Ekko: A Large-Scale Deep Learning Recommender System with Low-Latency Model Update

Chijun Sima[1*], **Yao Fu[2*]**, Man-Kit Sit[2], Liyi Guo[1], Xuri Gong[1], Feng Lin[1], Junyu Wu[1], Yongsheng Li[1], Haidong Rong[1], Pierre-Louis Aublin[3], Luo Mai[2]

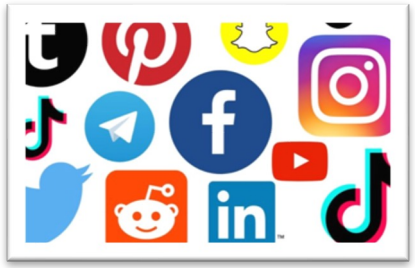
Tencent[1], University of Edinburgh[2], IIJ Innovation Institute[3]

*Co-primary authors

Deep learning recommender systems

Digital Content
2.7 Billion
Monthly Active Users

E-Commerce
2 Billion
Digital Shoppers



Social Media
3.8 Billion
Active Users

Digital Advertising
4.65 Billion
User Targeted

Characteristics of Deep Learning Recommender Systems (DLRSs):

- **Billions** of **global online users**
- **Service-Level-Objectives (SLOs)**

Data source: How NVIDIA Supports Recommender Systems feat. Even Oldridge, Stanford MLSys Seminar

The DLRS architecture

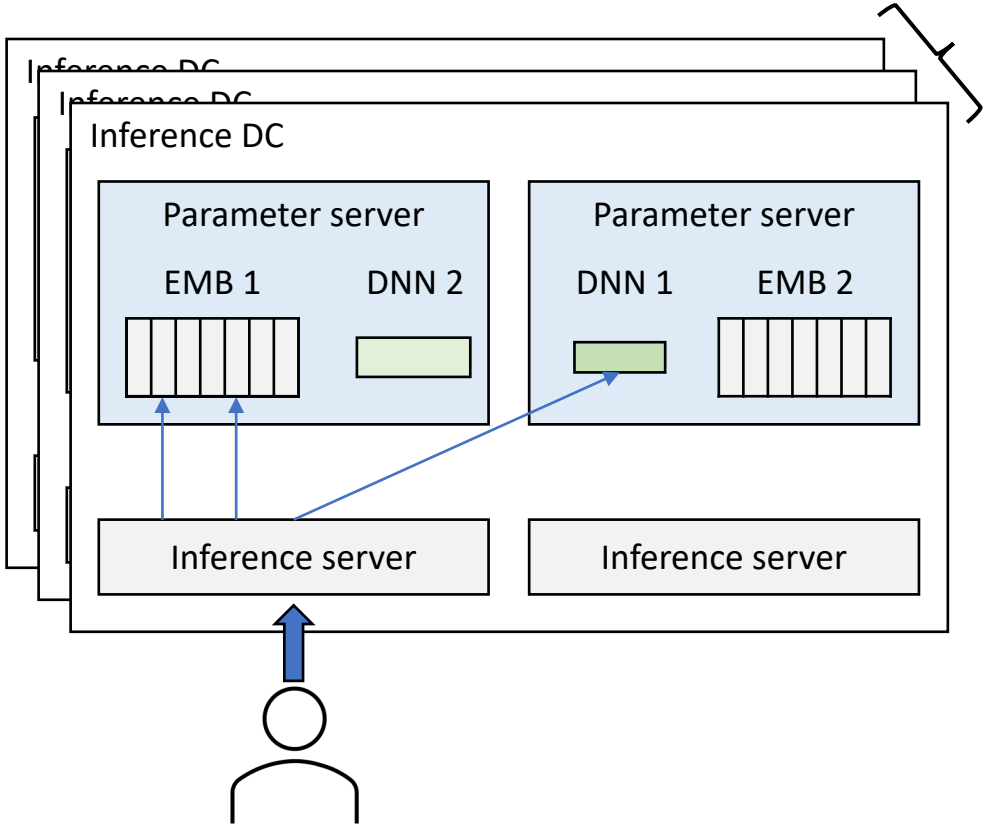
Models in a DLRS

- Embedding Table (EMB)
  - User & item embeddings
- Deep Neural Network (DNN)
  - Multi-layer perceptrons, transformers

Model access characteristics

- Massive parameters
  - EMB (GB - TB), DNN (MB - GB)
- Sparse & intensive reads
  - Hot content, hot users

