ISDS

TU Graz

SCIENCE
PASSION
TECHNOLOGY

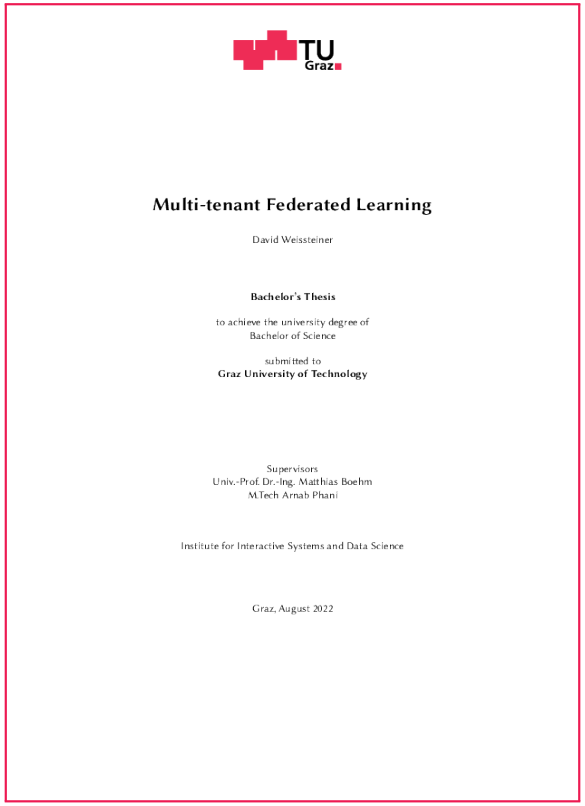
Multi-tenant Federated Learning

David Weissteiner

September 1, 2022

Outline

1. Federated Learning
2. Multi-tenant Federated Learning
   - Motivation
   - Isolation
   - Parallelization Strategies
   - Optimization
3. Experiments
   - Micro Benchmarks
   - Algorithm Performance
   - Hyper-parameter Tuning
4. Conclusions



TU Graz

Multi-tenant Federated Learning

David Weissteiner

Bachelor's Thesis

to achieve the university degree of
Bachelor of Science

submitted to
Graz University of Technology

Supervisors
Univ.-Prof. Dr.-Ing. Matthias Boehm
M.Tech Arnab Phani

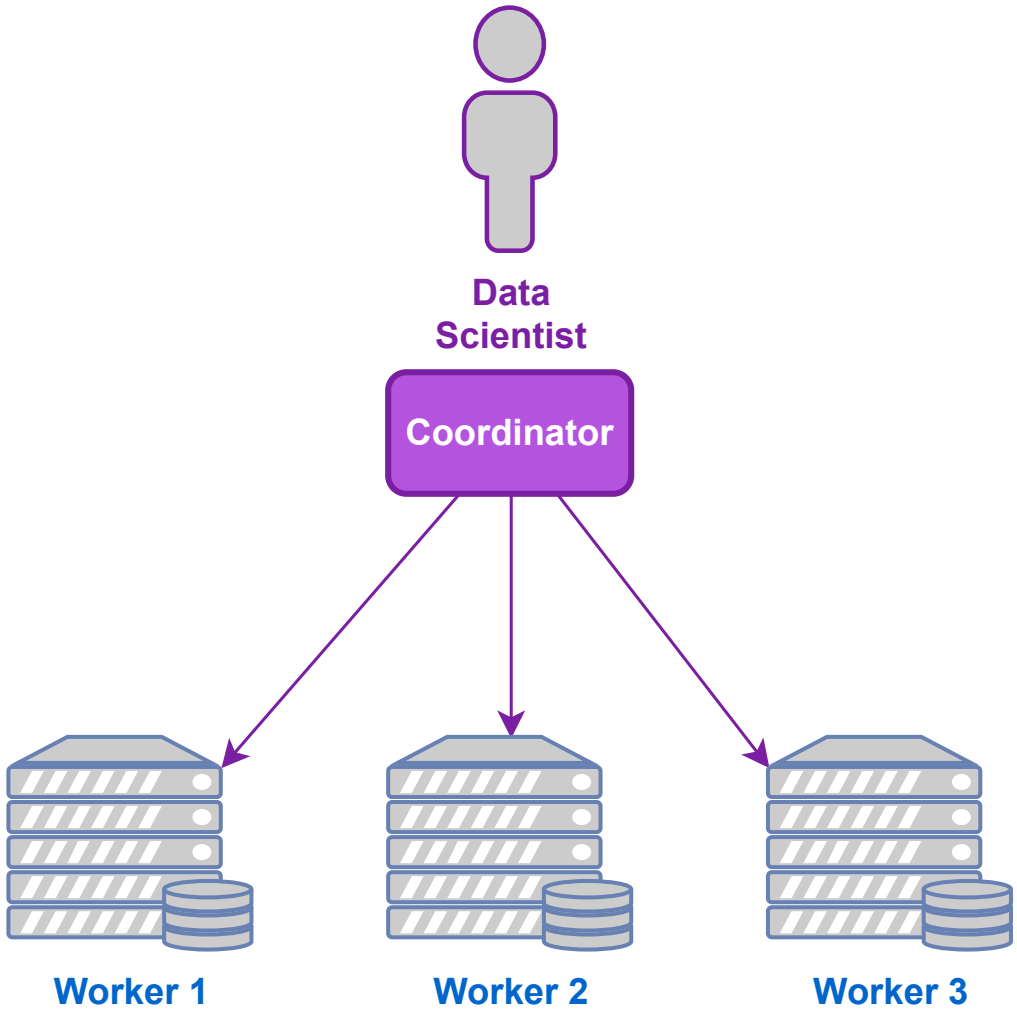
Institute for Interactive Systems and Data Science

