Adaptive Parameter Servers

Ph.D. Thesis Defense, TU Berlin

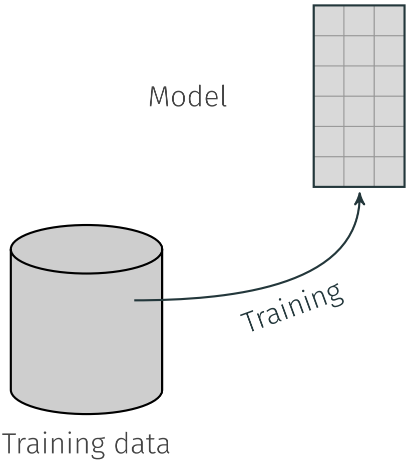
Alexander Renz-Wieland

December 16, 2022

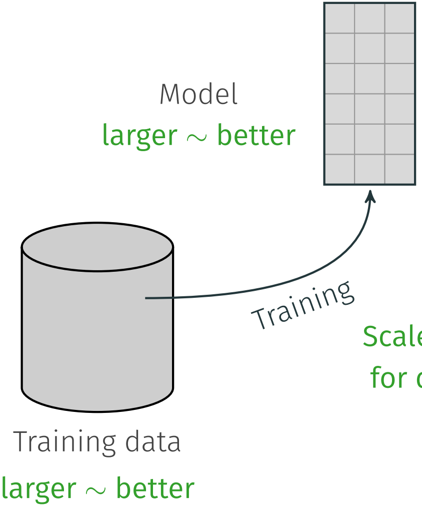
Machine Learning is Everywhere

«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

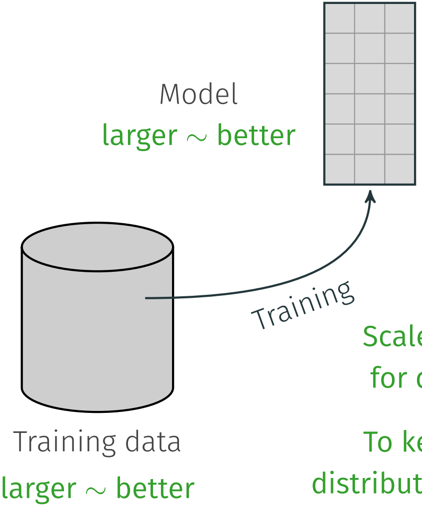


Scale Is Key for Quality



Scale is key
for quality

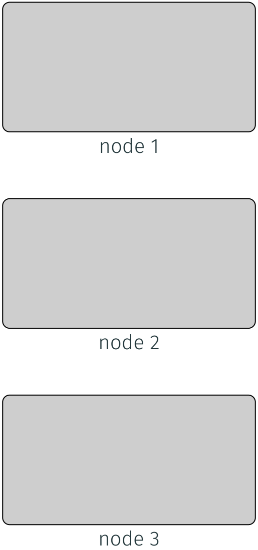
Scale Is Key for Quality



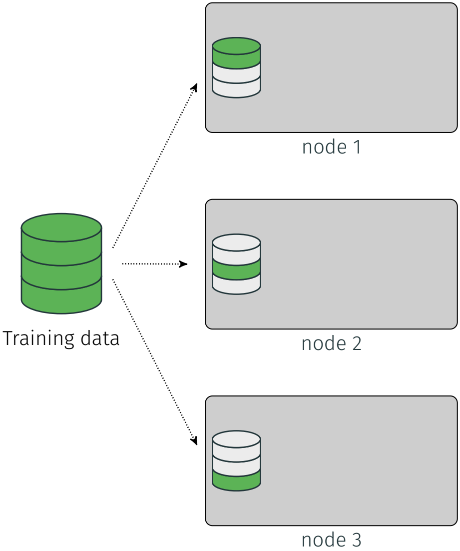
Scale is key
for quality

To keep up:
distributed training

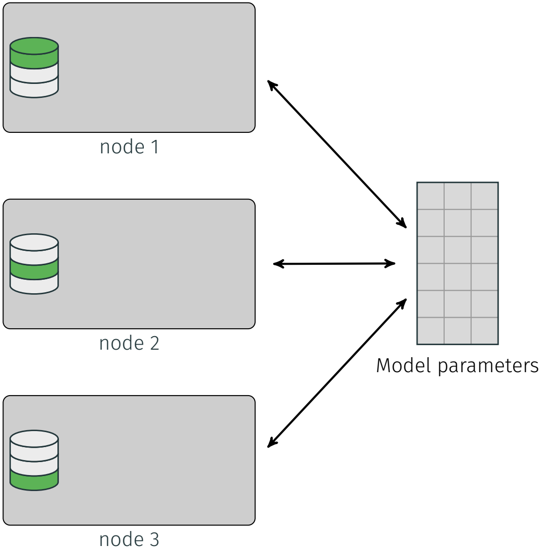
Distributed Training Has Become a Necessity



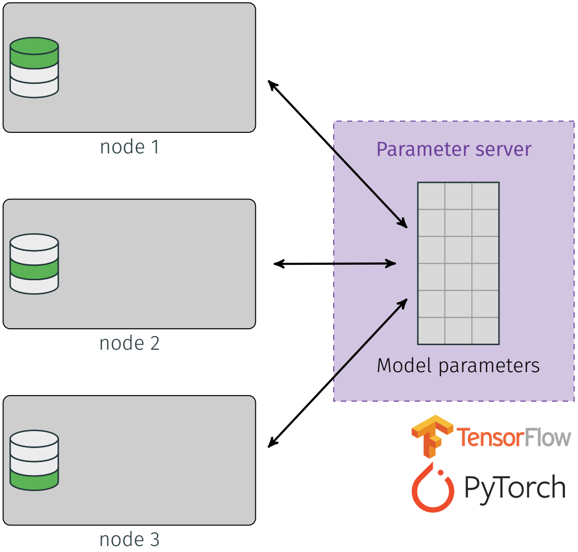
Distributed Training Has Become a Necessity



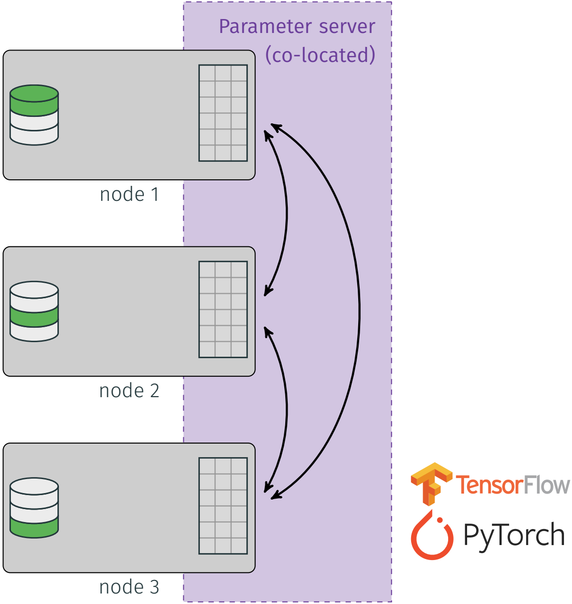
Distributed Training Has Become a Necessity