Parameter servers are replicated in geo-distributed inference DCs



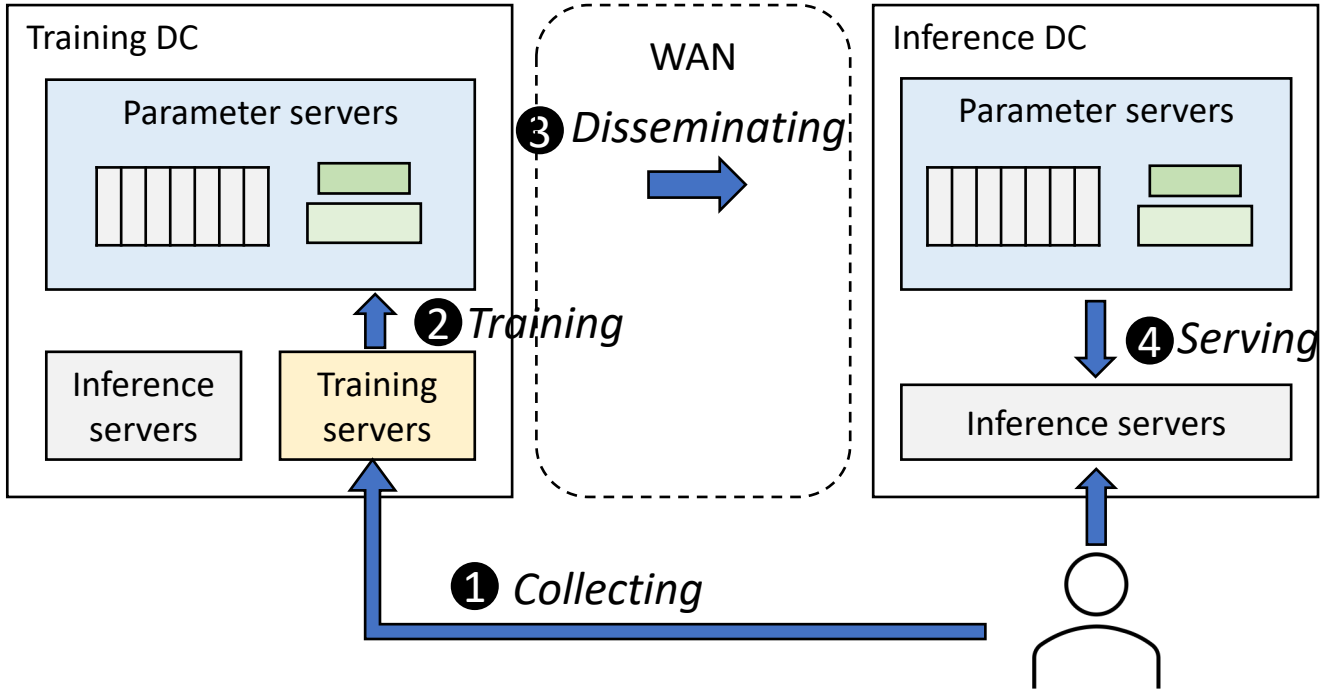
Updating models in a DLRS

Reasons for updating models

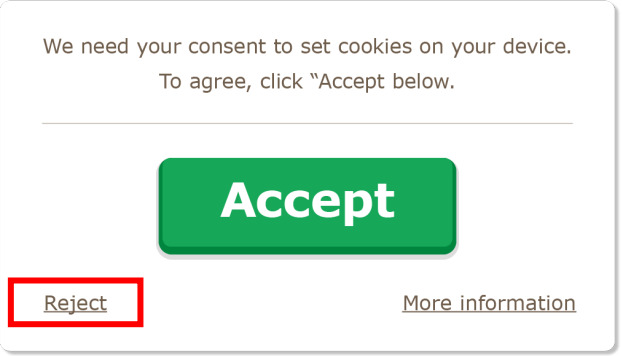
- New content
- New users
- New events

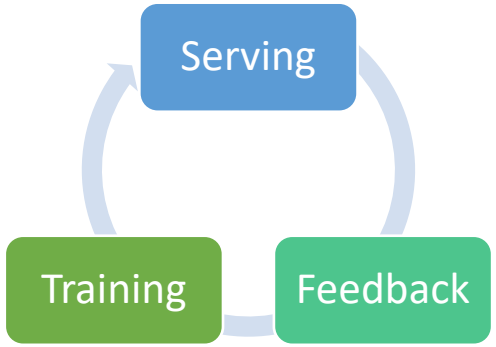
Update process:

1. **Collecting** training data
2. **Training** models online
3. **Disseminating** updated models
4. **Serving** user requests



Low-latency model update





Content freshness

- WeChat, TikTok, Instagram

Anonymous users

- Guest login
- Data protection rules (e.g., GDPR [1])

Online training methods [2]

- Capture user's current interests

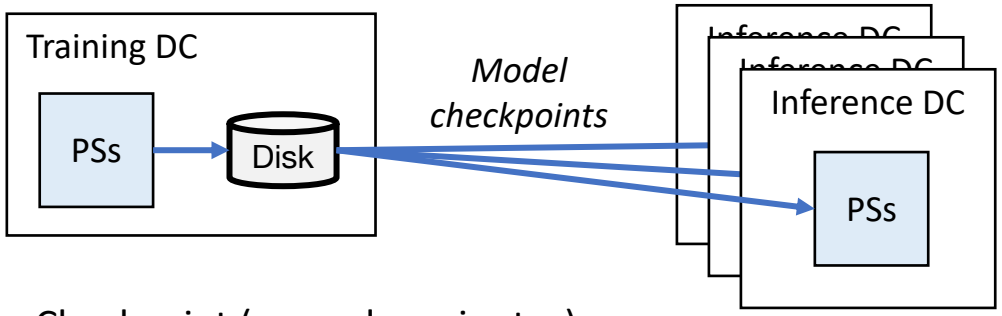
Question: How to support updating TB – PB model parameters in seconds

[1] General Data Protection Regulation, Official Journal of the European Union, 2016
[2] Kraken: Memory-efficient continual learning for large-scale real-time recommendations, SC 2020