Graz, August 2022

Federated Learning

- Learn model on decentralized data
- Addresses privacy and locality of data
- Coordinator
- Federated Worker resources:
  - Data w/ privacy constraints
  - Computing capabilities

Single-tenant Setup



Motivation

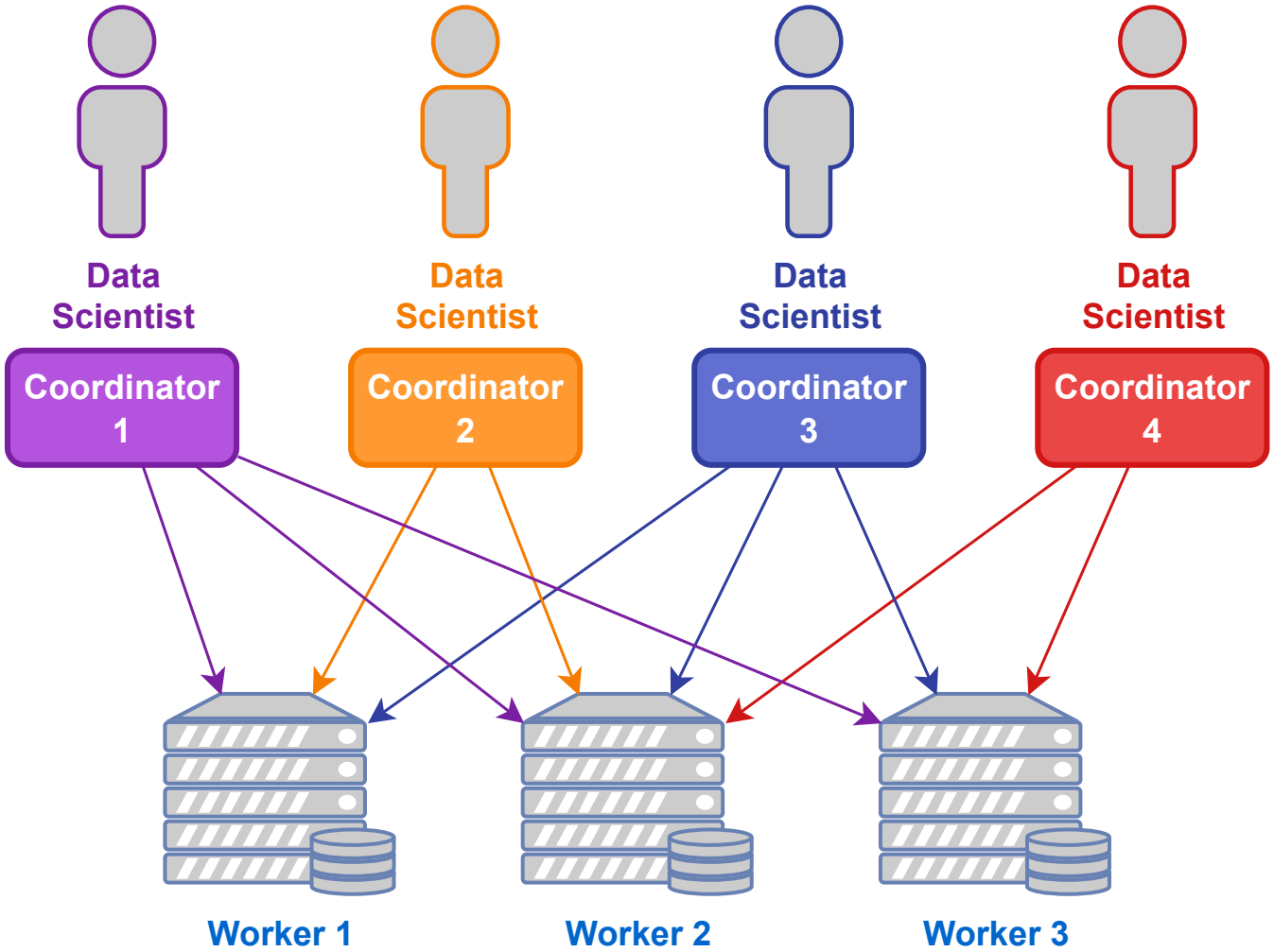
- Overcome the limitation to a single coordinator