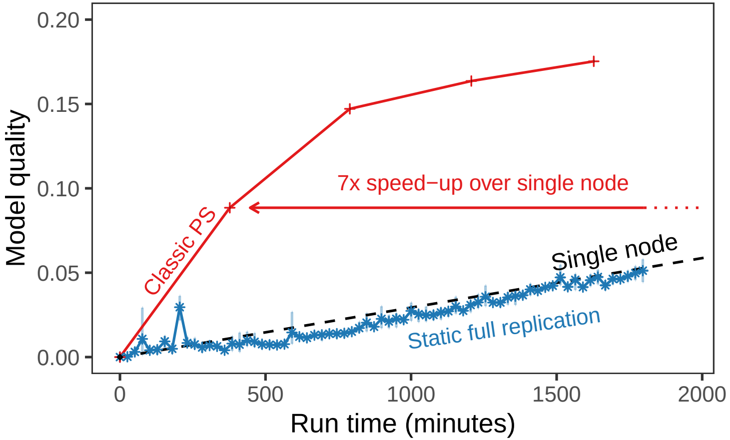
Parameter Servers Facilitate Distributed Training



Parameter Servers Facilitate Distributed Training



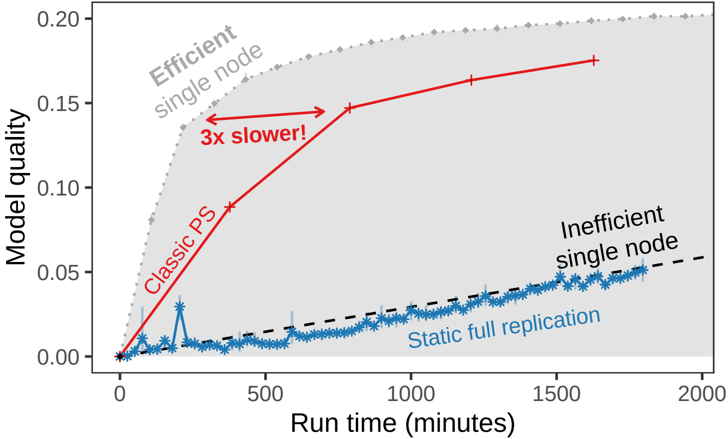
Training knowledge graph embeddings:



ComplEx-500 on Wikidata5M, 8 nodes x 32 threads, 100 Gbit/s InfiniBand

How Fast Is Distributed Training?

Training knowledge graph embeddings:

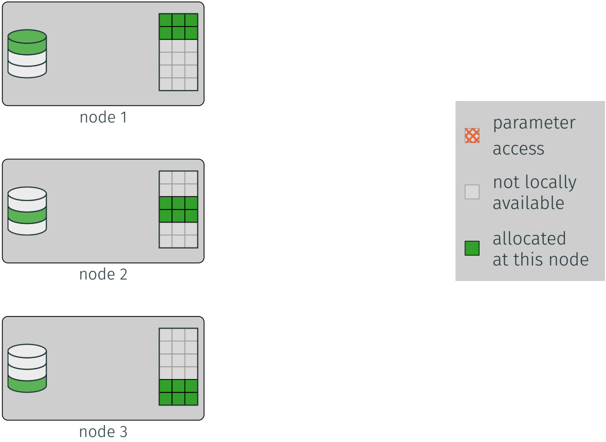


ComplEx-500 on Wikidata5M, 8 nodes x 32 threads, 100 Gbit/s InfiniBand

The problem:
Communication overhead

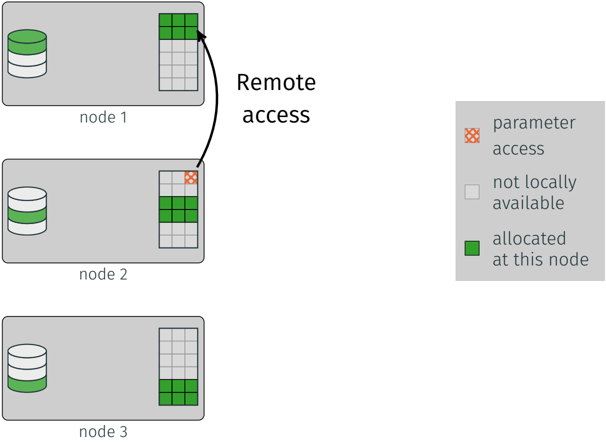
Communication Overhead in a Classic PS

Approach: partition the parameters to nodes



Communication Overhead in a Classic PS

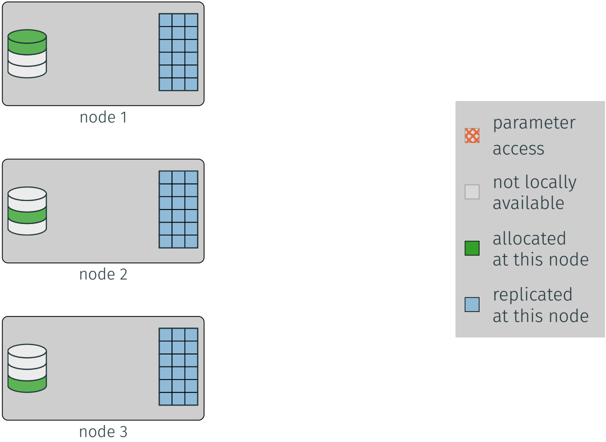
Approach: partition the parameters to nodes



Problem: remote parameter accesses

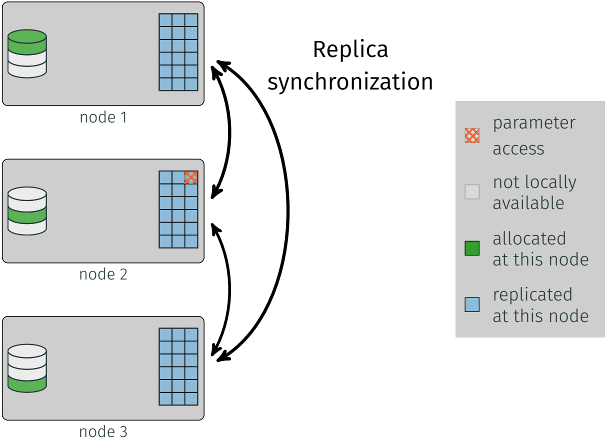
Communication Overhead with Full Static Replication