Existing model update techniques

SOTA PS [1] – **Checkpoint broadcast**

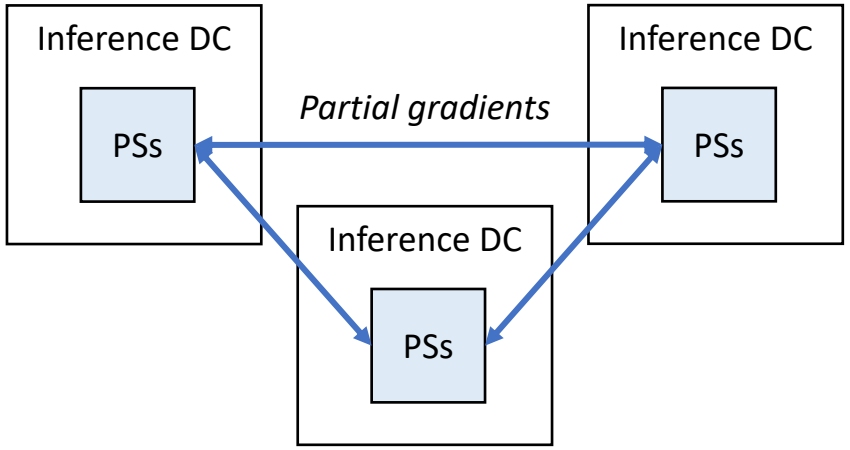
- Multiple long-latency model update steps



1. Checkpoint (seconds - minutes)
2. Validation (minutes - hours)
3. Broadcast (seconds – minutes)

Geo-distributed ML [2] – **Lazy update**

- SLO loss



Issue: Existing techniques compromise latency or SLO performance

[1] Check-N-Run: a checkpointing system for training deep learning recommendation models, NSDI 2022
[2] Gaia: Geo-distributed machine learning approaching LAN speeds, NSDI 2017