Example

- Company with a large data science team
- Data scientists analyze data from federated sources
- Some of the federated workers need to be shared among the data scientists

- Usage agreements - one coordinator at a time

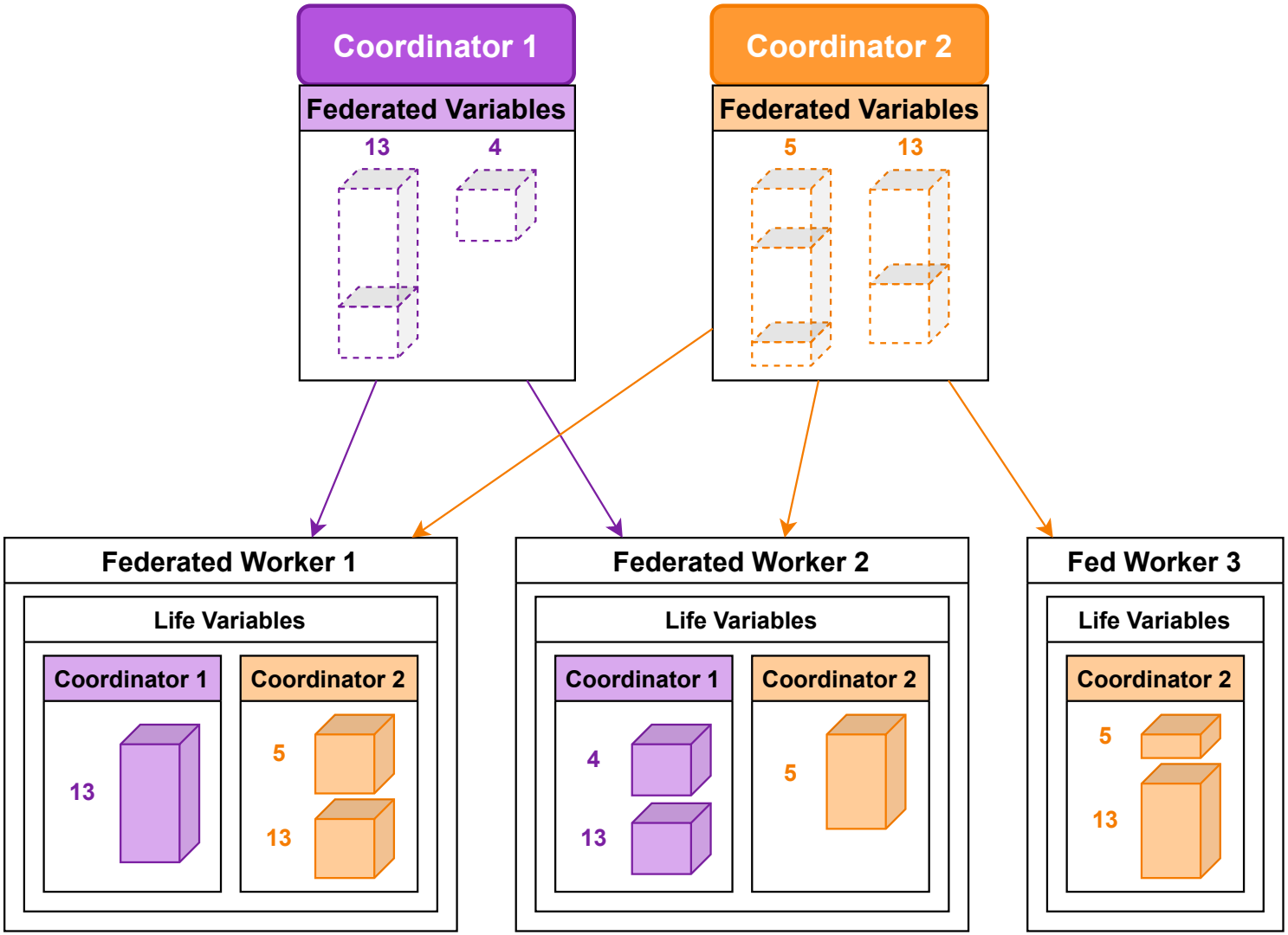
Multi-tenant Setup



Tenant Isolation

- Context: Concurrent requests from multiple coordinators
- Problem: Variable name conflicts
- Challenge: Ensure privacy
- Isolate the individual coordinators at the worker