Approach: replicate all parameters to all nodes



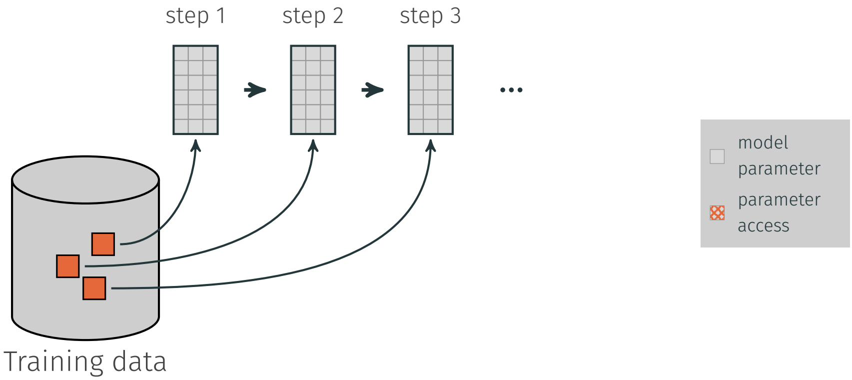
Communication Overhead with Full Static Replication

Approach: replicate all parameters to all nodes

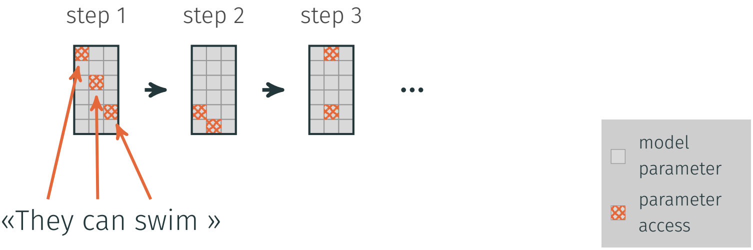


Problems: limited model size and communication intensive

Sparse Parameter Access

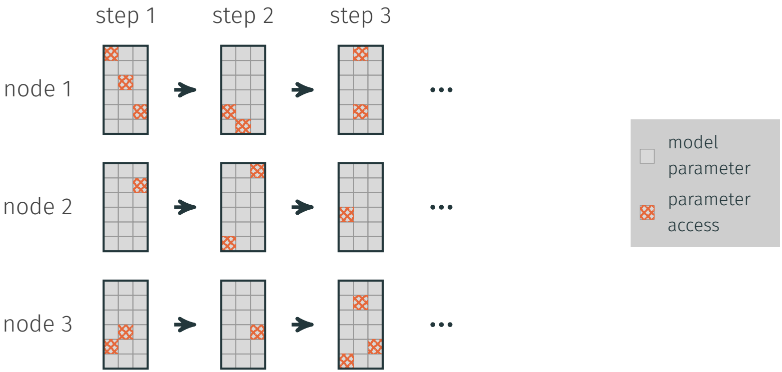


Sparse Parameter Access



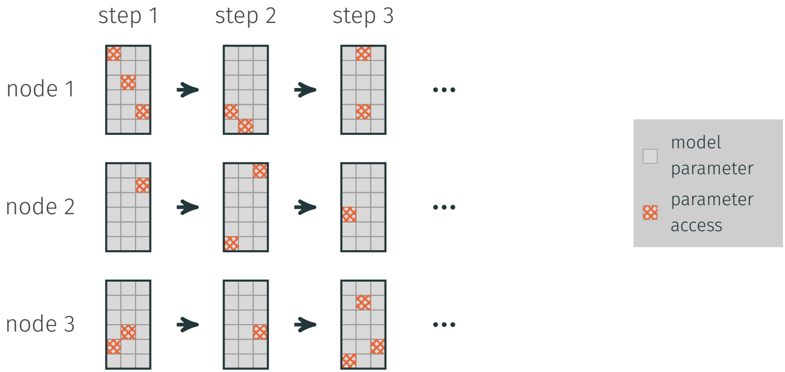
Sparse access:
each step accesses only a
subset of all parameters

Sparse Parameter Access



Challenge: dynamic access pattern

Sparse Parameter Access



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

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

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

Parameter Access Locality

- 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**

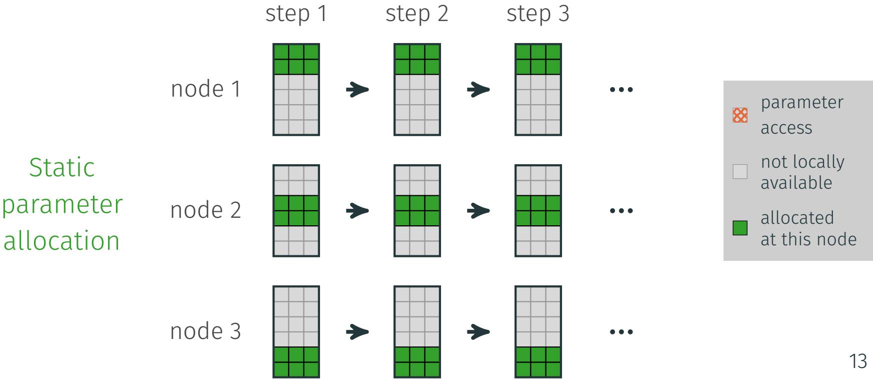
- Common locality techniques:
  - Data clustering
  - Parameter blocking
  - Latency hiding

$\left.\begin{array}{l}\text{Data clustering}\\\text{Parameter blocking}\end{array}\right\}$ Manually create locality

$\left.\text{Latency hiding}\right\}$ Exploits inherent locality

Problem: Parameter Servers Do Not Support Locality Techniques

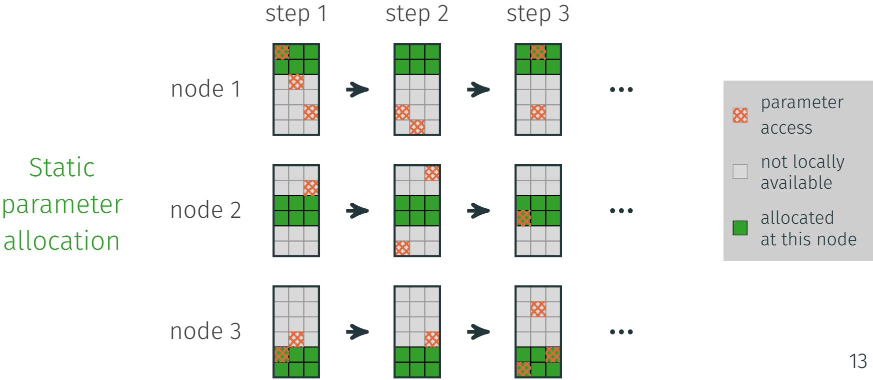
- Main limitation: parameter allocation is static



