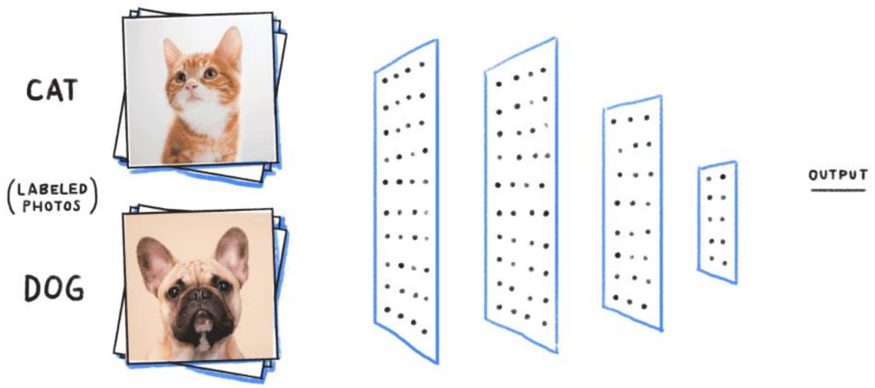
Herald: An Embedding Scheduler for Distributed Embedding Model Training

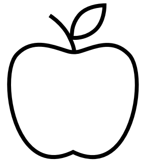
Chaoliang Zeng, Xiaodian Cheng, Han Tian, Hao Wang, Kai Chen