- Tenant-specific execution context maps
  - Symbol table
  - Lineage map
  - Parfor worker contexts

Tenant Isolation

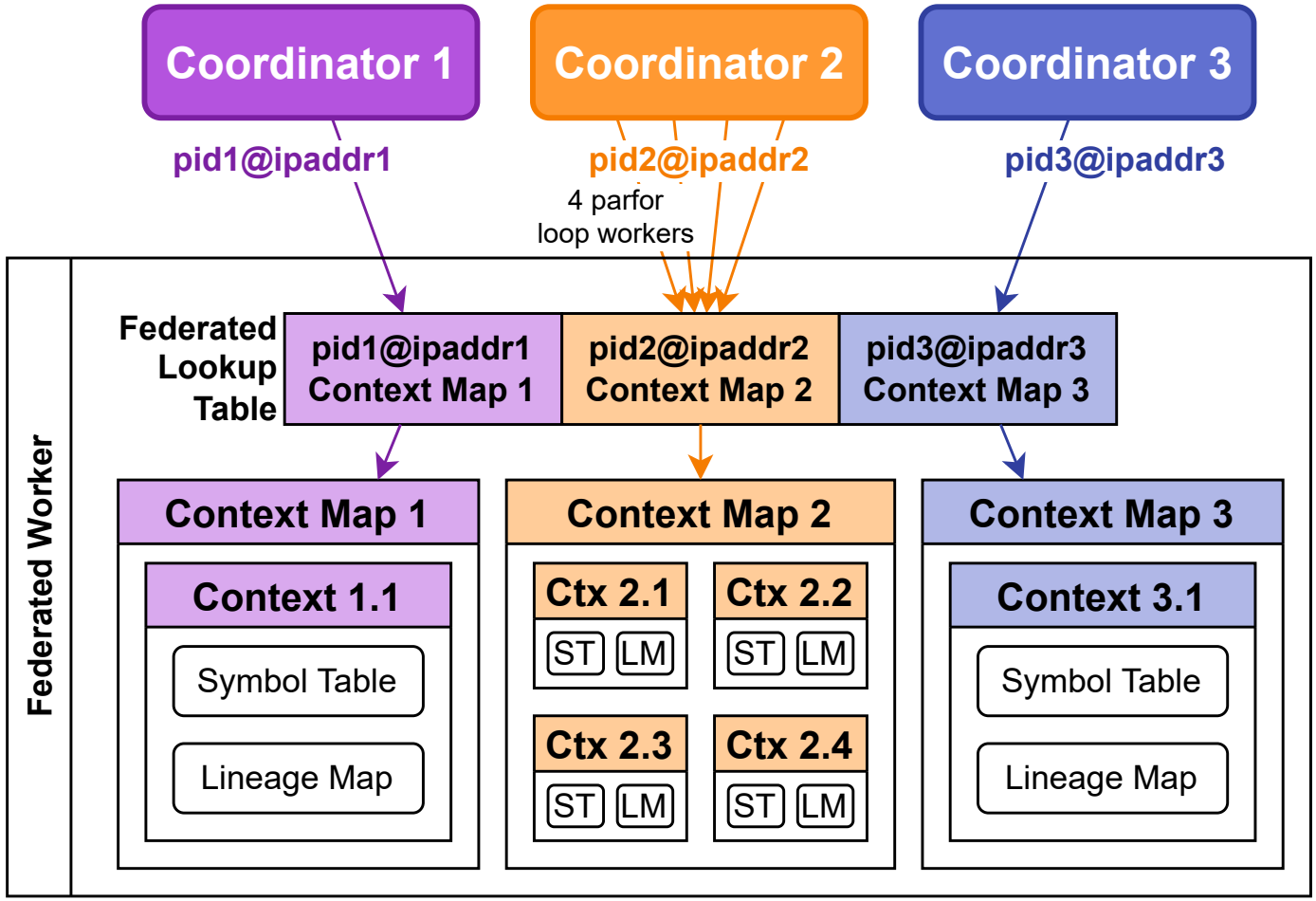


Tenant Distinction

- Differentiate the coordinators at the federated worker
- What makes a coordinator unique?
  (from the worker's perspective)
  - *IP address* from Netty's network channel
  - *Process ID* included in the federated request
- Unique coordinator identifier: IP + PID