Our idea and its associated challenges

Idea: Let the training DC directly send updates to inference DCs

Coordinating massive model updates

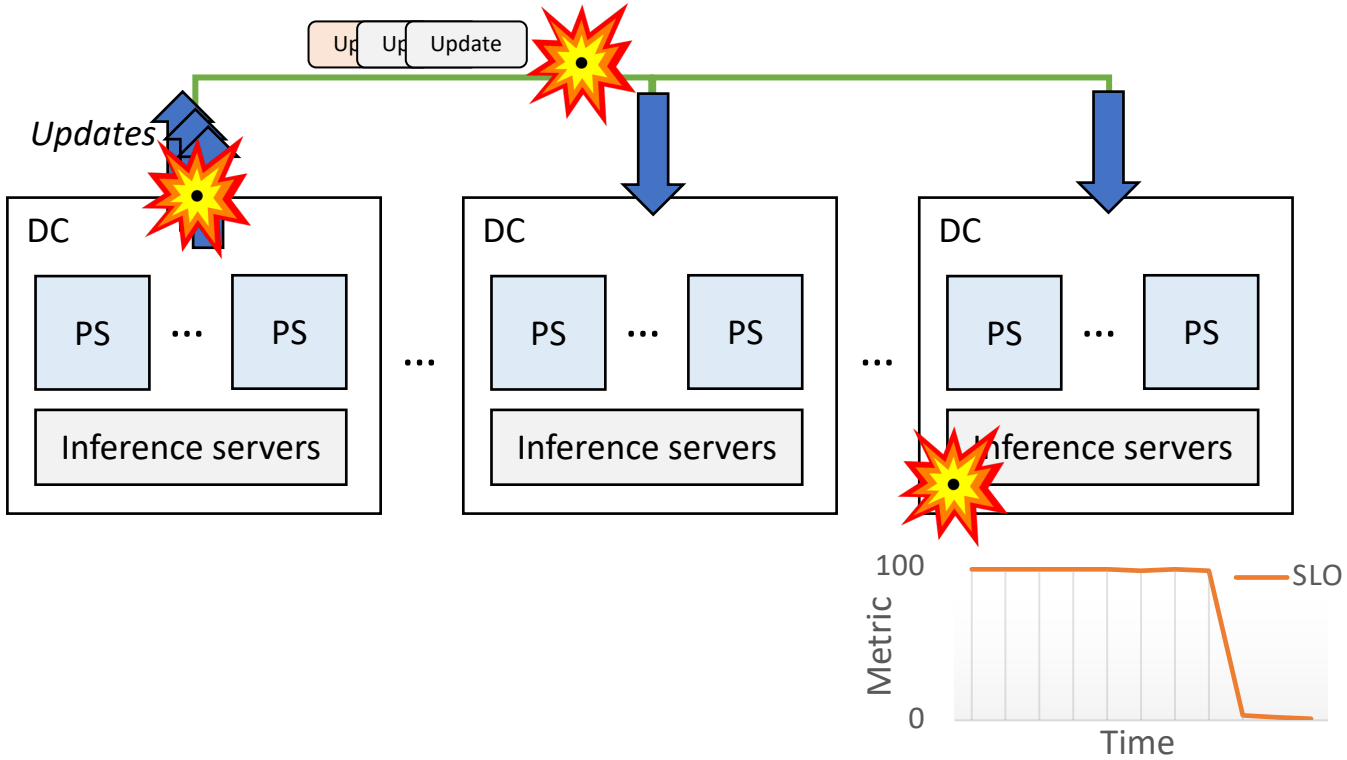
- Leader bottlenecks

Sending model updates over WANs

- Network congestions

Biased model updates

- SLO loss



Ekko: A DLRS with SLO-preserving low-latency model updates

Efficient peer-to-peer (P2P) model update

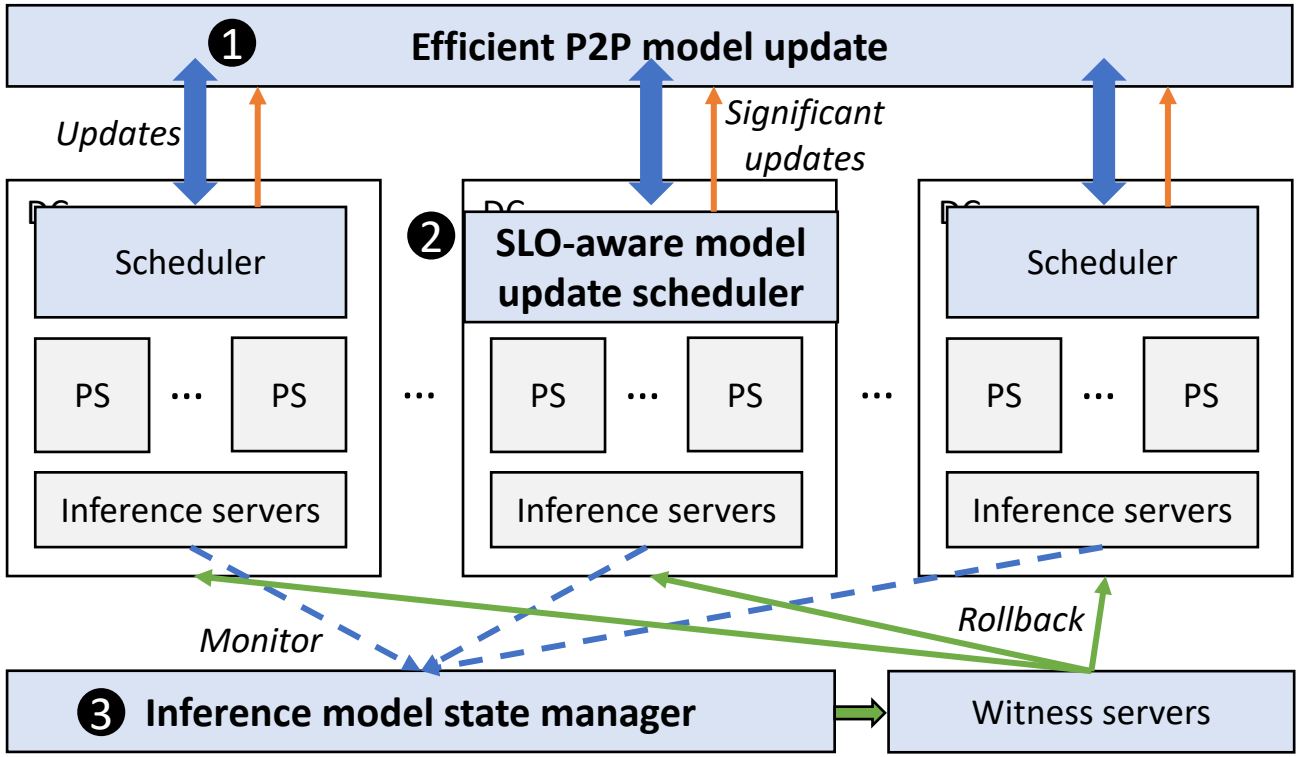
- Utilise more network paths
- Reduce dissemination cost

SLO-aware model update scheduler

- Prioritise significant updates

Inference model state manager

- Monitor and recover model state online

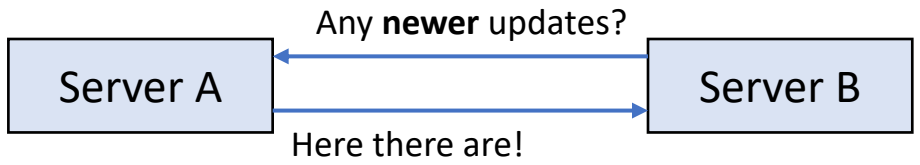


Contribution 1
Efficient P2P Model Update Mechanism

P2P complexity in model synchronisation

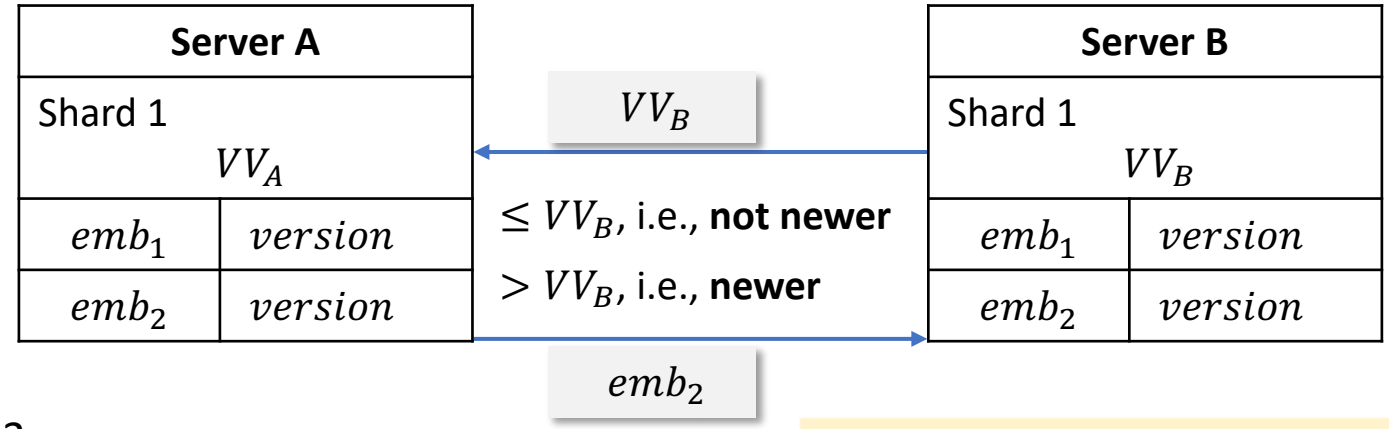
Format:

- A parameter is a *key-value pair*
- Parameters are distributed into shards



How can PSs find **newer** parameter update from each other?