iSingLab HKUST

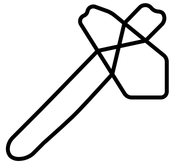
Deep Learning & Sparse Features



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

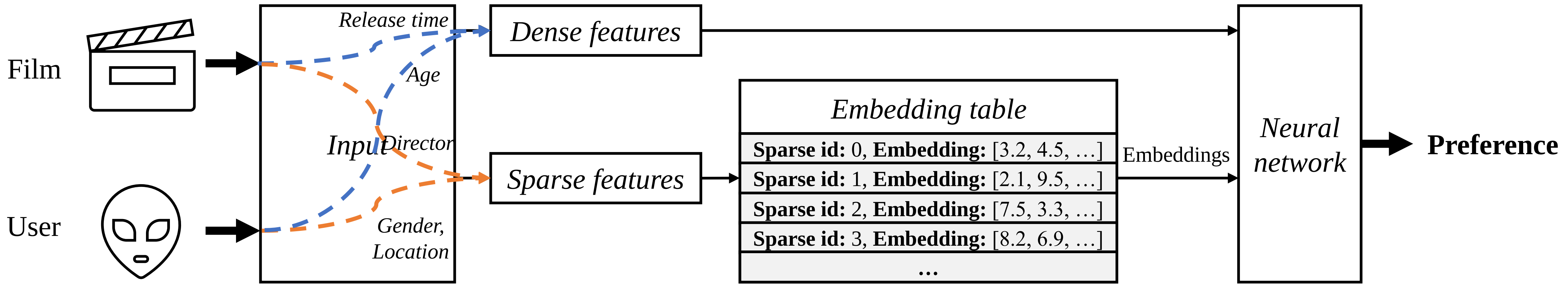


Color
Taste
Production place

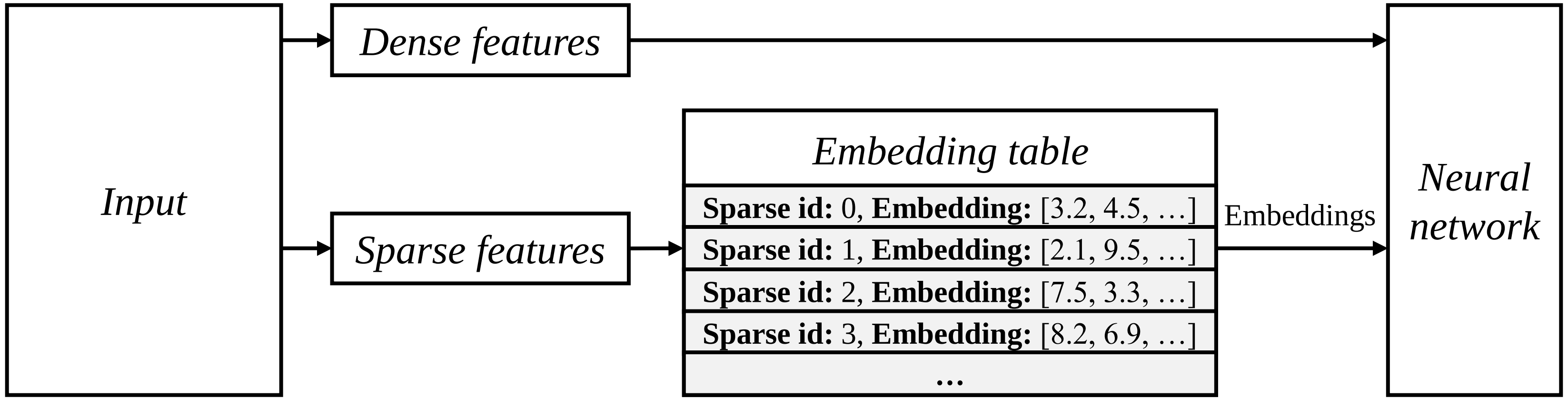


Function
Brand

Embedding Models



Two Conflicting Requirements



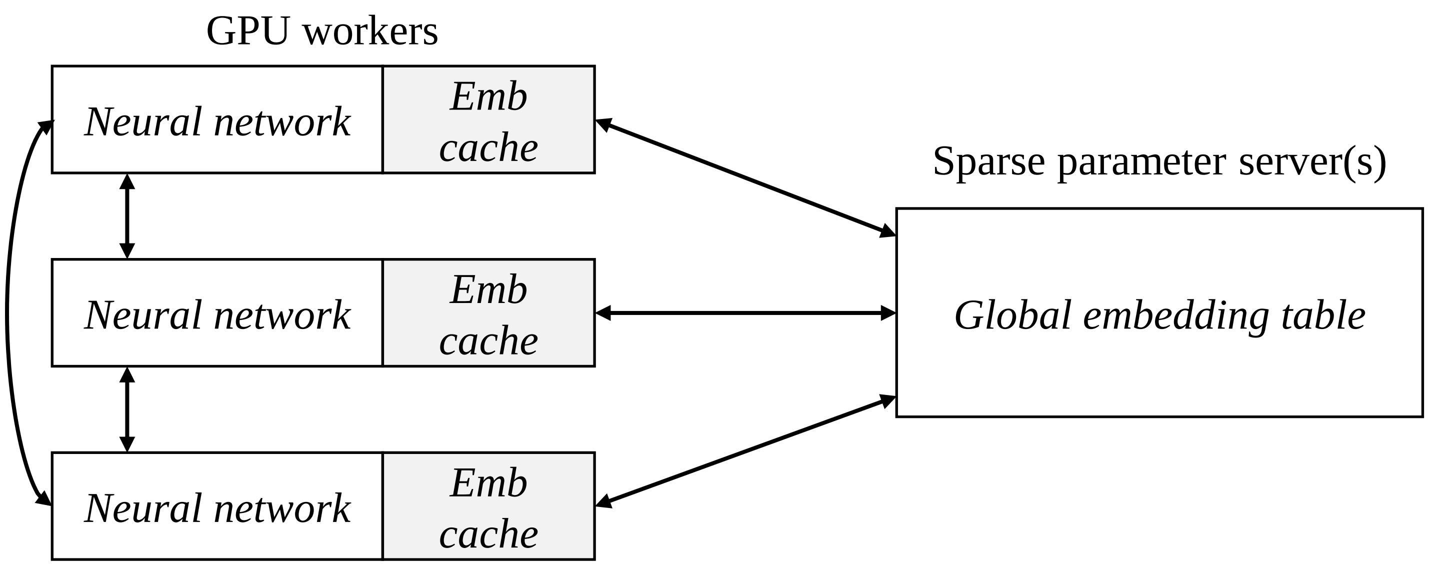
Large memory capacity (hundreds of GBs or TBs) for increasing embedding table size

