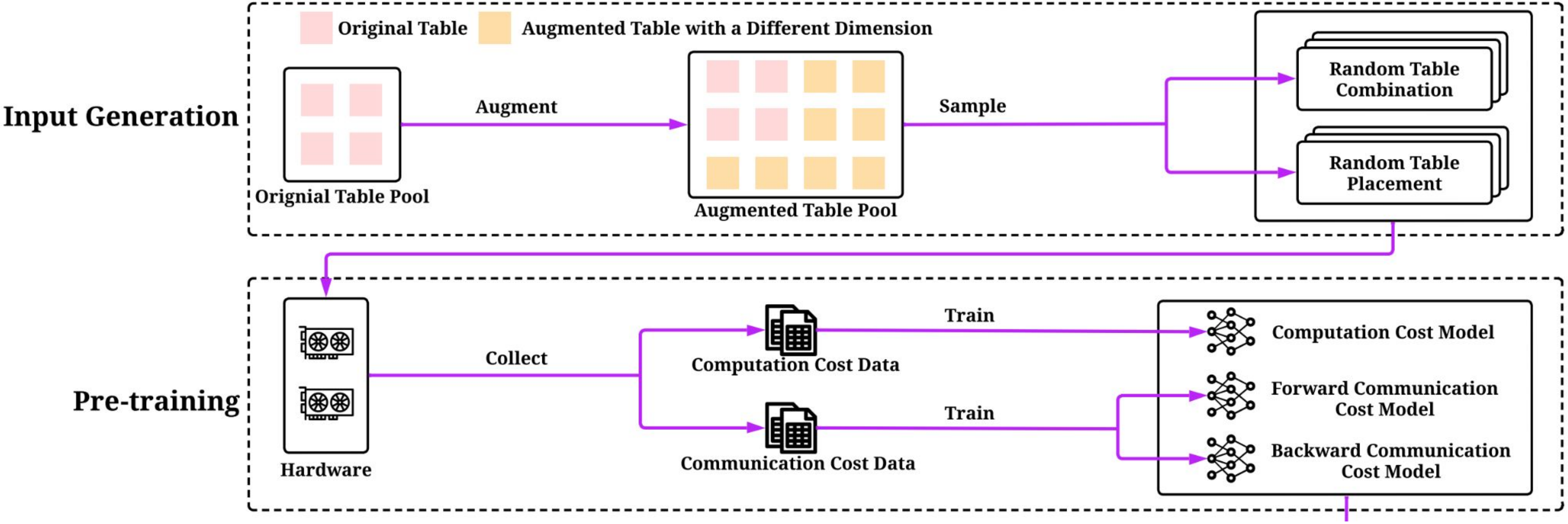


- **Why neural networks?**
  - The computation and communication costs have a nonlinear correlation with the sum of the costs of the individual tables.

Key Challenges

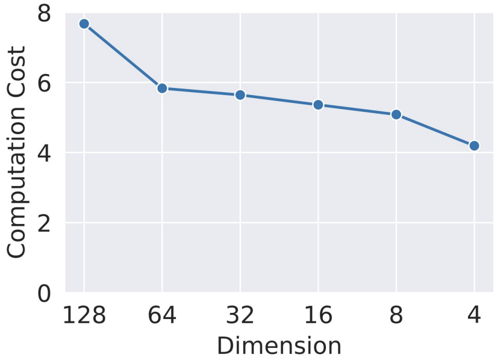
- **Challenges**
  - How to collect data and pre-train?
  - How to search (NP-hard problem).
- **Solution**
  - Neural cost models
  - Nested search process

Neural Cost Models

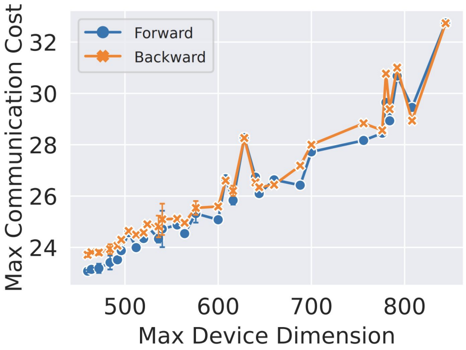


Nested Search Process

- **Key observations**
  - When partitioning a table into two halves column-wisely, the computation cost of each shard is larger than half the cost of the original table (left figure).
  - The max forward/backward communication cost among all the GPUs positively correlates with the max device dimension among all the GPUs (right figure).



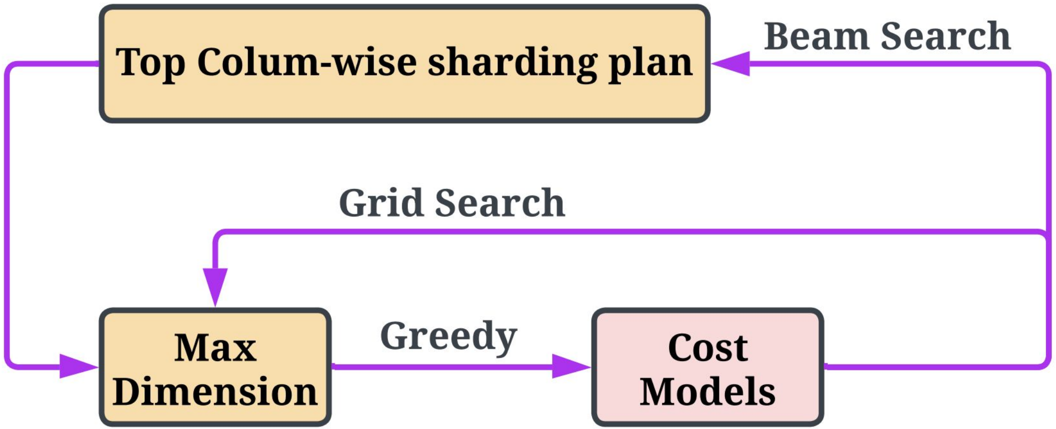
Computation cost vs. Table Dimension



Communication cost vs. Max device dimension

Nested Search Process

- **Key ideas**
  - In the outer loop, use beam search to perform column-wise sharding.
  - In the inner loop, use "greedy grid-search", i.e., 1) use max dimension as the proxy of communication cost and do grid search, and 2) use greedy algorithm (with max dimension as constraint) to assign tables.



Results

- **Settings**
  - 800+ tables sharded on 128 GPUs.

	Sharding Algorithm	Embedding Cost (Milliseconds)	Training Throughput Improvement
Heuristic	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%
Reinforcement Learning	AutoShard	86.6	+32.4%
	DreamShard	61.6	+45.3%
Planning	TorchRec	86.4	+34.6%
Proposed	NeuroShard	55.2	+54.9%

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**
  - NeuroShard with neural cost models and a nested search process.
  - Validated its effectiveness on both open-sourced and production data.

Paper

Code