Federated Lookup Table

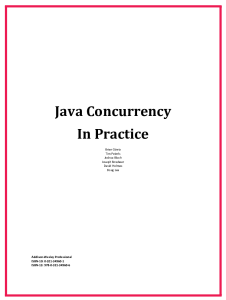
- Associate each coordinator to its context map



Multi-threaded Event Loop

- Reactor pattern of Netty
- Boss group: Single-threaded
  - Accept requests
  - Dispatch requests
- Worker group: Multi-threaded
  - Request deserialization
  - Computation
  - Response serialization

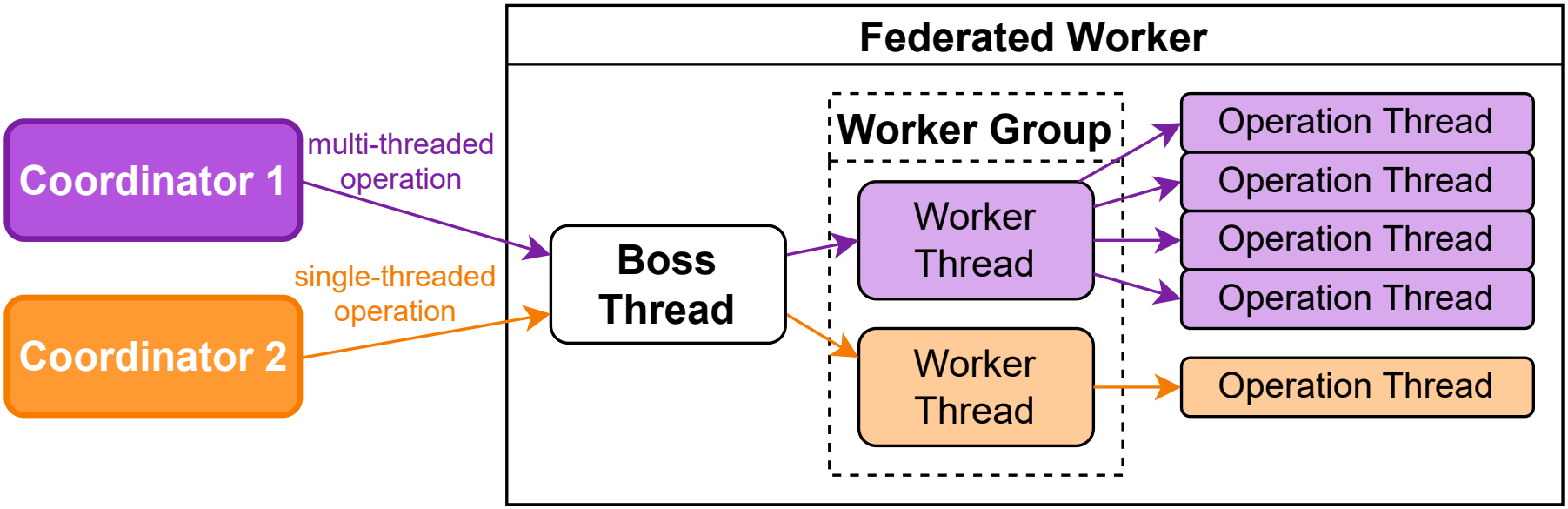
[Goetz et al. - JCIP 2006]