Hardware acceleration, e.g., GPUs, for increasing model complexity

A hardware accelerator fails to provide such large memory

Embedding Cache as a Remedy

Model parallelism + embedding cache



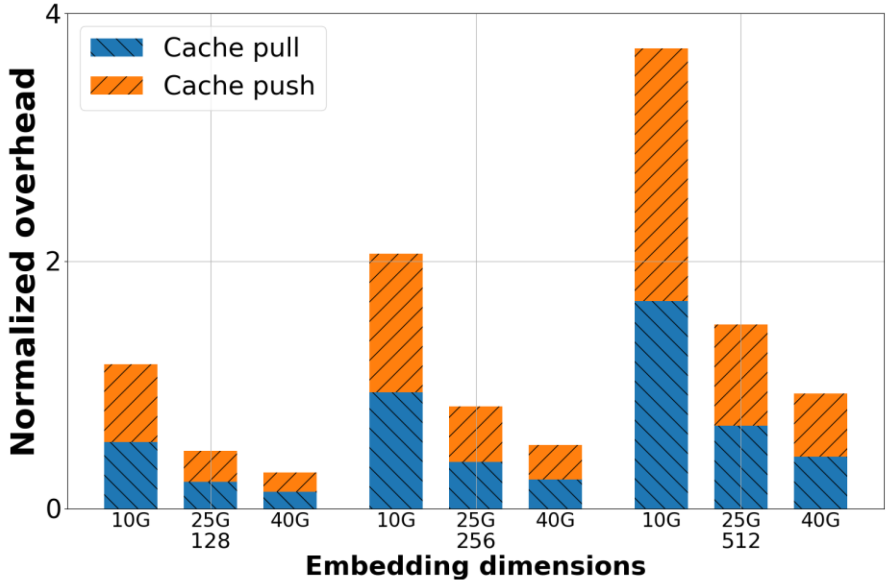
Cache Overhead Matters

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

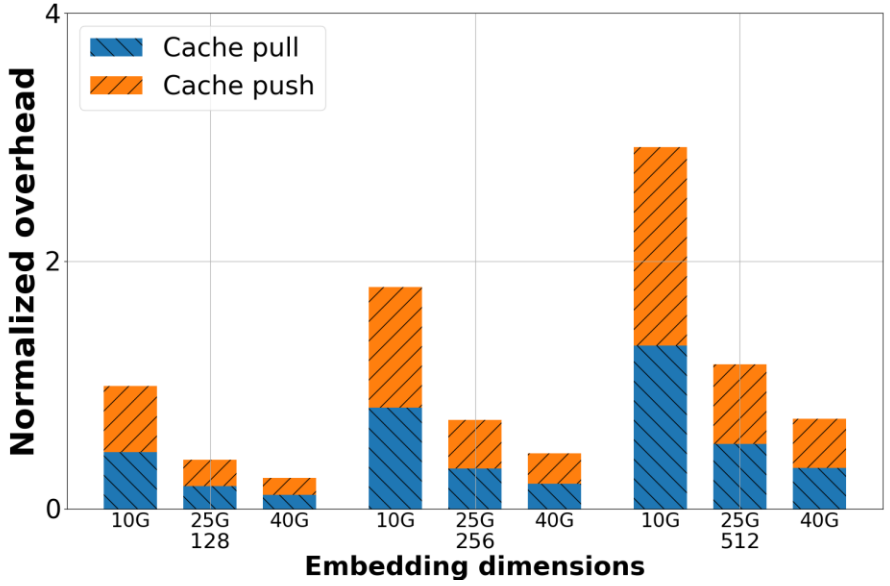
$$\frac{T_{cache}}{T_{comp}}$$

WDL



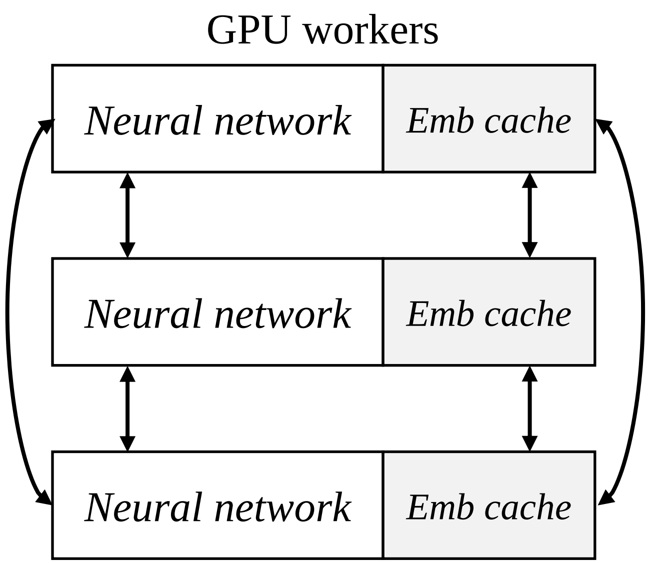
0.29 – 3.72

DFM

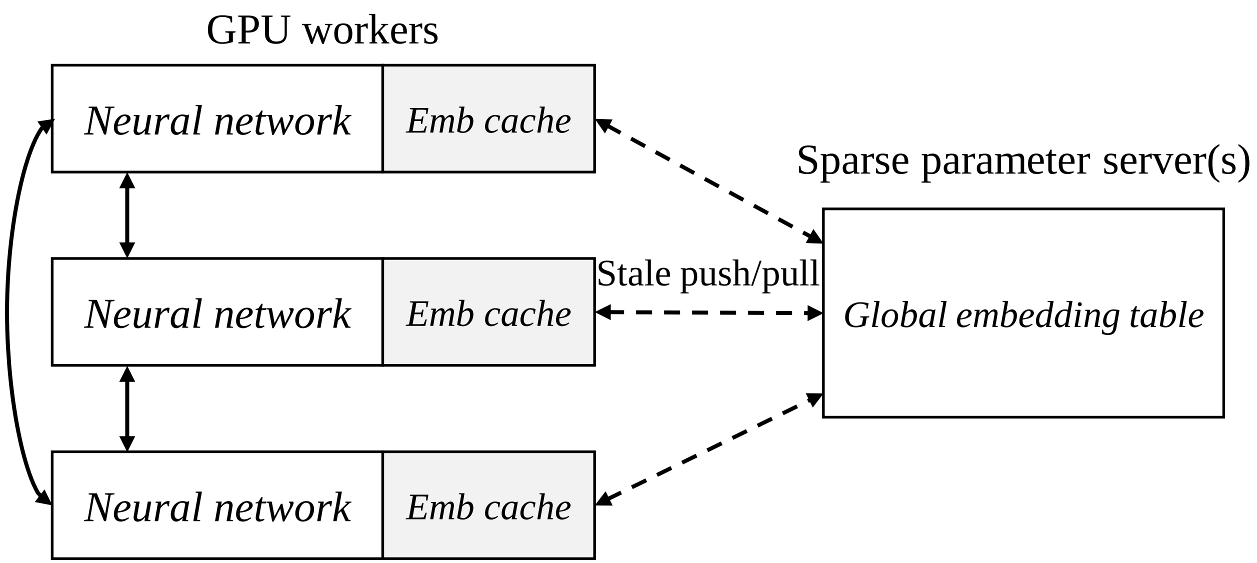