Problem: Parameter Servers Do Not Support Locality Techniques

- Main limitation: parameter allocation is static

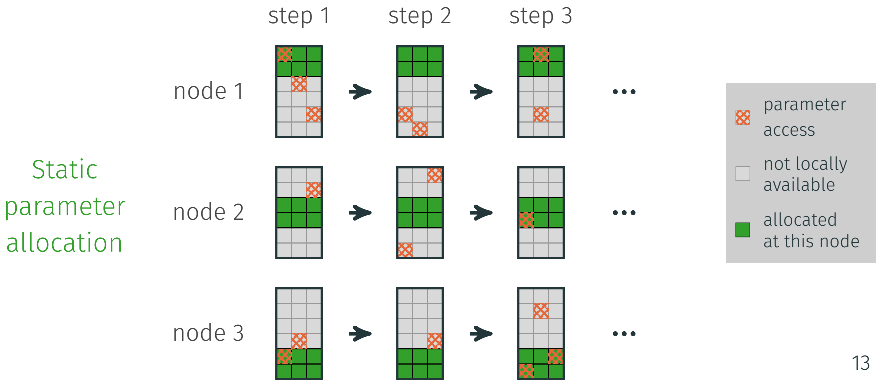
Example: latency hiding



Problem: Parameter Servers Do Not Support Locality Techniques

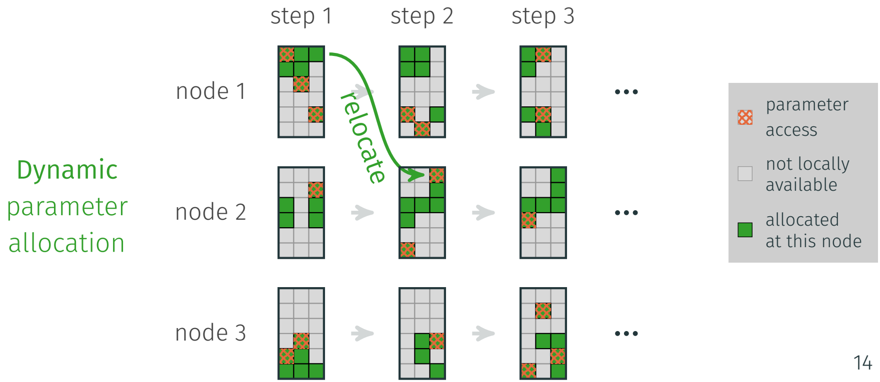
- Main limitation: parameter allocation is static
- Locality techniques need to be implemented with low-level communication primitives

Example: latency hiding



Our Approach: Dynamic Parameter Allocation

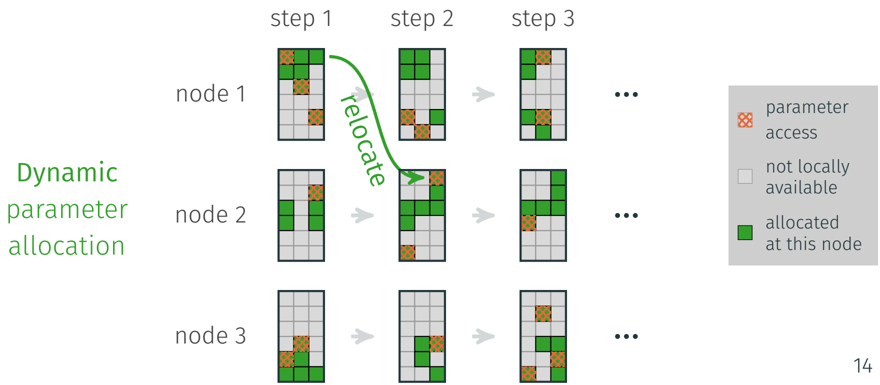
Relocate parameters to where they are accessed



Our Approach: Dynamic Parameter Allocation

Relocate parameters to where they are accessed

New primitive to control allocation: `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

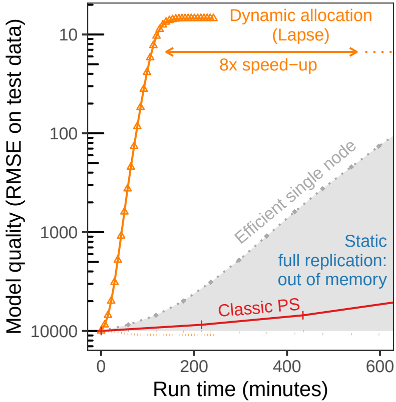
Lapse: the First Dynamic Allocation Parameter Server

- 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 ✓

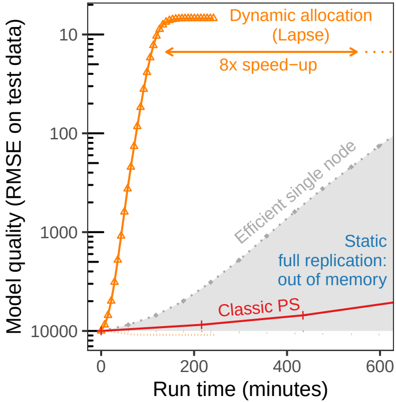


Matrix factorization (uniform matrix),
8 nodes x 8 threads, 100 Gbit/s InfiniBand

With **latency hiding**:

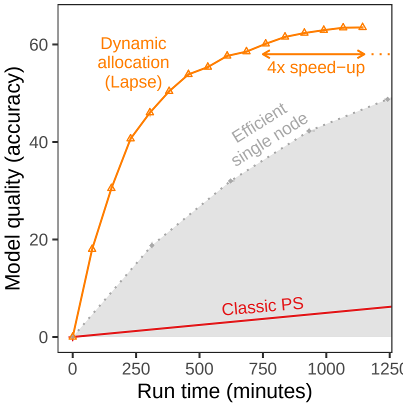
Can Dynamic Allocation Improve Efficiency?

With **data clustering** or **parameter blocking**:
Efficient ✓



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

With **latency hiding**:
Relatively efficient for some tasks, but not all

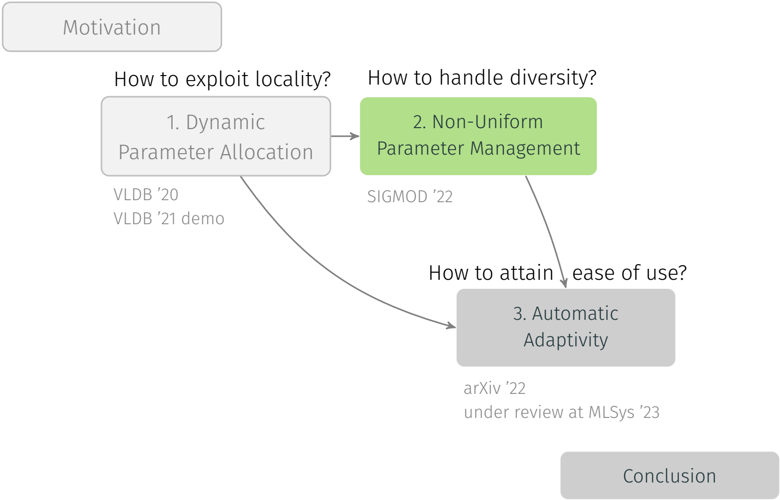


Word embeddings,
8 nodes x 4 threads, 10 Gbit/s Ethernet

Recap Part 1: Dynamic Parameter Allocation

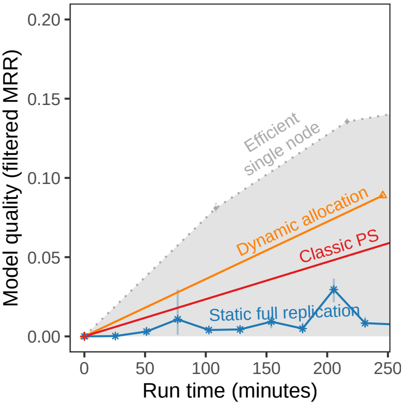
- Existing parameter servers cannot exploit locality
- Dynamic allocation enables support
- Efficiency:
  - With *data clustering* and *parameter blocking*: excellent
  - With *latency hiding*: varied

