DreamShard: Generalizable Embedding Table Placement for Recommender Systems

Daochen Zha, Louis Feng, Qiaoyu Tan, Zirui Liu, Kwei-Herng Lai, Bhargav Bhushanam, Yuandong Tian, Arun Kejariwal, Xia Hu

Rice University
Meta Platforms, Inc.
Texas A&M University
Distributed Recommender System

Combining data-parallelism and model-parallelism.
Embedding Table Placement Problem

• Problem Setting
  • We consider embedding table placement on GPU devices.
  • Embedding accounts for 48% and 65% of the computation and communication costs in production model.
Embedding Table Placement Problem

(a) Random placement

(b) The existing best human expert strategy

(c) DreamShard
Key Challenges

• Challenges
  • Operation fusion, which uses a single operation to subsume multiple tables, makes it hard estimate cost.
  • The adopted embedding tables and the available devices can change frequently (e.g., machine learning engineers may conduct experiments with various table combinations and numbers of devices).
Formulation of MDP

- Markov Decision Process

![Diagram showing the process of placing tables with actions and states.]

Unplaced Table -> Table to be Placed -> Placed to Device 1 -> Placed to Device 2
DreamShard Framework

Estimated MDP

State t=0

State t=1

Final State

Estimated State

Sampled Action

Estimated Reward

Cost Data

Policy Network

Device 1

Device 2

Placement

RL Agent

Real Hardware

Data Lab at Rice University

Daochen Zha (daochen.zha@rice.edu)
### Main Results

#### Observations
- DreamShard outperforms baselines significantly.
- DreamShard can generalize well (test performance is similar to train performance).

<table>
<thead>
<tr>
<th>Task</th>
<th>No strategy</th>
<th>Human Experts</th>
<th>RL</th>
<th>DreamShard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLRM-20 (4)</td>
<td>24.0±0.0</td>
<td>21.3±0.0</td>
<td>19.1±0.0</td>
<td>19.1±0.0</td>
</tr>
<tr>
<td>DLRM-40 (4)</td>
<td>41.3±0.2</td>
<td>37.7±0.0</td>
<td>33.6±0.0</td>
<td>33.6±0.0</td>
</tr>
<tr>
<td>DLRM-60 (4)</td>
<td>57.7±0.8</td>
<td>59.6±0.1</td>
<td>53.3±0.0</td>
<td>53.3±0.0</td>
</tr>
<tr>
<td>DLRM-80 (4)</td>
<td>75.7±1.0</td>
<td>77.7±0.2</td>
<td>67.8±0.0</td>
<td>67.8±0.0</td>
</tr>
<tr>
<td>DLRM-100 (4)</td>
<td>91.8±1.7</td>
<td>94.1±0.3</td>
<td>87.6±0.3</td>
<td>87.6±0.3</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLRM-20 (4)</td>
<td>23.0±0.5</td>
<td>21.3±0.0</td>
<td>19.1±0.0</td>
<td>19.1±0.0</td>
</tr>
<tr>
<td>DLRM-40 (4)</td>
<td>41.1±0.5</td>
<td>37.3±0.0</td>
<td>33.0±0.1</td>
<td>33.0±0.1</td>
</tr>
<tr>
<td>DLRM-60 (4)</td>
<td>59.1±0.6</td>
<td>59.6±0.1</td>
<td>53.2±0.0</td>
<td>53.2±0.0</td>
</tr>
<tr>
<td>DLRM-80 (4)</td>
<td>74.5±0.8</td>
<td>77.7±0.2</td>
<td>67.5±0.0</td>
<td>67.5±0.0</td>
</tr>
<tr>
<td>DLRM-100 (4)</td>
<td>94.5±6.5</td>
<td>94.1±0.3</td>
<td>87.6±0.3</td>
<td>87.6±0.3</td>
</tr>
</tbody>
</table>

**DLRM-20 (8)**
- Train: 15.6±0.4
- Test: 15.2±0.2

**DLRM-40 (8)**
- Train: 25.0±0.2
- Test: 25.2±1.3

**DLRM-60 (8)**
- Train: 34.0±0.3
- Test: 33.5±0.5

**DLRM-80 (8)**
- Train: 42.8±0.3
- Test: 41.1±0.0

**DLRM-100 (8)**
- Train: 51.5±1.2
- Test: 50.7±0.2

**Prod-20 (2)**
- Train: 41.3±0.7
- Test: 42.8±0.4

**Prod-40 (4)**
- Train: 35.1±0.3
- Test: 38.3±0.3

**Prod-80 (8)**
- Train: 43.2±0.2
- Test: 47.7±0.4

**Human Experts**
- Size-based
- Dim-based
- Lookup-based
- Size-lookup-based
- RNN-based
- RL

**Observations**
- DreamShard outperforms baselines significantly.
- DreamShard can generalize well (test performance is similar to train performance).
Takeaways

• Summary
  • We explore embedding table placement/sharding, a direction that has been rarely explored.
  • We propose DreamShard, which learns estimated MDP and an RL agent.
  • DreamShard significantly outperforms heuristic baselines.

Paper

Code