0.25 – 2.92

Existing Optimizations



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

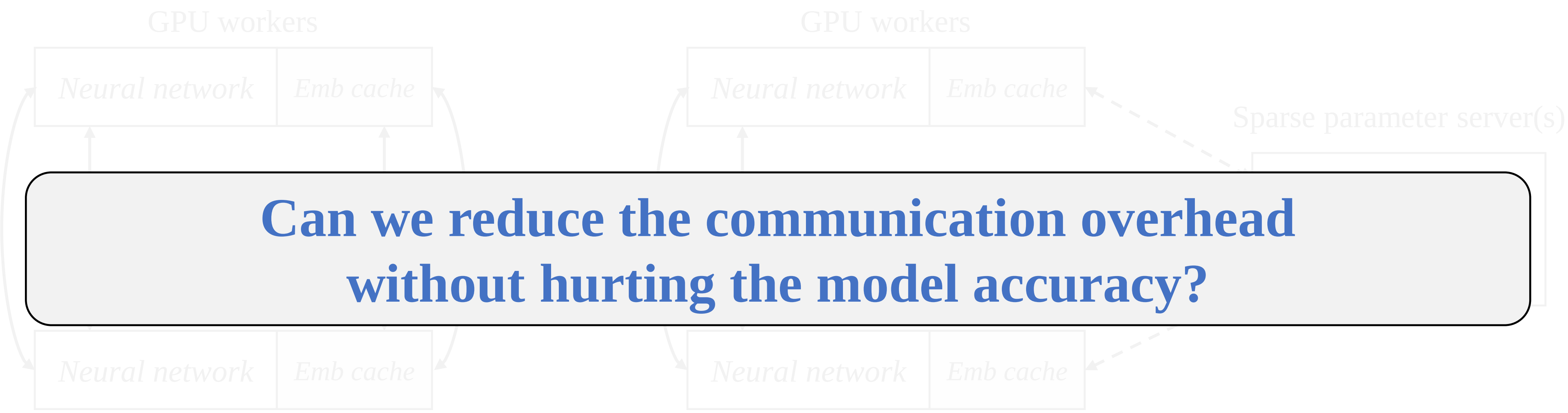


HET [2] applies a staleness-tolerant embedding update method

May affect model accuracy which is important in production

[1] M. Adnan, et al. Accelerating Recommendation System Training by Leveraging Popular Choices. VLDB, 2021.
[2] X. Miao, et al. HET: Scaling out Huge Embedding Model Training via Cache-enabled Distributed Framework. VLDB, 2021

