THE UNIVERSITY *of* EDINBURGH
informatics

WePS: Enabling Low-latency Giant Model Replication in Geo-distributed Parameter Servers

Chijun Sima[1*], **Yao Fu[2*]**, Man-Kit Sit[2], Feng Lin[1], Pierre-Louis Aublin[3], Luo Mai[2]

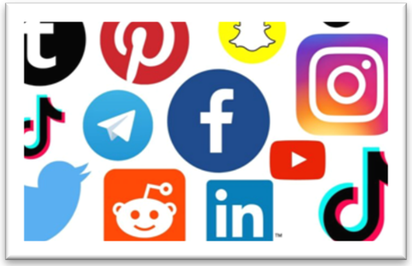
Tencent[1], University of Edinburgh[2], IIJ Innovation Institute[3]

*Co-primary authors

Recommendation 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 recommendation systems:

- **Billions** of **global online users**
- **Latency-sensitive Service-Level-Objectives** (e.g., latency of making new contents visible)

Geo-distributed parameter servers

Parameter Server (PS)

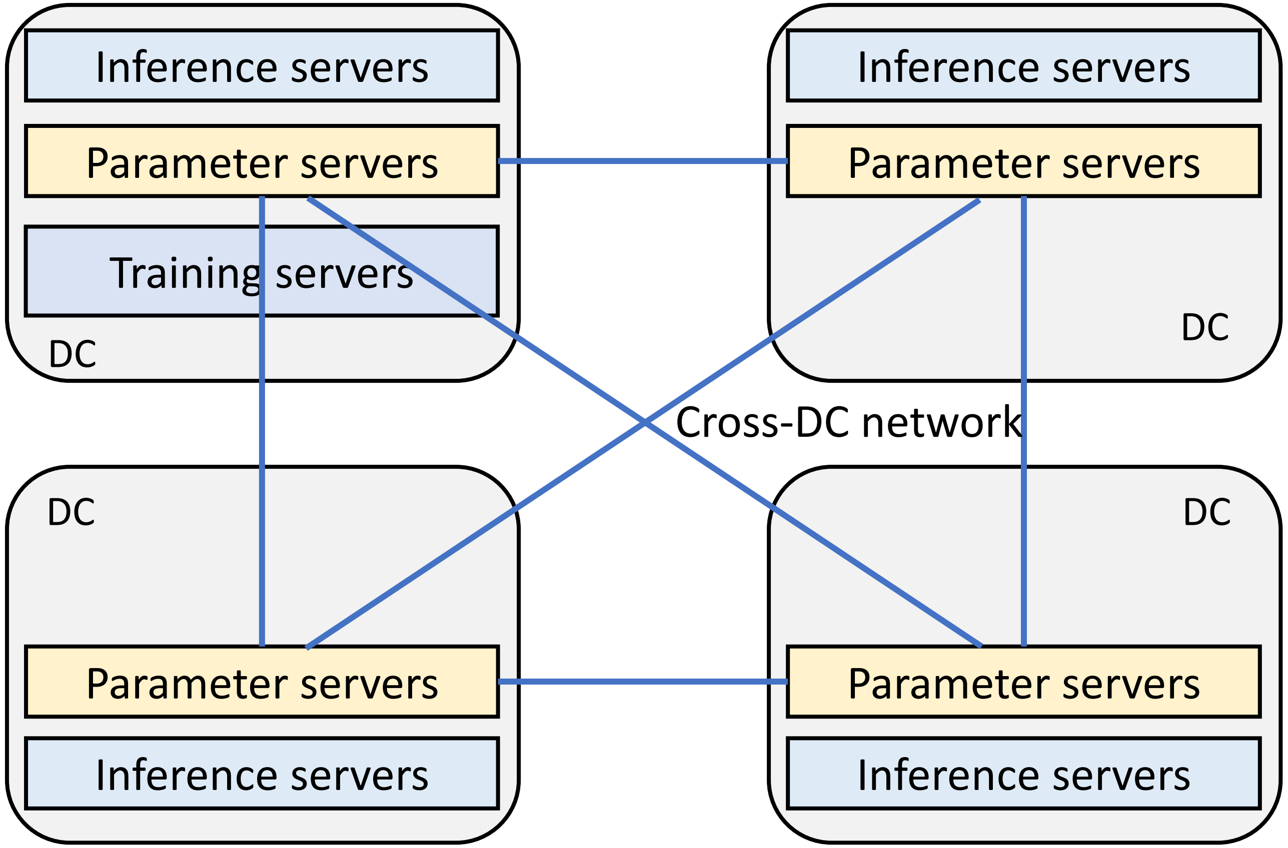
- Embedding table & neural networks

Model update (W = W + lr*grad)