Outline: Adaptive Parameter Servers



Dynamic Allocation Can Be Inefficient

Knowledge graph embeddings

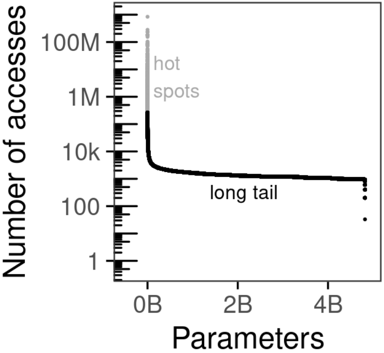


8 nodes x 32 threads, 100 Gbit/s InfiniBand

Problem: Non-Uniform Parameter Access

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

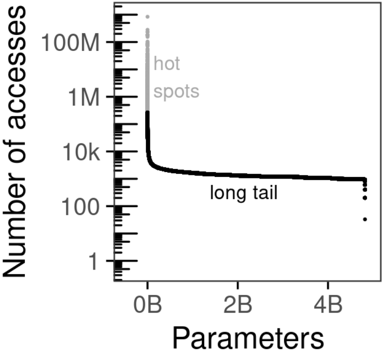
One source: Skew



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

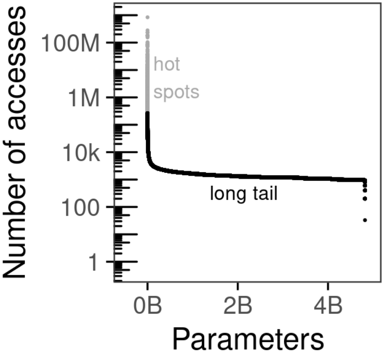
and

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

direct access

Not in this talk

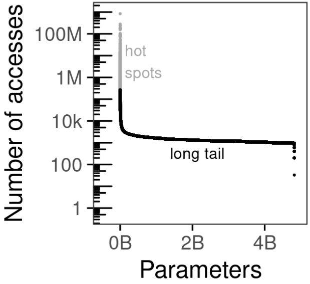
sampling access

Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters

Dynamic allocation (latency hiding) parameter server

Replication parameter server

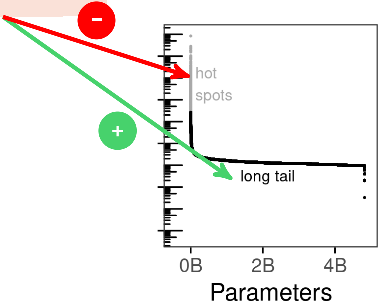


Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters

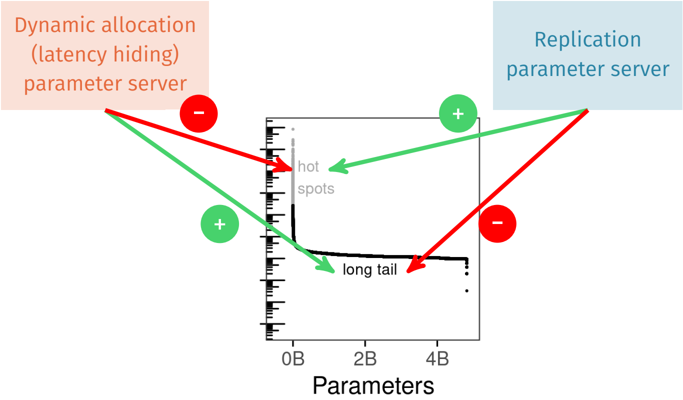
Dynamic allocation (latency hiding) parameter server

Replication parameter server



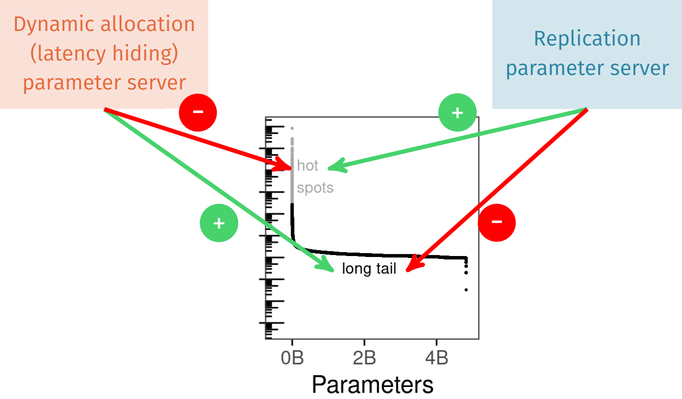
Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters



Skew Is a Challenge For Parameter Servers

Parameter servers use one technique for all parameters



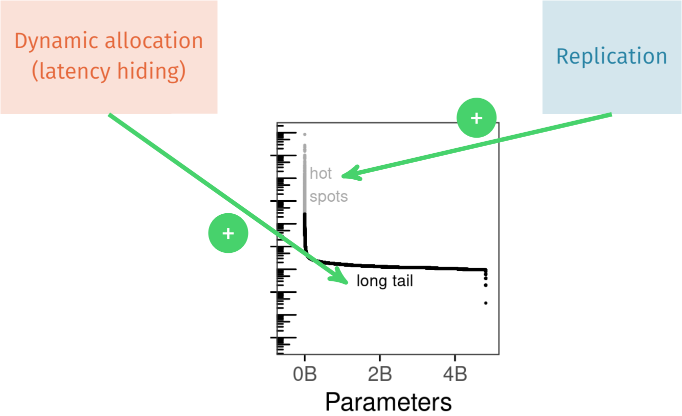
Problem: no technique is efficient for all access patterns

Our Approach: Multi-Technique Parameter Management

- In general: pick a suitable technique **per parameter**

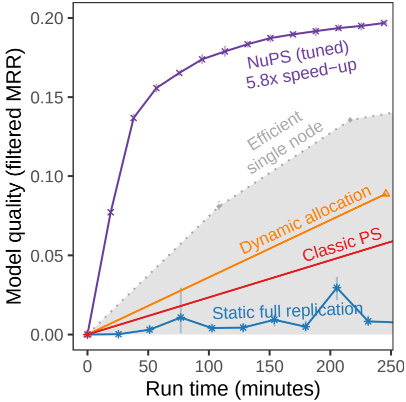
Our Approach: Multi-Technique Parameter Management

- In general: pick a suitable technique **per parameter**
- NuPS: dynamic allocation + replication

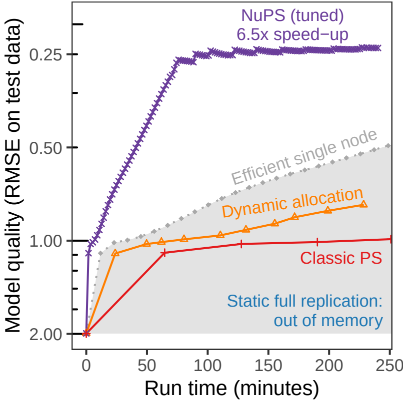


Can Non-Uniform Parameter Management Improve Efficiency?