Our Goal



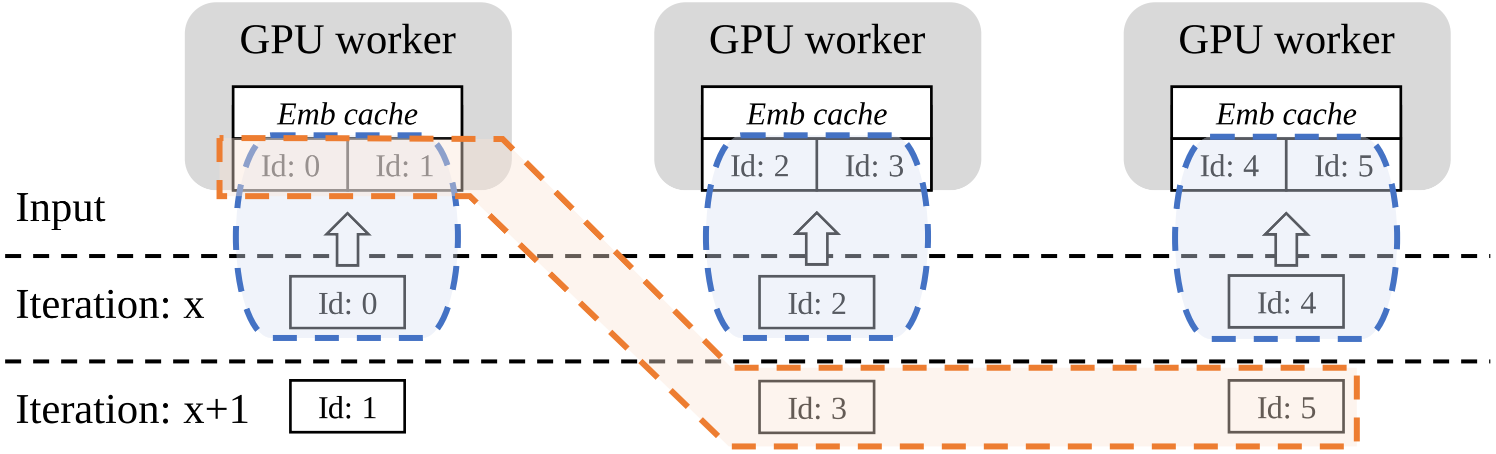
Can we reduce the communication overhead without hurting the model accuracy?

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

HET [2] applies staleness-tolerant embedding update

May hurt model accuracy which is important in production

Opportunities



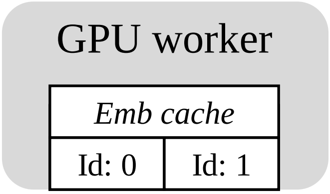
Cache hitting can avoid **cache pull**

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

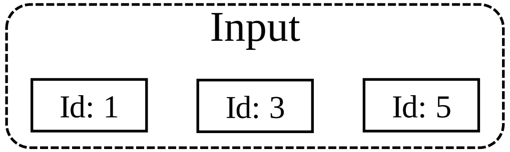
For a worker:
O1. Training as much as possible in-cache embeddings
O2. Performing on-demand embedding synchronizations