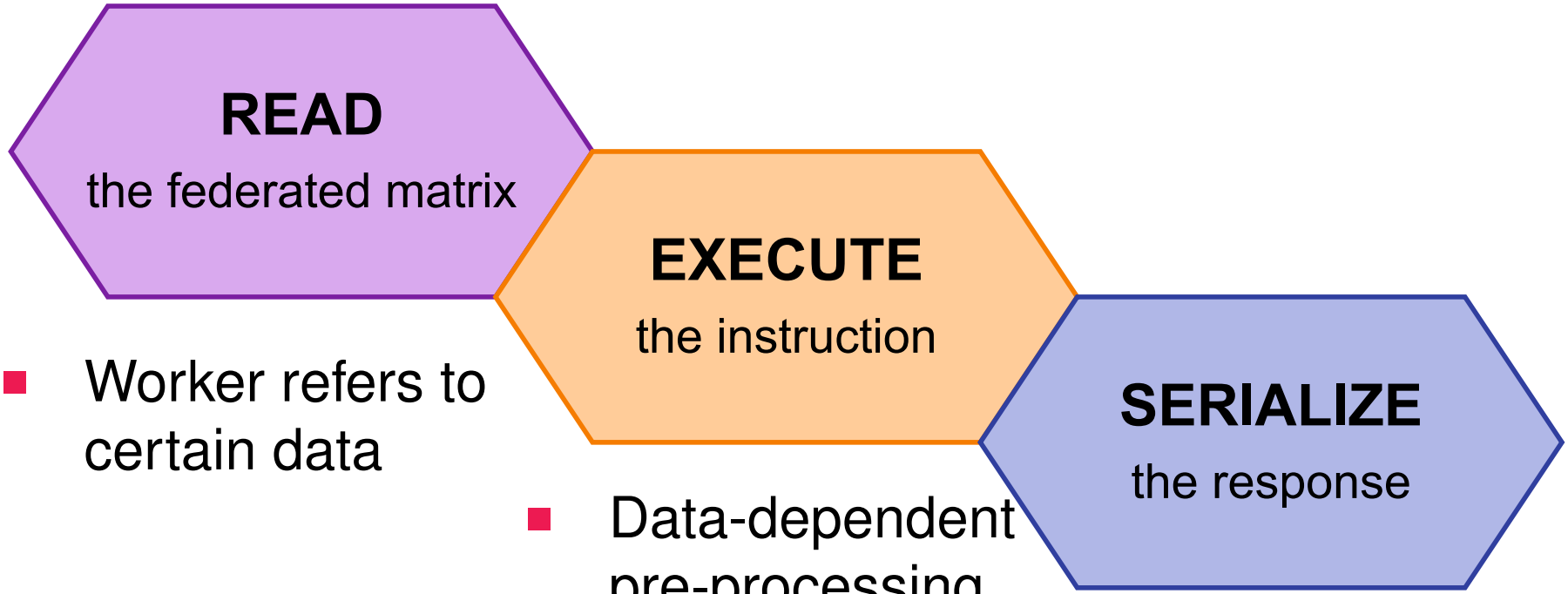
$$N_{threads} = N_{cpu} \cdot U_{cpu} \cdot \left(1 + \frac{W}{C}\right)$$

Multi-threaded Operator

- Adaptation of operator parallelism
- According to worker's resources



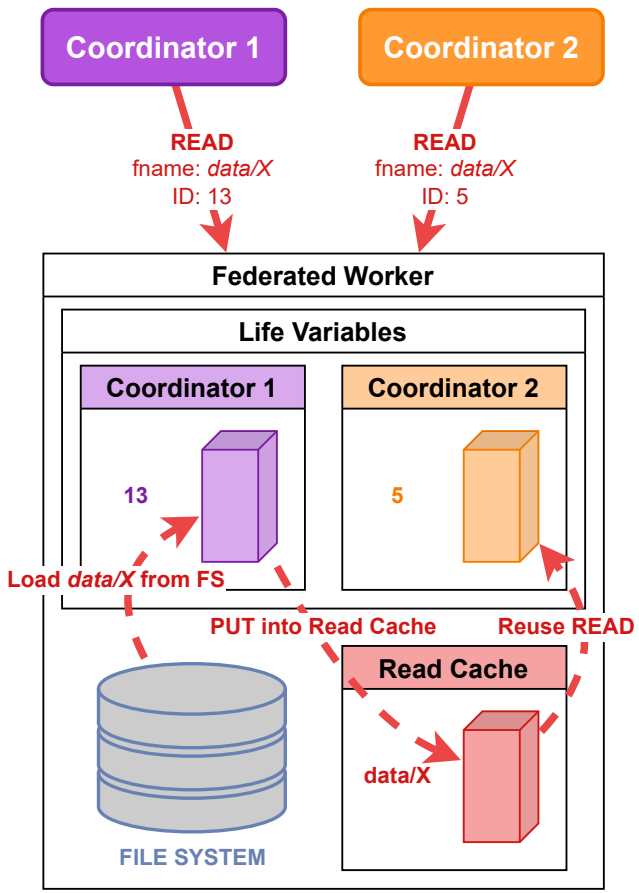
Multi-tenant Redundancy



- Worker refers to certain data
- Data-dependent pre-processing pipelines
- Federated pre-processing