Knowledge graph embeddings



Matrix factorization



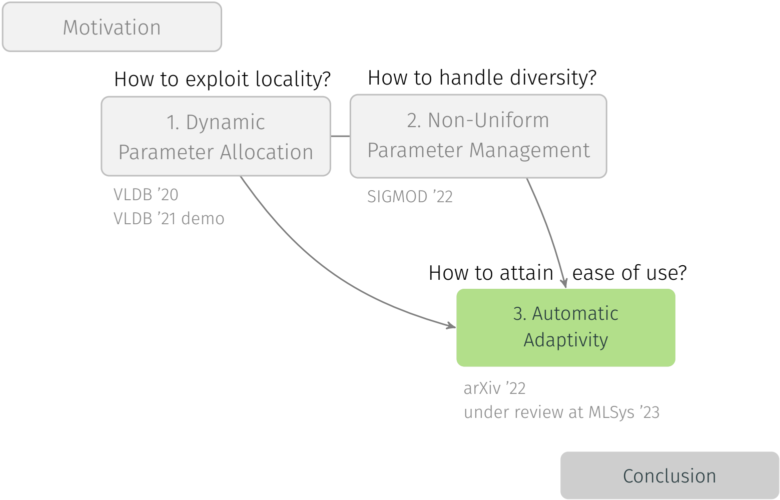
NuPS was efficient on multiple tasks.

8 nodes x 32 threads, 100 Gbit/s InfiniBand

Recap Part 2: Non-Uniform Parameter Management

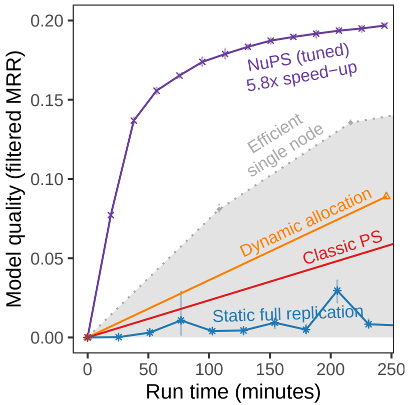
- Skew is common, and a challenge for single-technique parameter servers
- Parameter servers can improve efficiency by adapting management techniques to access patterns

Outline: Adaptive Parameter Servers

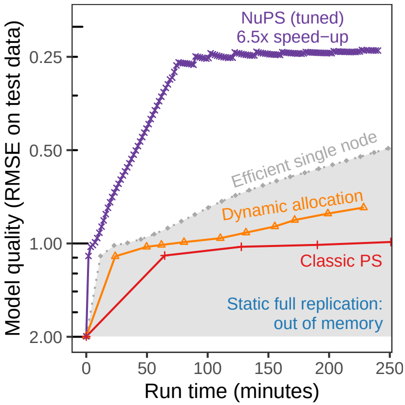


NuPS's Performance Depends on Configuration Knobs

Knowledge graph embeddings



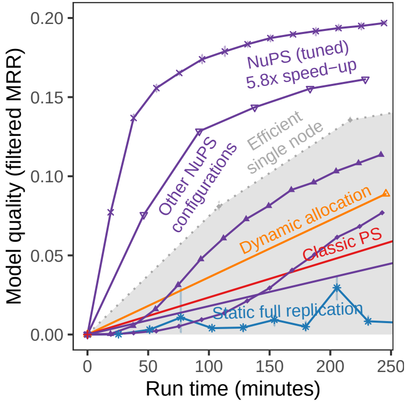
Matrix factorization



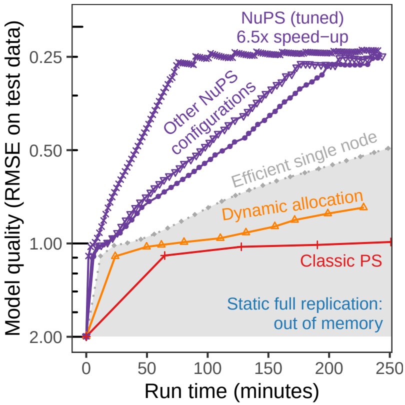
8 nodes x 32 threads, 100 Gbit/s InfiniBand

NuPS's Performance Depends on Configuration Knobs

Knowledge graph embeddings



Matrix factorization



NuPS performance depends on configuration knobs.

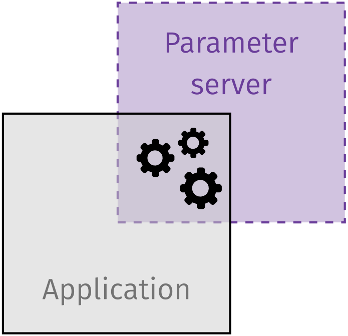
8 nodes x 32 threads, 100 Gbit/s InfiniBand

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

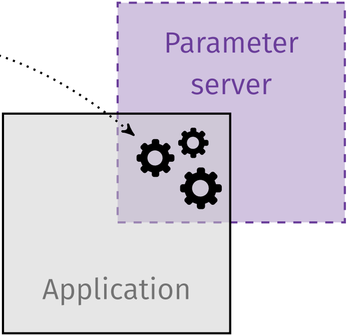


Problem: Efficient Parameter Servers Are Complex To Use

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)

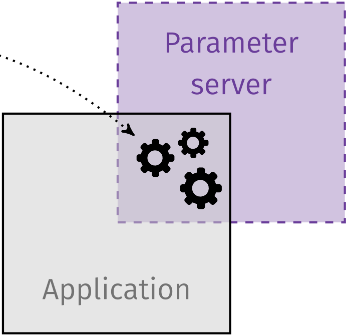


Problem: Efficient Parameter Servers Are Complex To Use

Parameter servers: easy to use OR efficient

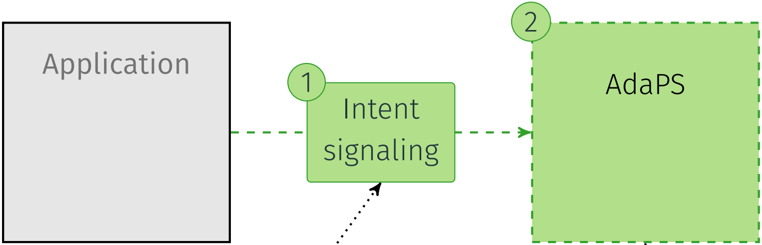
Configuration knobs:

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



A mechanism to pass information about upcoming parameter accesses

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:

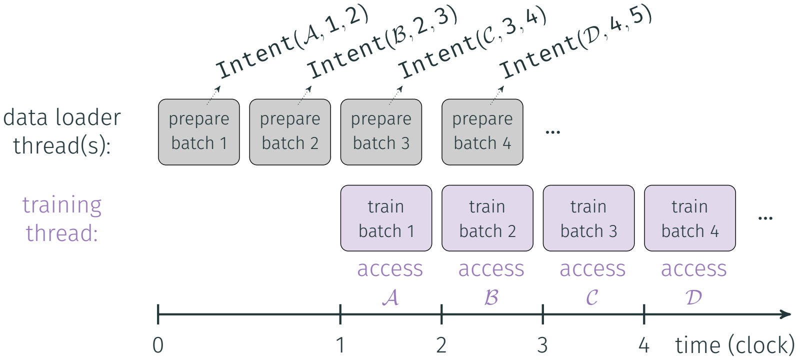
« *I will access parameters 13 and 16 in batch 2* »

Primitive:

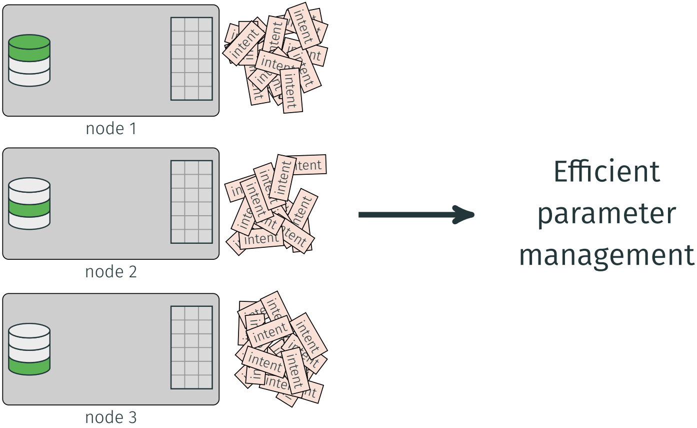
```
Intent(parameters, start, end)
```

Integration into Machine Learning Systems

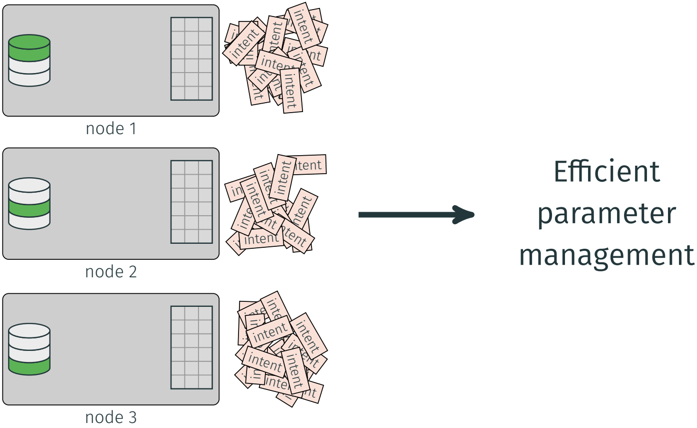
Integrates naturally into the data loader of common machine learning systems:



AdaPS: a Fully-Adaptive, Zero-Tuning Parameter Server



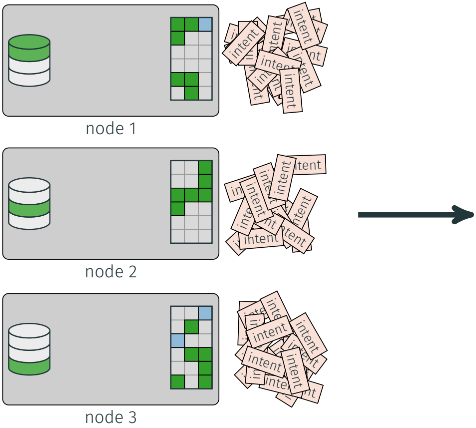
AdaPS: a Fully-Adaptive, Zero-Tuning Parameter Server



Intent challenges:

1. Large quantities
2. Signalled decentrally
3. Signalled ahead of time

AdaPS: a Fully-Adaptive, Zero-Tuning Parameter Server



AdaPS figures out:

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

Intent challenges:

1. Large quantities
2. Signalled decentrally
3. Signalled ahead of time

AdaPS Main Components

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

AdaPS Main Components

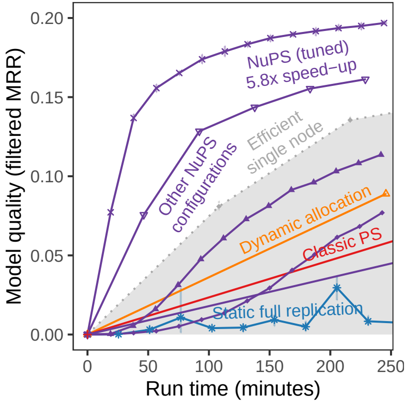
1. Automatic choice of technique
    - Collect intents
    - Rule-based decision
2. Automatic action timing
    - Challenge: timings unknown
    - Learn correct timing, probabilistic approach

AdaPS Main Components

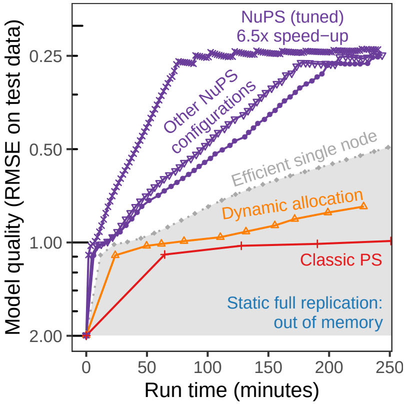
1. Automatic choice of technique
    - Collect intents
    - Rule-based decision
2. Automatic action timing
    - Challenge: timings unknown
    - Learn correct timing, probabilistic approach
3. Efficiency

Can Automatic Adaptation be Efficient?