Two Observations

Predictability



Cache snapshots are observable

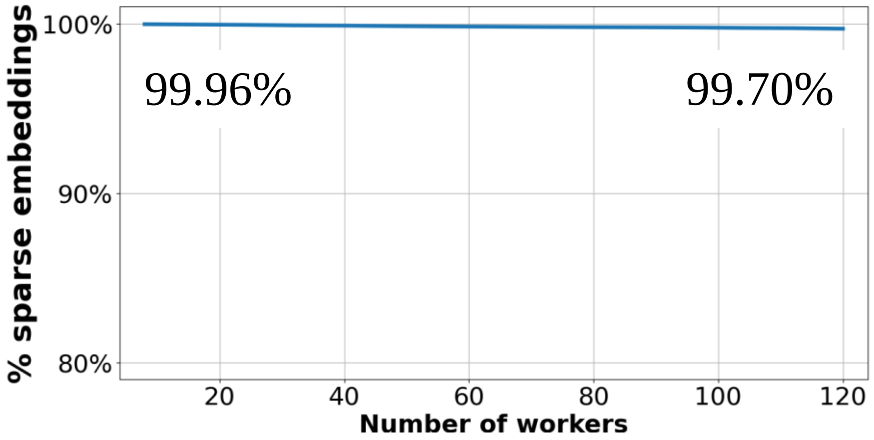


An inputs partition determines future embedding accesses and cache behaviors

Predictability provides the feasibility of the optimization opportunities

Sparsity

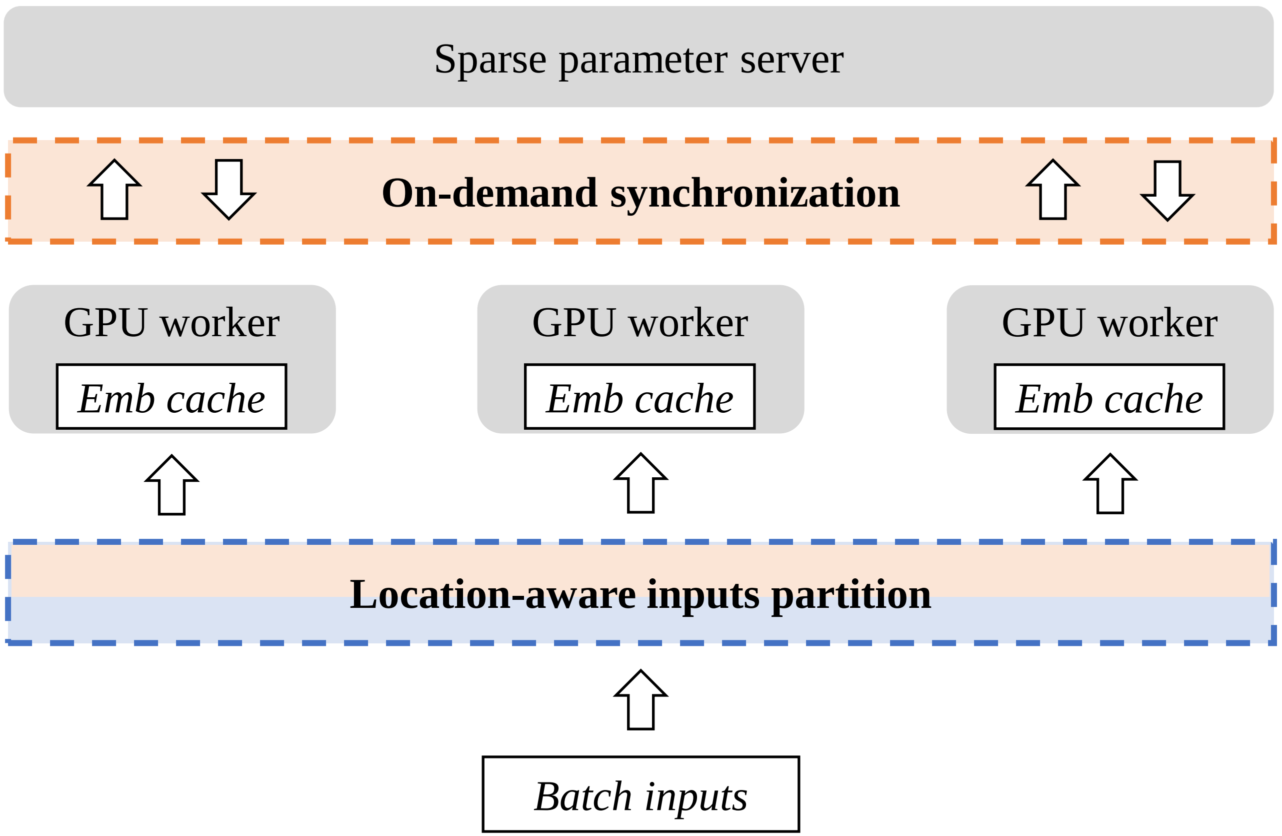
Criteo Kaggle



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