- Collect all data in a DC for best accuracy
- Training servers compute **gradients** which correct the parameters in PS

PS are **replicated** across Data Centres (DCs)

- Minimising model inference latency
- Cross-DC networks have limited bandwidth (e.g., 100 – 1000 Mbps [1])



[1] Gaia: geo-distributed machine learning approaching LAN speeds, NSDI 2017

Gigantic models and massive model updates

Gigantic models (> 1 TBs) are emerging in recommendation systems

- Embedding tables increased **100x** every year (production data)
- Neural networks increased **10x** every year [1]

Massive model updates (> 250 million/second) are in needs

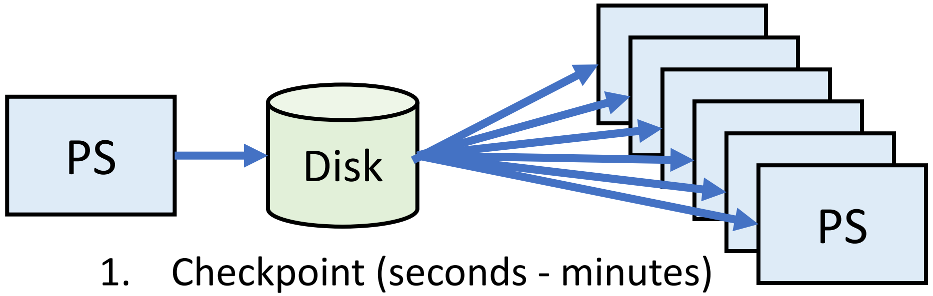
- Many new users, e.g., GDPR leads to massive anonymous users
- Massive new contents, e.g., TikTok, YouTube

[1] https://openai.com/blog/ai-and-compute/

Problems of existing PS systems

PS [1] and BytePS [2] – **Checkpoint broadcast**

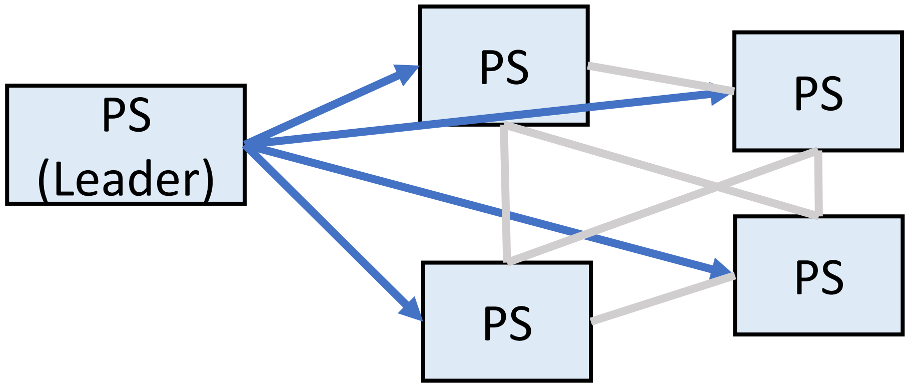
- Multiple long-latency model update steps



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

Adam [3] – **In-memory replication**

- Leader bottleneck
- Under-utilise network paths
- Eventual consistency hurts SLOs