Knowledge graph embeddings

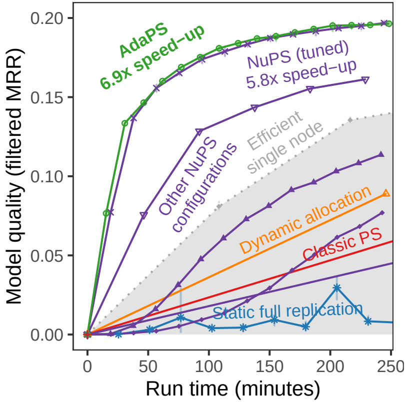


Matrix factorization

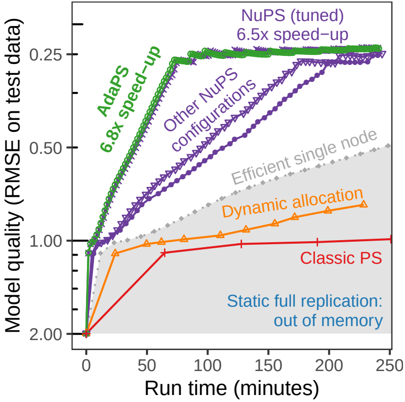


8 nodes x 32 threads, 100 Gbit/s InfiniBand

Knowledge graph embeddings



Matrix factorization



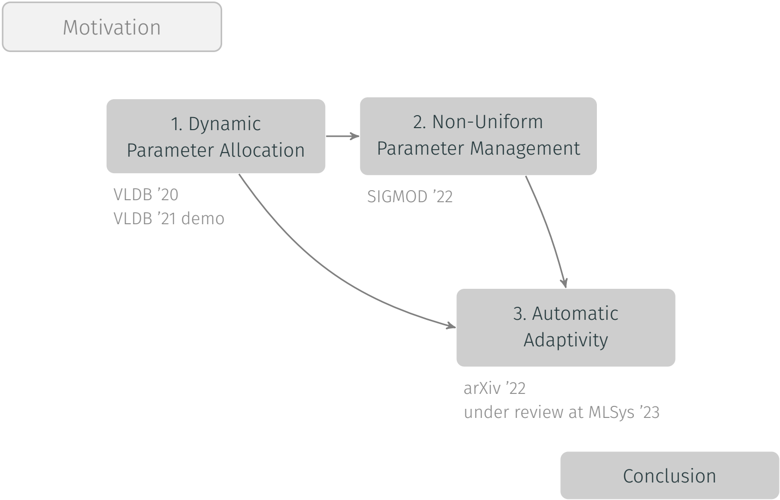
AdaPS was efficient out of the box for many ML tasks.

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



Conclusions: Adaptive Parameter Servers

Motivation

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

1. Dynamic Parameter Allocation

VLDB '20
VLDB '21 demo

2. Non-Uniform Parameter Management

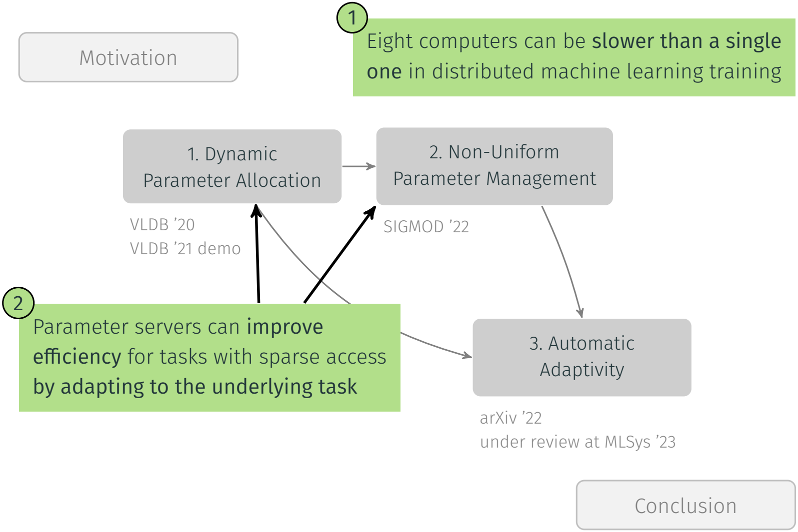
SIGMOD '22

3. Automatic Adaptivity

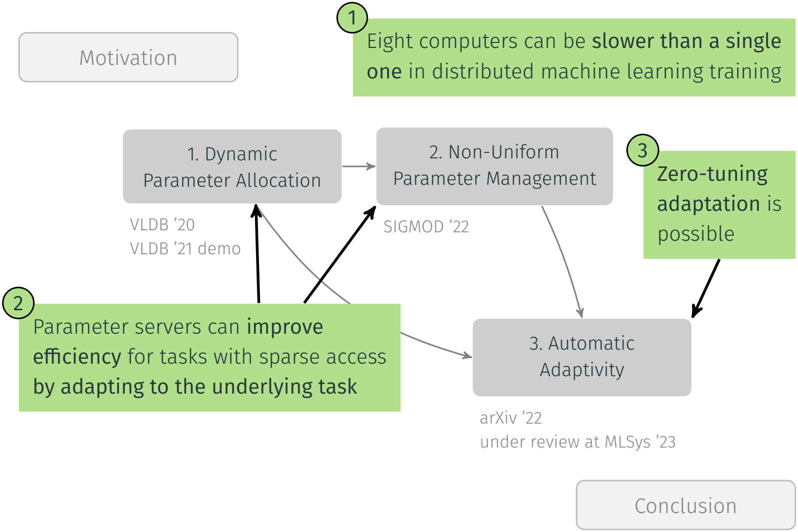
arXiv '22
under review at MLSys '23

Conclusion

Conclusions: Adaptive Parameter Servers



Conclusions: Adaptive Parameter Servers



Conclusions: Adaptive Parameter Servers

Motivation

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

1. Dynamic Parameter Allocation

VLDB '20
VLDB '21 demo
Lapse

2. Non-Uniform Parameter Management

SIGMOD '22
NuPS

3 **Zero-tuning adaptation** is possible

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

3. Automatic Adaptivity

arXiv '22
under review at MLSys '23
AdaPS

Conclusion

Conclusions: Adaptive Parameter Servers

Motivation

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

1. Dynamic Parameter Allocation

VLDB ’20
VLDB ’21 demo
Lapse

2. Non-Uniform Parameter Management

SIGMOD ’22
NuPS

3 **Zero-tuning adaptation** is possible

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

3. Automatic Adaptivity

arXiv ’22
under review at MLSys ’23
AdaPS

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

Conclusion

Publications

The thesis is based on the following publications:

- A. Renz-Wieland, R. Gemulla, S. Zeuch, V. Markl. Dynamic Parameter Allocation in Parameter Servers. PVLDB, 13(11): 1877–1890. 2020.
- A. Renz-Wieland, R. Gemulla, Z. Kaoudi, V. Markl. NuPS: A Parameter Server for Machine Learning with Non-Uniform Parameter Access. In *Proceedings of the 2022 International Conference of Management of Data*. ACM, New York, NY, USA. 2022.
- A. Renz-Wieland, A. Kieslinger, R. Gericke, R. Gemulla, Z. Kaoudi, V. Markl. Good Intentions: Adaptive Parameter Servers via Intent Signaling. CoRR, abs/2206.00470. 2022.

It also draws material from:

- A. Renz-Wieland, T. Drobisch, R. Gemulla, S. Zeuch, V. Markl. Just Move It! Dynamic Parameter Allocation in Action. PVLDB, 14(12): 2707-2710, 2021.