Adaptive Parameter Servers

Ph.D. Thesis Defense, TU Berlin

Alexander Renz-Wieland
December 16, 2022
Machine Learning is Everywhere

«A crowded urban yet green scenery with multiple people, one looking at their smartphone, with one car nearby, digital art », DALL·E
Scale Is Key for Quality

Model

Training data

Training
Scale Is Key for Quality

Training data
larger $\sim$ better

Model
larger $\sim$ better

Training

Scale is key for quality
Scale Is Key for Quality

Model
larger ~ better

Training data
larger ~ better

Scale is key for quality
To keep up: distributed training
Distributed Training Has Become a Necessity

Parameter server (co-located)

node 1

Training data

Parameter server

Model parameters

Remote access

Replica synchronization

node 2

node 3
Distributed Training Has Become a Necessity

Parameter server (co-located)

Training data

node 1

node 2

node 3
Distributed Training Has Become a Necessity

- Parameter server (co-located)
- Node 1
- Node 2
- Node 3
- Training data
- Model parameters
- Remote access
- Replication synchronization
Parameter Servers Facilitate Distributed Training

- Parameter server (co-located)
- Node 1
- Node 2
- Node 3
- Training data
- Model parameters
- Remote access
- Replica synchronization
Parameter Servers Facilitate Distributed Training

Parameter server (co-located)

node 1

node 2

node 3

Training data

Parameter server

Model parameters

Remote access

Replica synchronization
How Fast Is Distributed Training?

Training knowledge graph embeddings:

- Classic PS
- Static full replication

7x speed-up over single node

Run time (minutes)

Model quality

ComplEx-500 on Wikidata5M, 8 nodes x 32 threads, 100 Gbit/s InfiniBand
How Fast Is Distributed Training?

Training knowledge graph embeddings:

ComplEx-500 on Wikidata5M, 8 nodes x 32 threads, 100 Gbit/s InfiniBand
The problem: 
Communication overhead
Communication Overhead in a Classic PS

Approach: partition the parameters to nodes

- Parameter server (co-located)
- Node 1: Partitioned parameters
- Node 2: Partitioned parameters, not locally available
- Node 3: Partitioned parameters, not locally available

- Training data
- Parameter server
- Model parameters
- Remote access
- Replicas synchronization

Problem: remote parameter accesses
Communication Overhead in a Classic PS

Approach: partition the parameters to nodes

Problem: remote parameter accesses
Approach: replicate all parameters to all nodes
Approach: replicate all parameters to all nodes

Problems: limited model size and communication intensive
Sparse Parameter Access

Training data

step 1 ➔ step 2 ➔ step 3 ➔ ...

They can swim

Sparse access:
each step accesses only a subset of all parameters

Challenge: dynamic access pattern

Sparse access is common in:
• Knowledge graph embeddings
• Some graph neural networks
• Natural language processing
• Click-through prediction
• Recommender systems
Sparse Parameter Access

«They can swim »

Sparse access: each step accesses only a subset of all parameters
Sparse Parameter Access

Training data «They can swim»

Sparse access: each step accesses only a subset of all parameters

Challenge: dynamic access pattern

Sparse access is common in:
• Knowledge graph embeddings
• Some graph neural networks
• Natural language processing
• Click-through prediction
• Recommender systems
Sparse Parameter Access

Plot showing parameter access replicated at each node across different steps.

Step 1: node 1, node 2, node 3
Step 2: node 1, node 2, node 3
Step 3: node 1, node 2, node 3

Challenge: dynamic access pattern

Sparse access is common in:

- Knowledge graph embeddings
- Some graph neural networks
- Natural language processing
- Click-through prediction
- Recommender systems
Thesis:
Parameter servers should adapt to the underlying task
Outline: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   - VLDB '20
   - VLDB '21 demo

2. Non-Uniform Parameter Management
   - SIGMOD '22

3. Automatic Adaptivity
   - arXiv '22
   - under review at MLSys '23

Conclusion
Outline: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   How to exploit locality?
   VLDB ’20
   VLDB ’21 demo

2. Non-Uniform Parameter Management
   How to handle diversity?
   SIGMOD ’22

3. Automatic Adaptness
   How to attain ease of use?
   arXiv ’22
   under review at MLSys ’23

Conclusion
• Common principle for reducing communication overhead:

Create and/or exploit parameter access locality
Parameter Access Locality

• Common principle for reducing communication overhead:

  Create and/or exploit parameter access locality

• Common locality techniques:

  • Data clustering
  • Parameter blocking
  • Latency hiding

  } Manually create locality
  } Exploits inherent locality
Problem: Parameter Servers Do Not Support Locality Techniques

- Main limitation: parameter allocation is static

Static parameter allocation
Problem: Parameter Servers Do Not Support Locality Techniques

- Main limitation: parameter allocation is static

Example: latency hiding

Static parameter allocation

- node 1
- node 2
- node 3
Main limitation: parameter allocation is static

Locality techniques need to be implemented with low-level communication primitives

Example: latency hiding

Static parameter allocation

node 1

node 2

node 3
Our Approach: Dynamic Parameter Allocation

Relocate parameters to where they are accessed
Our Approach: Dynamic Parameter Allocation

Relocate parameters to where they are accessed

New primitive to control allocation: \texttt{Localize(parameters)}
Lapse: the First Dynamic Allocation Parameter Server

- Makes relocation simple for applications

- System challenges
  - Parameter relocation
  - Location management
  - Routing
  - Reads and writes during relocation
Lapse: the First Dynamic Allocation Parameter Server

- Makes relocation simple for applications

- System challenges
  - Parameter relocation
  - Location management
  - Routing
  - Reads and writes during relocation

- Key properties
  - Location transparency
  - Sequential consistency
Can Dynamic Allocation Improve Efficiency?

With data clustering or parameter blocking:

Efficient ✓

With latency hiding:

Relative efficient for some tasks, but not all

Matrix factorization (uniform matrix), 8 nodes x 8 threads, 100 Gbit/s InfiniBand
Can Dynamic Allocation Improve Efficiency?

With data clustering or parameter blocking:

Efficient ✓

With latency hiding:

Relatively efficient for some tasks, but not all

Matrix factorization (uniform matrix), 8 nodes x 8 threads, 100 Gbit/s InfiniBand

Word embeddings, 8 nodes x 4 threads, 10 Gbit/s Ethernet
Recap Part 1: Dynamic Parameter Allocation

• Existing parameter servers cannot exploit locality

• **Dynamic allocation** enables support

• Efficiency:
  • With *data clustering* and *parameter blocking*: excellent
  • With *latency hiding*: varied
Outline: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   - How to exploit locality?
   - VLDB '20
   - VLDB '21 demo

2. Non-Uniform Parameter Management
   - How to handle diversity?
   - SIGMOD '22

3. Automatic Adaptivity
   - How to attain ease of use?
   - arXiv '22
   - under review at MLSys '23

Conclusion
Dynamic Allocation Can Be Inefficient

Knowledge graph embeddings

8 nodes x 32 threads, 100 Gbit/s InfiniBand
Problem: Non-Uniform Parameter Access

Parameter access frequency is often non-uniform: different parameters have different access patterns

One source: Skew
Problem: Non-Uniform Parameter Access

Parameter access frequency is often non-uniform: different parameters have different access patterns

One source: **Skew**

Another source: **Sampling**

direct access
and
sampling access
Problem: Non-Uniform Parameter Access

Parameter access frequency is often non-uniform: different parameters have different access patterns.

One source: *Skew*

Another source: *Sampling*

Not in this talk:
- direct access
- sampling access
Skew is a challenge for parameter servers. Parameter servers use one technique for all parameters. Dynamic allocation (latency hiding) and replication are two techniques used by parameter servers. However, problem arises because no technique is efficient for all access patterns.
Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters

Dynamic allocation (latency hiding) parameter server

Replication parameter server

Diagram:
- Dynamic allocation (latency hiding)
- Replication

Parameters:
- Hot spots
- Long tail

Parameters: 0B, 2B, 4B
Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters

Dynamic allocation (latency hiding) parameter server

Replication parameter server

Problem: no technique is efficient for all access patterns
Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters

Dynamic allocation (latency hiding) parameter server

Replication parameter server

Problem: no technique is efficient for all access patterns
Our Approach: Multi-Technique Parameter Management

- In general: pick a suitable technique **per parameter**
Our Approach: Multi-Technique Parameter Management

- In general: pick a suitable technique per parameter
- **NuPS**: dynamic allocation + replication
Can Non-Uniform Parameter Management Improve Efficiency?