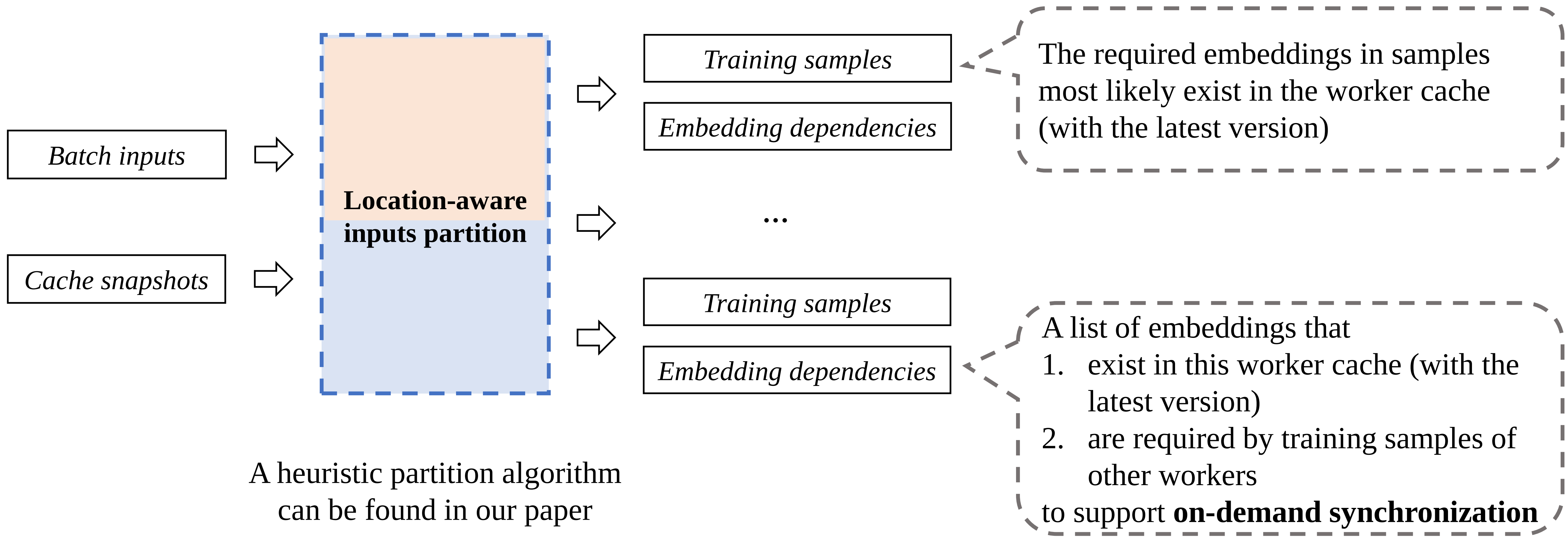
Herald: An Embedding Scheduler



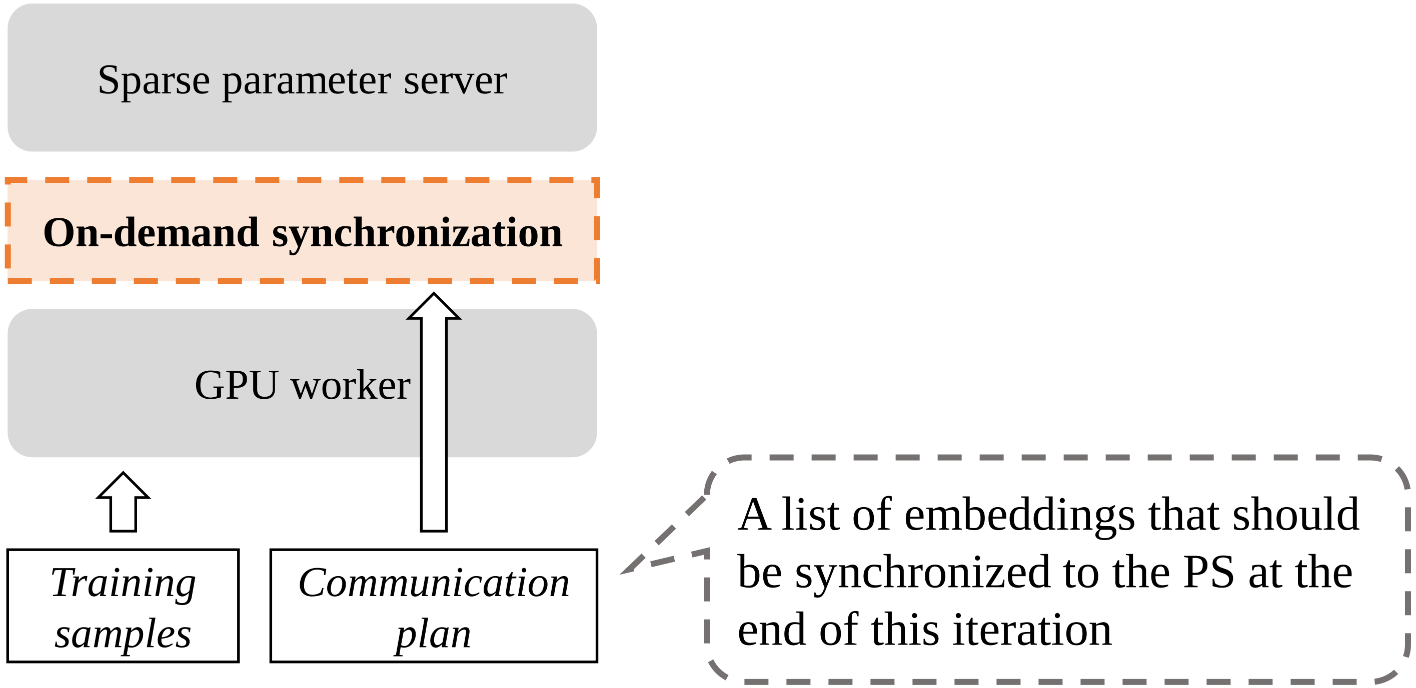
O2: Performing on-demand embedding synchronizations

O1: Training as much as possible in-cache embeddings

Location-aware Inputs Partition



On-demand Synchronization



Communication plan at iteration i = *Embedding dependencies* at iteration $i + 1$