Read Reuse

- Federated worker refers to certain data
- Redundant reads from file system
- Cache and reuse read data objects
- Federated Read Cache and Lineage-based Read Reuse



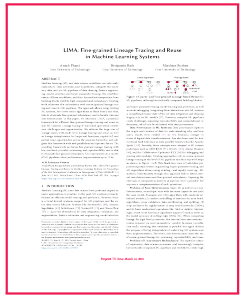
Lineage Trace Transfer

- Lineage item: Operation and Inputs
- Lineage trace: DAG of lineage items
- Lineage map: Associates a variable with a lineage item
- Idea: Transfer the lineage trace with the variable
- Complete lineage map at the worker
- Serialization and deserialization of the lineage trace

```
X = read("mat");
Y = rowSums(X);
Z = Y + sum(X);
```

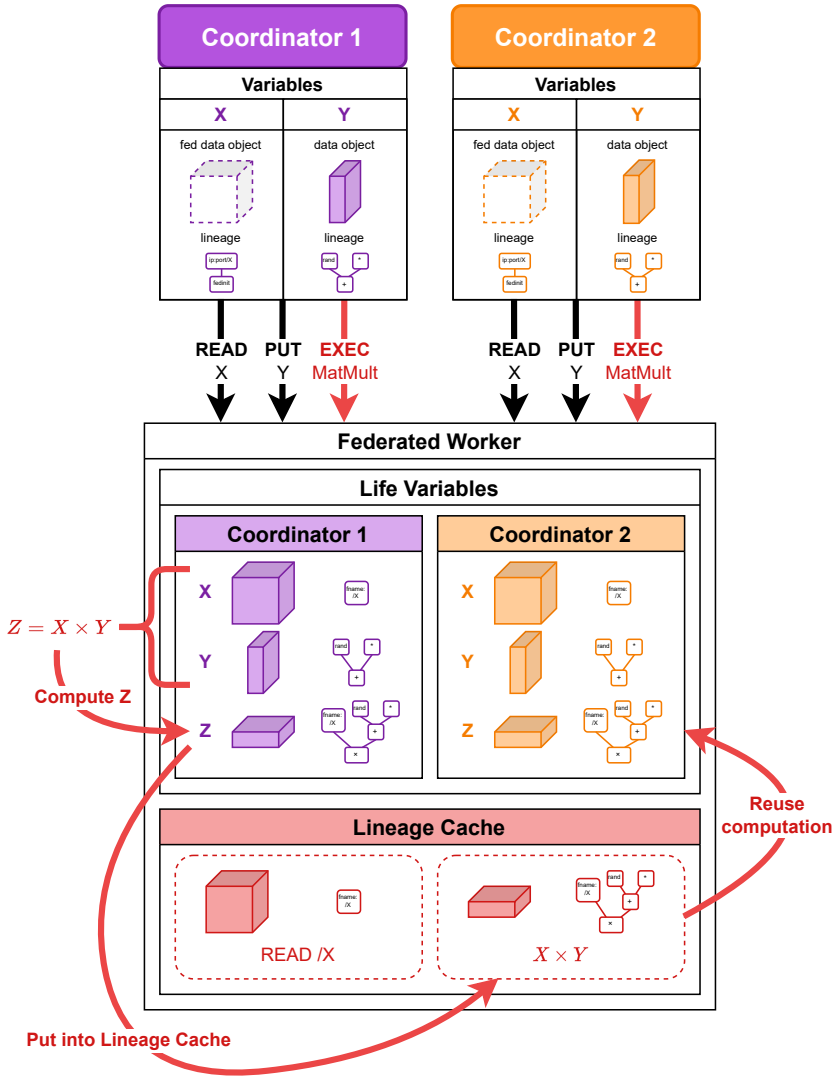
```
(0) (C) READ mat
(1) (I) uark+ (0)
(2) (I) uak+ (0)
(3) (I) + (1) (2)
```

[Phani et al. - LIMA 2021]



Federated Lineage-based Reuse

- Redundancy at the federated worker
- One shared lineage cache at the worker
- Reuse intermediates within and across processing contexts



Response Serialization Reuse

- Sometimes, only pre-processing is federated
  - Federated worker serializes pre-processed data and sends it to the coordinator
- Redundant serializations of the same response
- Obtain the bytes of the serialized response from Netty's encode buffer
- Leverage the lineage cache for bytes of serialized responses
- Reuse the serialized response

Outline

Experimental Setup

Hardware

α-node	β-node
2× Intel Xeon Gold 6238R	AMD EPYC 7302
2.2-2.5 GHz	3.0-3.3 GHz
28/56 phys./virt. cores	16/32 phys./virt. cores
768 GB DDR4 RAM	128 GB DDR4 RAM

Dataset

Purely synthetic data
Varying number of rows; static 500/1000 columns

Experimental Setup cont.