- **P2P synchronisation** [1, 2]: parameter $version$, version vector (VV, shard knowledge)



Details in §4.2, §4.3

N comparisons are not acceptable

N # of parameters per shard $O(100{,}000)$

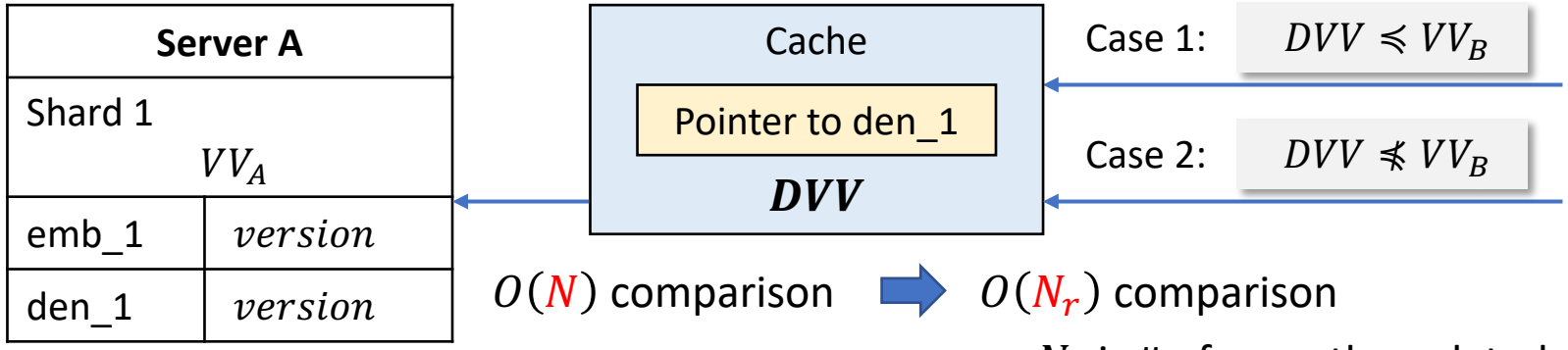
[1] P2P replica synchronization with vector sets, SIGOPS 2007
[2] Concise Version Vectors in WinFS, Distributed Computing 2005

Parameter update cache

Observation: Only a small portion of "hot" parameters are touched (<1% per minute)

Idea: Cache which parameters have been updated recently

Key design: Dominator Version Vector (DVV), which summarises all parameters **NOT** in the cache



$O(N)$ comparison ➡ $O(N_r)$ comparison

N_r is # of recently updated parameters

$$N_r \ll N$$

In production,

- **0.13-0.2%** parameters are kept in caches
- **99.4%** cache hit ratio

Cache maintenance algorithms in § 4.4.1

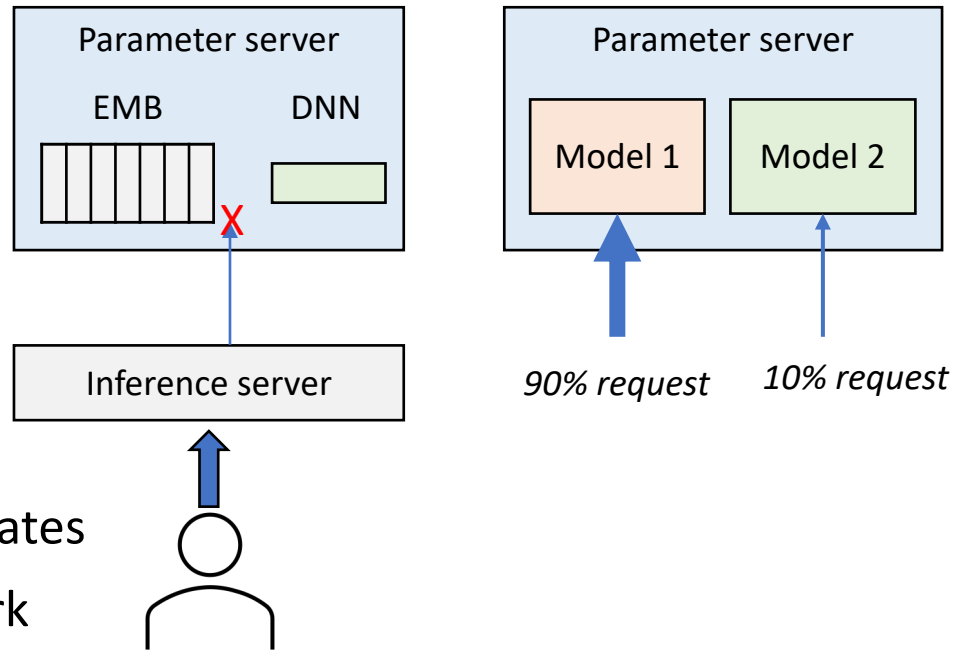
More optimisations: i.e., shard version, WAN-optimisation in paper

Contribution 2
SLO-aware Model Update Scheduler

SLO-critical model updates

SLO-critical updates:

- Newly created embedding items are critical
- Large gradients have significant impact [1, 2]
- Popular models can influence large number of users