Knowledge graph embeddings

Matrix factorization

NuPS was efficient on multiple tasks.

8 nodes x 32 threads, 100 Gbit/s InfiniBand
Recap Part 2: Non-Uniform Parameter Management

- Skew is common, and a challenge for single-technique parameter servers

- Parameter servers can improve efficiency by adapting management techniques to access patterns
Outline: Adaptive Parameter Servers

Motivation

How to exploit locality?
1. Dynamic Parameter Allocation
   - VLDB '20
   - VLDB '21 demo

How to handle diversity?
2. Non-Uniform Parameter Management
   - SIGMOD '22

How to attain ease of use?
3. Automatic Adaptivity
   - arXiv '22
   - under review at MLSys '23

Conclusion
NuPS’s Performance Depends on Configuration Knobs

Knowledge graph embeddings

Matrix factorization

NuPS (tuned) 5.8x speed-up
Dynamic allocation
Efficient single node

NuPS (tuned) 6.5x speed-up
Dynamic allocation
Efficient single node

Static full replication: out of memory

8 nodes x 32 threads, 100 Gbit/s InfiniBand
NuPS’s Performance Depends on Configuration Knobs

Knowledge graph embeddings

Model quality (filtered MRR)

Run time (minutes)

Matrix factorization

Model quality (RMSE on test data)

Run time (minutes)

NuPS performance depends on configuration knobs.

8 nodes x 32 threads, 100 Gbit/s InfiniBand
Problem: Efficient Parameter Servers Are Complex To Use

Parameter servers: easy to use OR efficient
Problem: Efficient Parameter Servers Are Complex To Use

Parameter servers: easy to use OR efficient
Problem: Efficient Parameter Servers Are Complex To Use

Parameter servers: easy to use OR efficient

Configuration knobs:
1. Initiate relocations at the right time (part 1)
2. Choose a technique for each parameter (part 2)
3. Choose staleness threshold (related work)
Problem: Efficient Parameter Servers Are Complex To Use

Parameter servers: easy to use OR efficient

Configuration knobs:
1. Initiate relocations at the right time (part 1)
2. Choose a technique for each parameter (part 2)
3. Choose staleness threshold (related work)

→ Efficiency requires domain knowledge and tuning
Our Approach: Automatic Adaptation

1. Intent signaling
   A mechanism to pass information about upcoming parameter accesses

2. AdaPS
   A fully adaptive, zero tuning parameter server
Intent Signaling

Intent: a declaration that a worker intends to access a specific set of parameters in a specific time window in the future

Example intent:

« I will access parameters 13 and 16 in batch 2 »
Intent Signaling

Intent: a declaration that a worker intends to access a specific set of parameters in a specific time window in the future

Example intent:

« I will access parameters 13 and 16 in batch 2 »

Primitive:

```
Intent(parameters, start, end)
```
Integrates naturally into the data loader of common machine learning systems:

- **Data Loader Thread(s):**
  - Prepare Batch 1
  - Prepare Batch 2
  - Prepare Batch 3
  - Prepare Batch 4
  - ... (more batches)

- **Training Thread:**
  - Access Batch 1: A
  - Access Batch 2: B
  - Access Batch 3: C
  - Access Batch 4: D
  - ... (more accesses)

- Time (clock):
  - 0
  - 1
  - 2
  - 3
  - 4

- Intents:
  - Intent(A,1,2)
  - Intent(B,2,3)
  - Intent(C,3,4)
  - Intent(D,4,5)
AdaPS: a Fully-Adaptive, Zero-Tuning Parameter Server

Intent challenges:
1. Large quantities
2. Signalled decentrally
3. Signalled ahead of time

AdaPS figures out:
1. Which technique?
2. Where to allocate?
3. When to relocate?
4. Where to replicate?
5. How long to replicate?