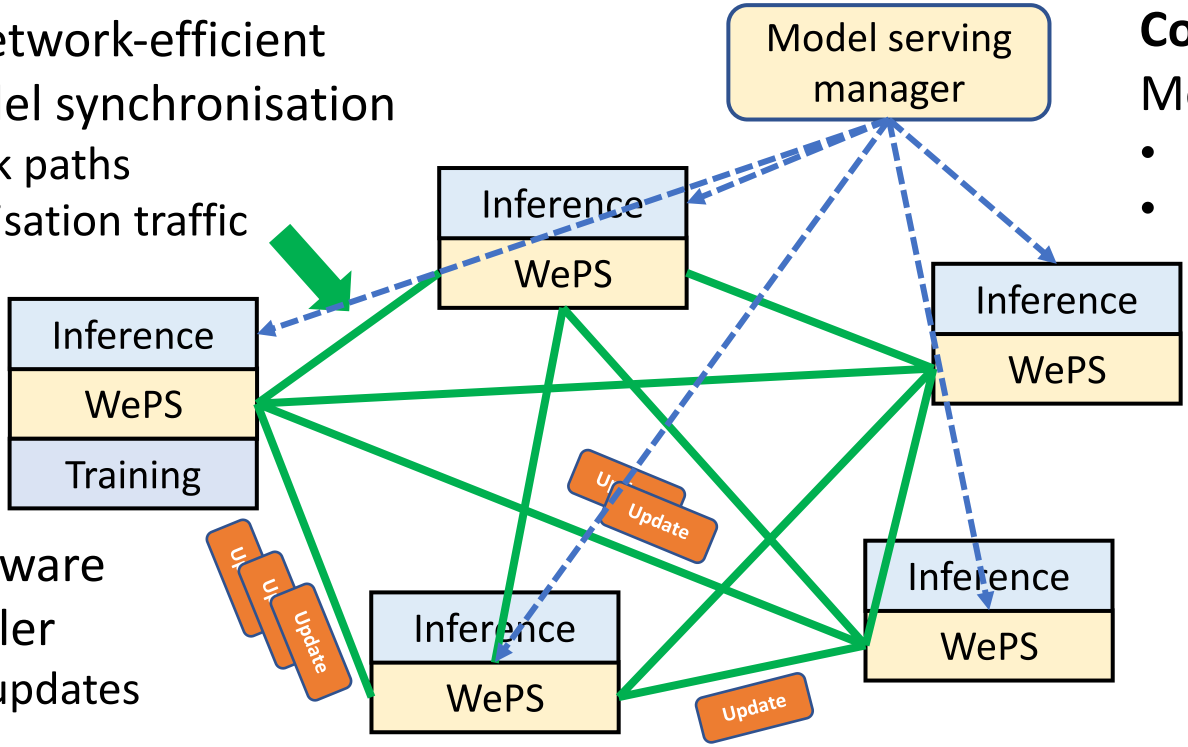
Our goal: Supporting high-throughput, low-latency parameter update for gigantic model replicas

[1] Scaling distributed machine learning with the parameter server, OSDI 2014
[2] A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters, OSDI 2020
[3] Project Adam: building an efficient and scalable deep learning training system, OSDI 2014

WePS overview

Contribution 1: Network-efficient decentralised model synchronisation

- Utilise all network paths
- Reduce synchronisation traffic



Contribution 3:
Model serving manager

- Evaluate & recall model online
- Recover failure online

Contribution 2: SLO-aware model update scheduler

- Prioritise significant updates

How to reduce synchronisation latency?

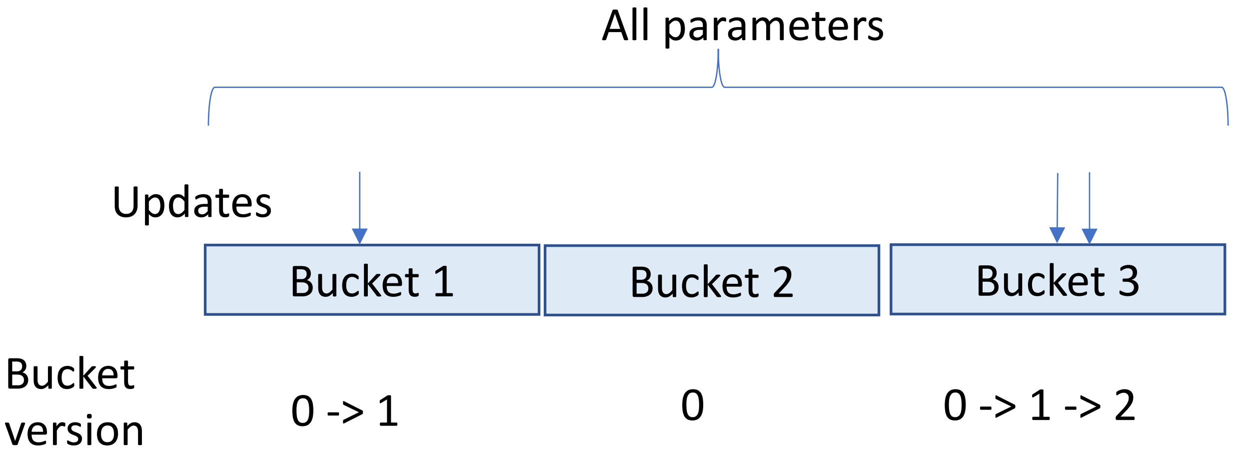
Observation: Only a small portion of “hot” parameters are touched (<1% per minute)