Problem: Congested networks delay SLO-critical model updates

i.e., O(100) GB/s updates vs. O(100) Mbps network

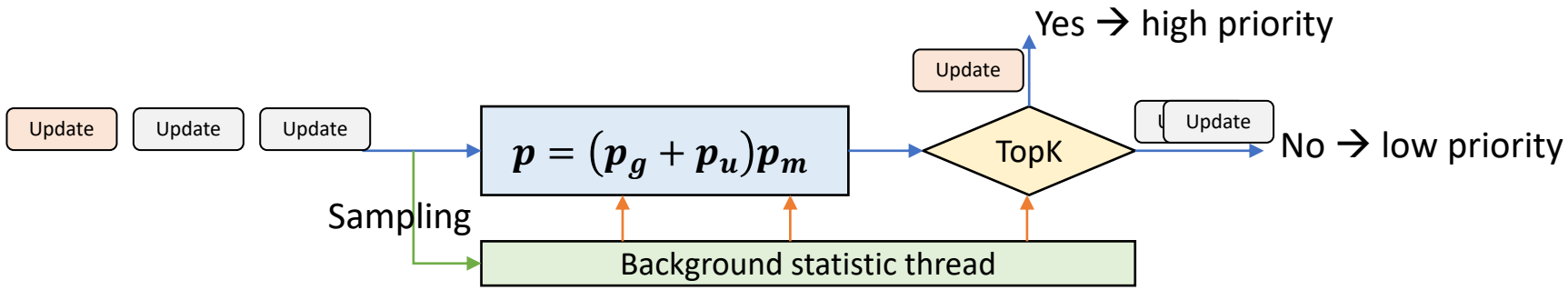
[1] Gradient Compression Supercharged High-Performance Data Parallel DNN Training, SOSP 2021
[2] Gaia: Geo-distributed machine learning approaching LAN speeds, NSDI 2017

SLO-aware model update scheduler

Idea: Prioritising SLO-critical updates

Key priority metrics:

- Freshness: $p_u = +\infty$ when parameter created; otherwise $p_u = 0$
- Update significance: $p_g = |g|/\overline{|g|}$, i.e., normalised gradients of a model
- Model priority: $p_m = c_m / \sum_{i=1}^{M} c_i$, i.e., percentage of requests to a model m



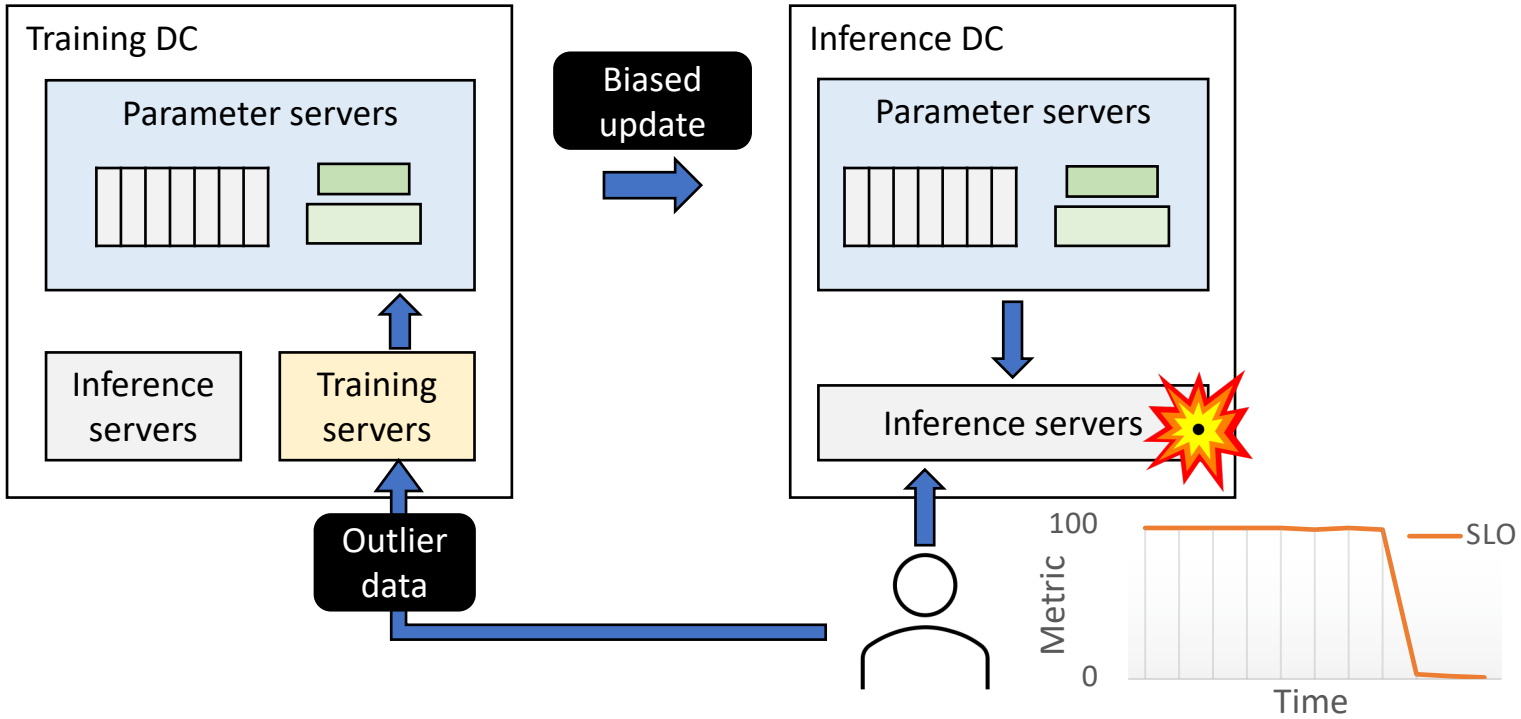
Ekko supports UDF for custom metrics (e.g., update count, update interval, parameter positions)