Efficient parameter management
AdaPS: a Fully-Adaptive, Zero-Tuning Parameter Server

Intent challenges:
1. Large quantities
2. Signalled decentrally
3. Signalled ahead of time

Efficient parameter management
Intent challenges:
1. Large quantities
2. Signalled decentrally
3. Signalled ahead of time

AdaPS figures out:
1. Which technique?
2. Where to allocate?
3. When to relocate?
4. Where to replicate?
5. How long to replicate?
AdaPS Main Components

1. Automatic choice of technique
   • Collect intents
   • Rule-based decision
AdaPS Main Components

1. Automatic choice of technique
   - Collect intents
   - Rule-based decision

2. Automatic action timing
   - Challenge: timings unknown
   - Learn correct timing, probabilistic approach
AdaPS Main Components

1. Automatic choice of technique
   - Collect intents
   - Rule-based decision

2. Automatic action timing
   - Challenge: timings unknown
   - Learn correct timing, probabilistic approach

3. Efficiency
Can Automatic Adaptation be Efficient?

Knowledge graph embeddings

Matrix factorization

8 nodes x 32 threads, 100 Gbit/s InfiniBand
Can Automatic Adaptation be Efficient?

Knowledge graph embeddings

Matrix factorization

AdaPS was efficient out of the box for many ML tasks.

8 nodes x 32 threads, 100 Gbit/s InfiniBand
Recap Part 3: Automatic Adaptivity

• So far: parameter servers were easy to use OR efficient

• With automatic adaptation: easy to use AND efficient
  • Intent signaling: passes information
  • AdaPS: adapts automatically

• Efficient out of the box
Conclusions: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   - VLDB ’20
   - VLDB ’21 demo

2. Non-Uniform Parameter Management
   - SIGMOD ’22

3. Automatic Adaptivity
   - arXiv ’22
   - under review at MLSys ’23

Conclusion
Conclusions: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   - VLDB ’20
   - VLDB ’21 demo

2. Non-Uniform Parameter Management
   - SIGMOD ’22

3. Automatic Adaptivity
   - arXiv ’22
   - under review at MLSys ’23

Conclusion

Eight computers can be slower than a single one in distributed machine learning training.
Conclusions: Adaptive Parameter Servers

**Motivation**

1. Dynamic Parameter Allocation
2. Non-Uniform Parameter Management
3. Automatic Adaptivity

**Conclusion**

Eight computers can be **slower than a single one** in distributed machine learning training.

Parameter servers can **improve efficiency** for tasks with sparse access by adapting to the underlying task.

- VLDB ’20
- VLDB ’21 demo
- SIGMOD ’22
- arXiv ’22
- under review at MLSys ’23
Conclusions: Adaptive Parameter Servers

Motivation

1. Dynamic Parameter Allocation
   - VLDB '20
   - VLDB '21 demo

2. Non-Uniform Parameter Management
   - SIGMOD '22

3. Automatic Adaptivity
   - arXiv '22
   - under review at MLSys '23

Eight computers can be slower than a single one in distributed machine learning training.

Parameter servers can improve efficiency for tasks with sparse access by adapting to the underlying task.

Zero-tuning adaptation is possible.
Conclusions: Adaptive Parameter Servers

1. Dynamic Parameter Allocation
   - VLDB '20
   - VLDB '21 demo
   - Lapse

2. Non-Uniform Parameter Management
   - SIGMOD '22
   - NuPS

3. Automatic Adaptivity
   - arXiv '22
   - AdaPS
   - under review at MLSys '23

Motivation

Eight computers can be slower than a single one in distributed machine learning training

Parameter servers can improve efficiency for tasks with sparse access by adapting to the underlying task

Conclusion
Conclusions: Adaptive Parameter Servers

1. Dynamic Parameter Allocation
   - VLDB '20
   - VLDB '21 demo

2. Non-Uniform Parameter Management
   - SIGMOD '22
   - NuPS

3. Automatic Adaptivity
   - arXiv '22
   - AdaPS
   - under review at MLSys '23

4. Efficient distributed training is possible, and achievable with limited additional effort

Motivation

Eight computers can be slower than a single one in distributed machine learning training.

Parameter servers can improve efficiency for tasks with sparse access by adapting to the underlying task.

Conclusion

Efficient distributed training is possible, and achievable with limited additional effort.

 VLDB '20
 VLDB '21 demo
 SIGMOD '22
 NuPS
 arXiv '22
 AdaPS
 under review at MLSys '23

Conclusion
The thesis is based on the following publications:


It also draws material from: