Herald: An Embedding Scheduler for Distributed Embedding Model Training

Chaoliang Zeng, Xiaodian Cheng, Han Tian, Hao Wang, Kai Chen
Deep Learning & Sparse Features

Many categorical/sparse features need to be modeled in our world

- Color
- Taste
- Production place
- Function
- Brand
Embedding Models

Dense features

Sparse features

Embedding table
- Sparse id: 0, Embedding: [3.2, 4.5, …]
- Sparse id: 1, Embedding: [2.1, 9.5, …]
- Sparse id: 2, Embedding: [7.5, 3.3, …]
- Sparse id: 3, Embedding: [8.2, 6.9, …]

Neural network

Preference
Two Conflicting Requirements

A hardware accelerator fails to provide such large memory

Large memory capacity (hundreds of GBs or TBs) for increasing embedding table size

Hardware acceleration, e.g., GPUs, for increasing model complexity
Embedding Cache as a Remedy

Model parallelism + embedding cache

GPU workers

Neural network  Emb cache

Neural network  Emb cache

Neural network  Emb cache

Sparse parameter server(s)

Global embedding table
Cache Overhead Matters

**Cache pull** when a cache does not hit the required embedding with the latest version

**Cache push** when a cache evicts or synchronizes an updated embedding

- **WDL**
  - \( T_{\text{cache}} / T_{\text{comp}} \)
  - Normalized overhead
  - Embedding dimensions: 10G, 25G, 512
  - Range: 0.29 – 3.72

- **DFM**
  - \( T_{\text{cache}} / T_{\text{comp}} \)
  - Normalized overhead
  - Embedding dimensions: 10G, 25G, 512
  - Range: 0.25 – 2.92
Existing Optimizations

**FAE** [1] has a bias to train hot inputs that contain only hot embeddings

**HET** [2] applies a staleness-tolerant embedding update method

May affect model accuracy which is important in production

---

Our Goal

Can we reduce the communication overhead without hurting the model accuracy?

FAE [1] has a bias to train hot inputs that contain entirely hot embeddings.


May hurt model accuracy which is important in production.
Opportunities

Input

Iteration: x

Iteration: x+1

Cache hitting can avoid **cache pull**

Updated embeddings that are not required by other workers later can avoid **cache push**

For a worker:

O1. Training as much as possible in-cache embeddings
O2. Performing on-demand embedding synchronizations
Two Observations

**Predictability**

An inputs partition determines future embedding accesses and cache behaviors

*Predictability provides the feasibility of the optimization opportunities*

**Sparsity**

Most of training workload of an in-cache embedding can be accepted by only one worker

*Sparsity indicates the potential benefits of the optimization opportunities*
Herald: An Embedding Scheduler

- **O1**: Training as much as possible in-cache embeddings
- **O2**: Performing on-demand embedding synchronizations

Sparse parameter server

On-demand synchronization

GPU worker

Emb cache

Location-aware inputs partition

Batch inputs
Location-aware Inputs Partition

A heuristic partition algorithm can be found in our paper.

The required embeddings in samples most likely exist in the worker cache (with the latest version).

A list of embeddings that
1. exist in this worker cache (with the latest version)
2. are required by training samples of other workers
to support on-demand synchronization.
On-demand Synchronization

```
Communication plan at iteration \( i \) = Embedding dependencies at iteration \( i + 1 \)
```

A list of embeddings that should be synchronized to the PS at the end of this iteration.
Preliminary Evaluations

- Model consistency analysis (details in our paper)

- Simulation evaluations on Herald performance improvement
  
  ➢ A simulator with 8 LRU cache instances, each of which has a 1.6 GB capacity
  
  ➢ A baseline with naïve manner: random inputs partition + synchronizing on every updated embeddings
Overall Performance

Herald can significantly reduce cache overhead compared to a naïve manner.
Herald reduces cache pull rate by up to 48.2% and cache push rate by up to 58.4%
# Contribution Breakdown

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Pull</th>
<th>Push</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>On-demand synchronizations</td>
<td>1</td>
<td>0.84</td>
<td>0.91</td>
</tr>
<tr>
<td>Location-aware input partition</td>
<td>0.52</td>
<td>0.85</td>
<td>0.69</td>
</tr>
<tr>
<td>Herald</td>
<td>0.52</td>
<td>0.42</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 1: Breakdown of contribution by each optimization (embedding dimensions = 128).
Future Work

- **Optimization on cache replacement** to further reduce the cache miss rate
- **Prefetching communication plan** to address the workload imbalance among workers during the synchronization
- **Point-to-point embedding synchronization** to eliminate the potential network bottleneck caused by the PS architecture
Conclusion

• Problem:
  ➢ Large-scale embedding models training suffers from high embedding communication overhead
  ➢ Prior optimizations on embedding communication may hurt model accuracy

• Observations:
  ➢ Embedding cache accesses exhibit two characteristics: predictability and sparsity

• Solution:
  ➢ A runtime embedding scheduler, Herald, with two optimizations: location-aware inputs partition and on-demand embedding synchronization
  ➢ Herald can reduce the cache overhead while preserving the model accuracy

Thank you!

Contact email: czengaf@connect.ust.hk