Contribution 3
Inference Model State Manager

Biased model updates

Causes of biased updates

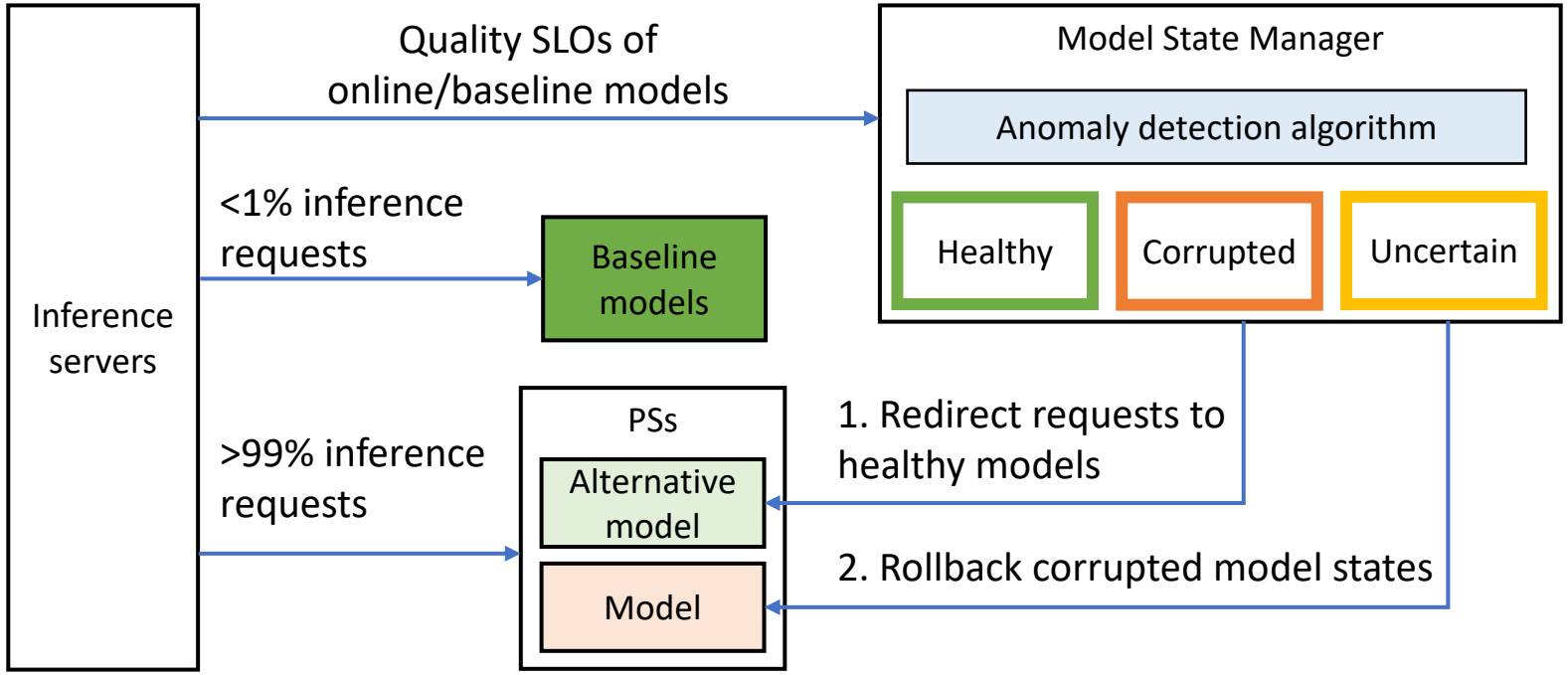
- Gradient overflow
- Data outlier



Problem: biased updates can be detrimental to SLOs

Inference model state manager

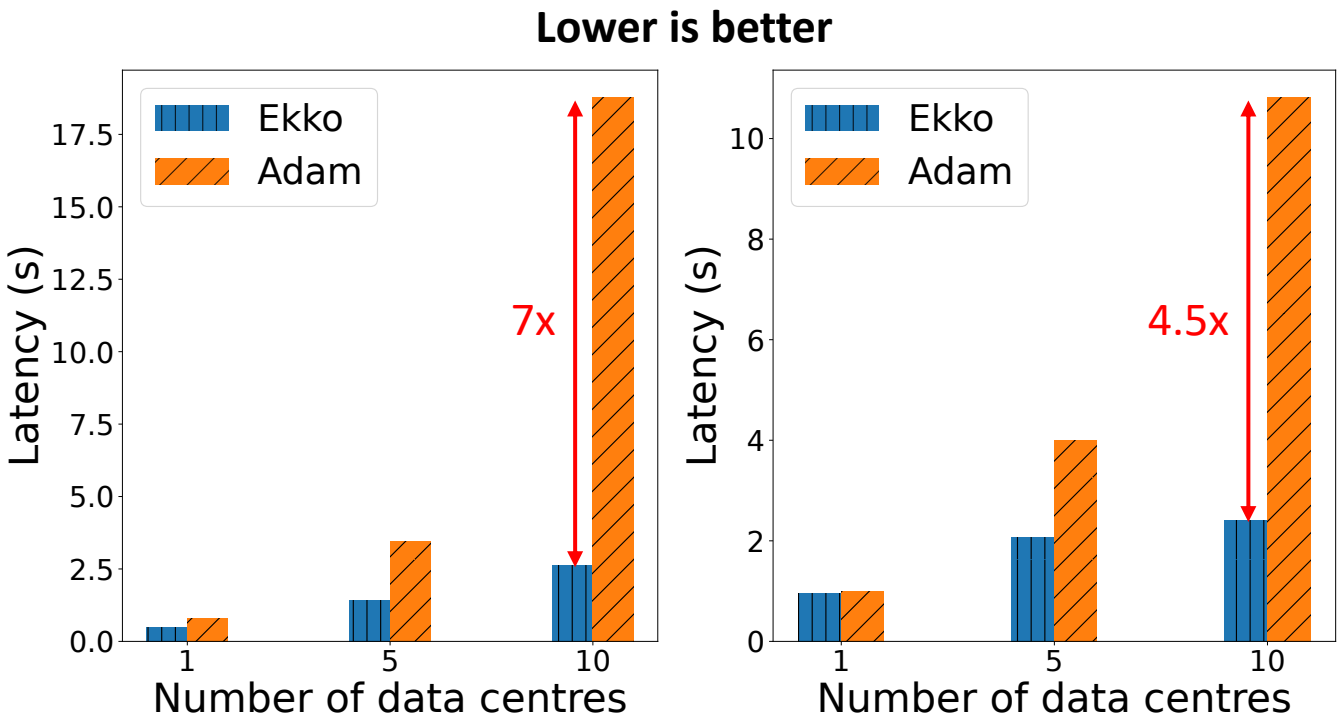
Idea: using baseline models as a reference for model healthiness