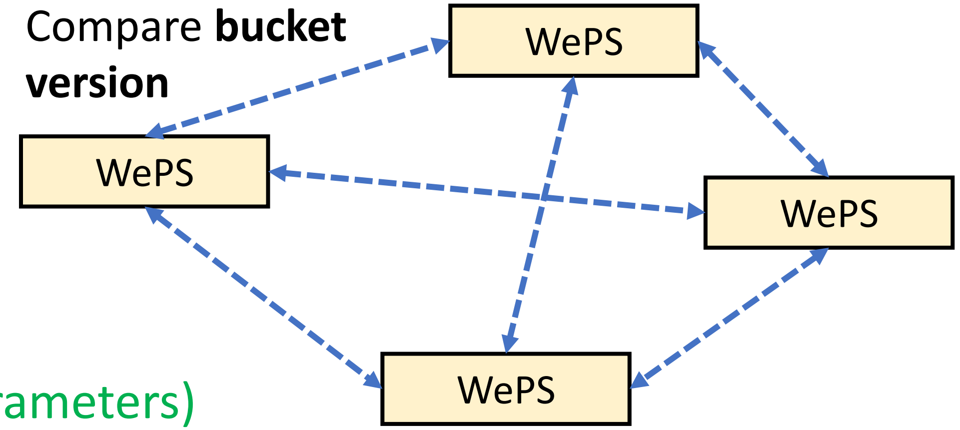
Idea: Enabling PS to compare parameters and only synchronise updated parameters

Challenge: Compare parameters all parameters is very expensive -- O(#Parameters)

Benefits

- Use all network paths
- Reduce network traffic





Compare bucket versions incurs O(#Buckets) << O(#Parameters)

Can we further reduce synchronisation latency?

Problem: Find updated parameters in a bucket is expensive (up to 1M parameters per bucket)

Idea: Cache which parameters have been updated

Updates

Update cache	Update cache	Update cache
Bucket 1	Bucket 2	Bucket 3

Cache result after 2 updates

Parameter Name	Parameter Weight	Timestamp
DenseLayer09	[0.6, 0.9, 9.6, 0.3]	1
Embedding01	[0.1, 0.7]	4

Cache size is usually 100 – 1000 (<< 1M)

Details in incoming report
- Cache retirement policy
- Cache update policy

How to handle big model updates?

Problem: Big model updates (e.g., GBs) take long time to complete and affect model serving result

Idea: Prioritise significant parameter update (Why? Only significant update largely change model serving result).

Significance += $\sum_{i=1}^{l} |gradient[i]|$

Requester

Parameter Name	Parameter Weight	Timestamp	Significance
Transformer01	[0.6, 0.9, ……, 0.3]	1	**3.6**
Transformer02	[0.1, 0.7, ……, 0.1]	4	**6.2**

Difference = 4.9

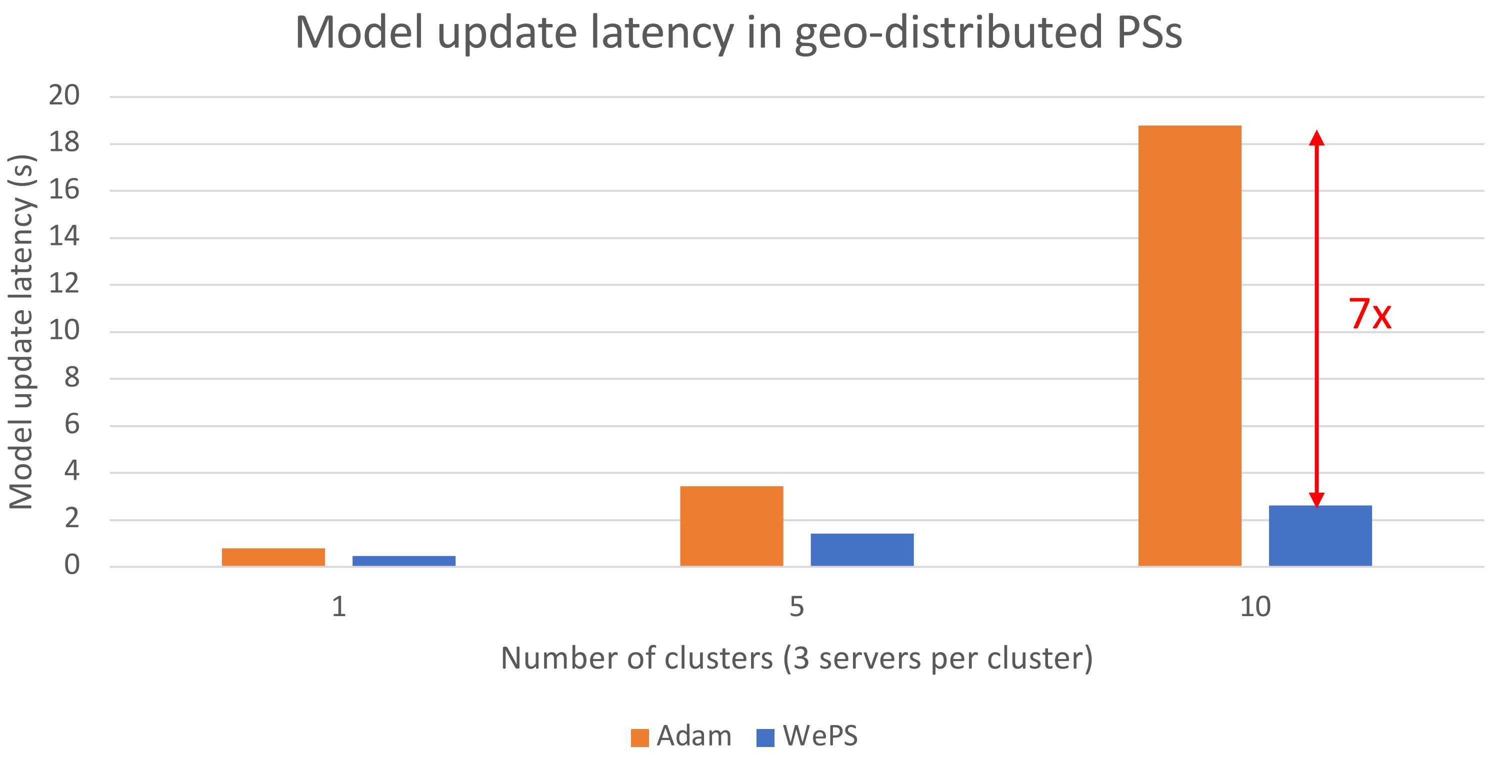
Responder

Parameter Name	Parameter Weight	Timestamp	Significance
Transformer01	[0.4, 0.1, ……, 0.5]	5	**8.5**
Transformer02	[0.2, 0.3, ……, 0.8]	7	**9.2**

Details in incoming report
- Multi-hop synchronisation
- Multi-model coordination

Test-bed Experiments

Lower the better



30 servers (10 clusters), 5 TB model, Production model update workload

THE UNIVERSITY of EDINBURGH informatics

Large-scale Production Deployment

Improve the synchronisation latency by up to **100x.**

# replicas	# machines	# models	Size of parameters	Model update per second	Avg. latency (inter-DC)	Avg. latency (intra-DC)
6	1986	100	18 TB	250 M/s	**4.5 s**	**2.1 s**

Latency of existing PS systems [1, 2]:
10 minutes

System availability:

- Model inference > 99.999%
- Model update > 99.9999%

[1] Scaling distributed machine learning with the parameter server, OSDI 2014
[2] A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters, OSDI 2020

Summary

- Geo-distributed recommendation systems must support gigantic models and massive model updates
- WePS: A system for supporting low-latency updates towards geo-distributed gigantic models
  - Network-efficient decentralised model synchronisation
  - SLO-aware model update scheduler
  - Online model serving manager
- Many future directions
  - Support emerging storage hardware (e.g., persistent memory)
  - Support multi-modalities deep learning models (e.g., MoE)

Thank You — Any Questions?

Yao Fu
Y.Fu@ed.ac.uk