Synchronized embeddings

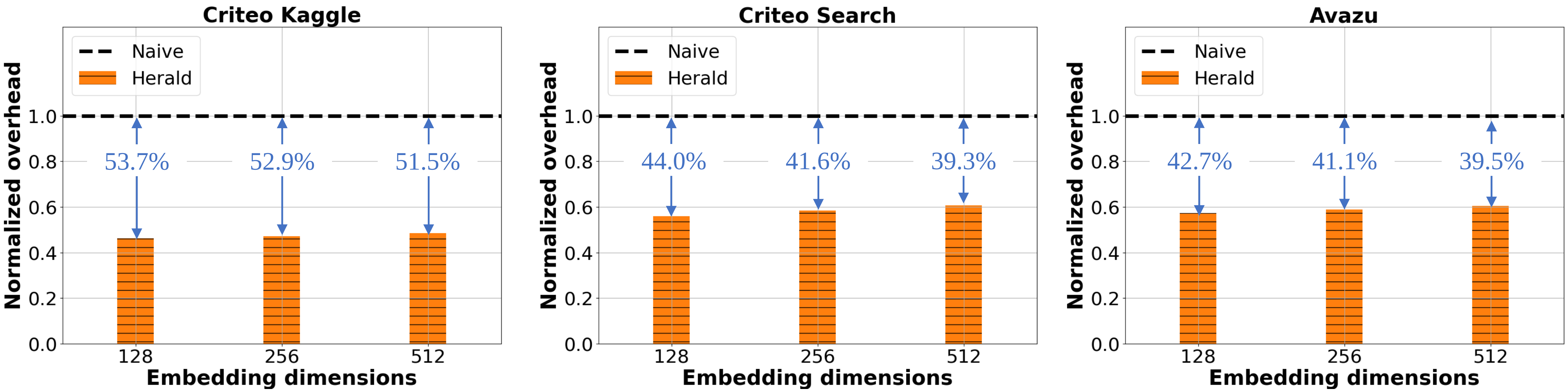
Required embeddings

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

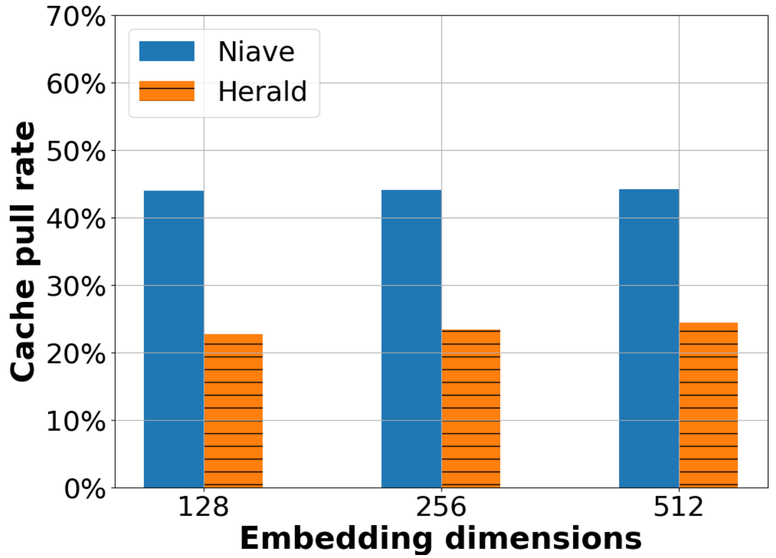
Overall Performance

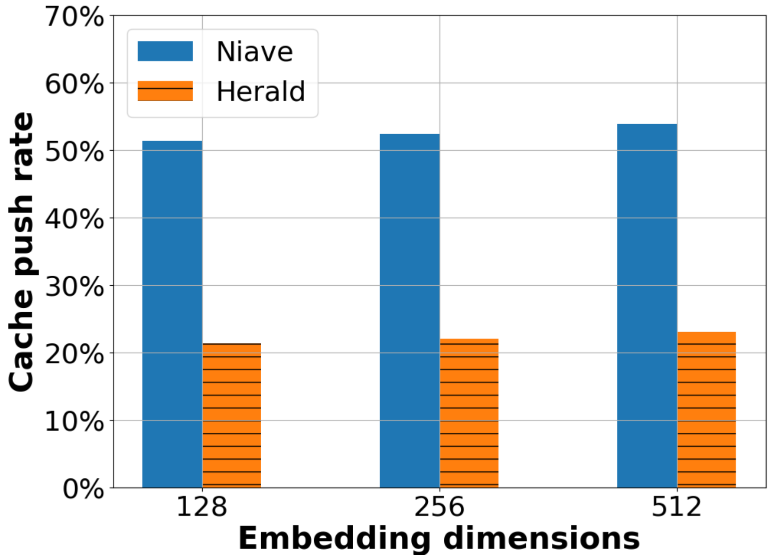


Herald can significantly reduce cache overhead compared to a naïve manner

Performance Breakdown

Dataset: Criteo Kaggle





Herald reduces cache pull rate by up to 48.2% and cache push rate by up to 58.4%

Contribution Breakdown

Optimization	Pull	Push	Overall
Naive	1	1	1
On-demand synchronizations	1	0.84	0.91
Location-aware input partition	0.52	0.85	0.69
Herald	0.52	0.42	0.46

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