Evaluation

End-to-end model update latency

- Setup
  - 30 servers, 3 servers per DC
- Adam
  - A PS with high-performance model update synchronisation
- 7x faster
  - Avoiding leader bottleneck
  - Utilising more network paths
  - Accelerated P2P

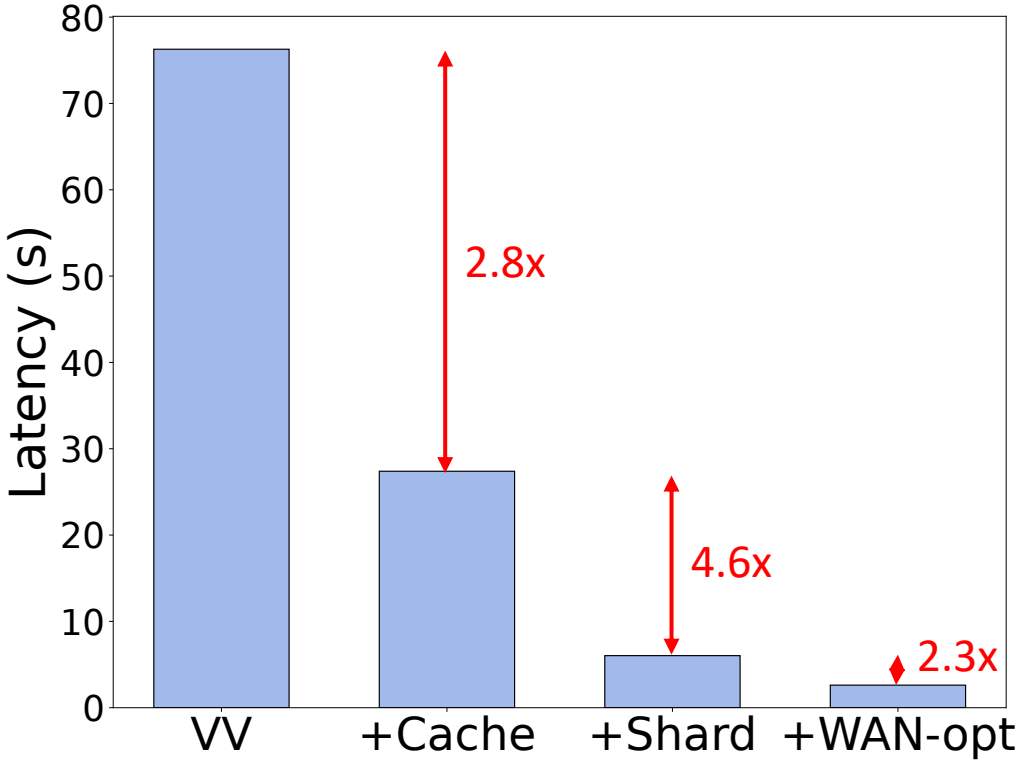


(a) Production workload

(b) Criteo workload

Model update latency breakdown

- Setup
  - 10 DC
  - Production workload
- 29.3x lower latency
  - Model update cache
  - Shard version (details in §4.4.2)
  - WAN-optimisation (details in §4.5)



Large-scale Production Deployment

Improve the model update latency by up to **100x.**

# data centres	# servers	# models	Size of parameters	Model update per second	Avg. latency (inter-DC)	Avg. latency (intra-DC)
6	4,600	100s	40 TB	212 GB/s	**2.4 s**	**0.7 s**

Latency of prior systems (similar to [1]):
10 minutes

Benefits of our designs

- Low-latency model updates: 1.30 – 3.82% SLO improvement
- SLO-aware model update scheduler: avoiding 2.32% SLO dropping
- Fast recovery: 6.4 seconds to rollback 113 GB parameters

[1] https://www.tensorflow.org/api_docs/python/tf/train/Checkpoint

Summary

- Ekko: support low-latency model updates without compromising SLOs
  - Efficient P2P model update
  - SLO-aware model update scheduler
  - Inference model state manager
- Many potential applications beyond DLRSs

Thank You — Any Questions?

Chijun Sima*
chijunsima@tencent.com

Yao Fu*
Y.Fu@ed.ac.uk

*Co-primary authors

Tencent 腾讯

THE UNIVERSITY of EDINBURGH

IIJ Internet Initiative Japan