Software

Ubuntu 20.04.1, Open-JDK Java 11.0.13, and SystemDS 3.0.0++

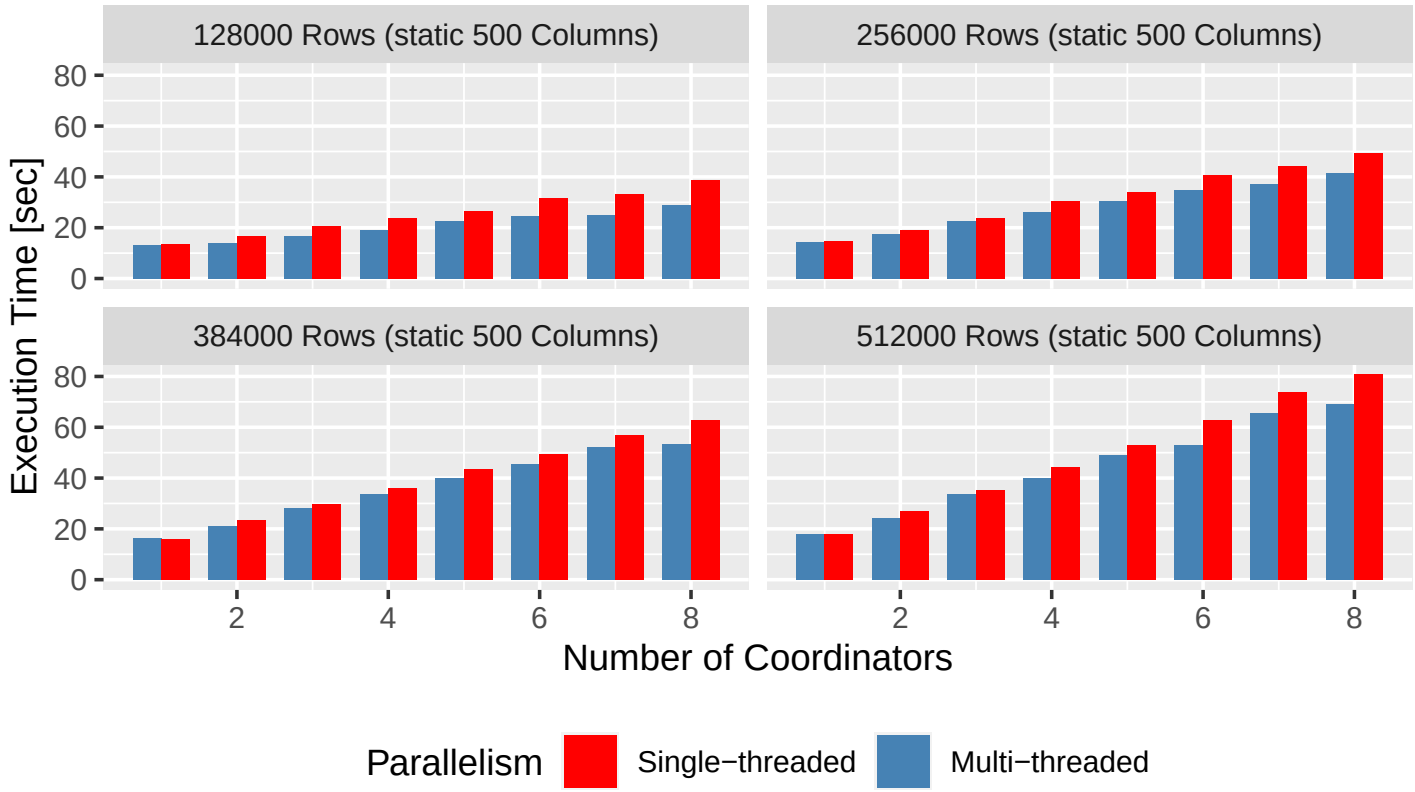
Setup of the federated infrastructure

- *Local*: Coordinators and workers on the same node.
- *LAN*: All coordinators on a single node, each worker on a separate node.
- *WAN*: LAN setup with added queueing discipline to emulate ping 47.5±12.5 ms and 2 MB/s bandwidth.

Multi-threaded Event Loop

WAN setup	Coordinators: β-node	1 worker: β-node

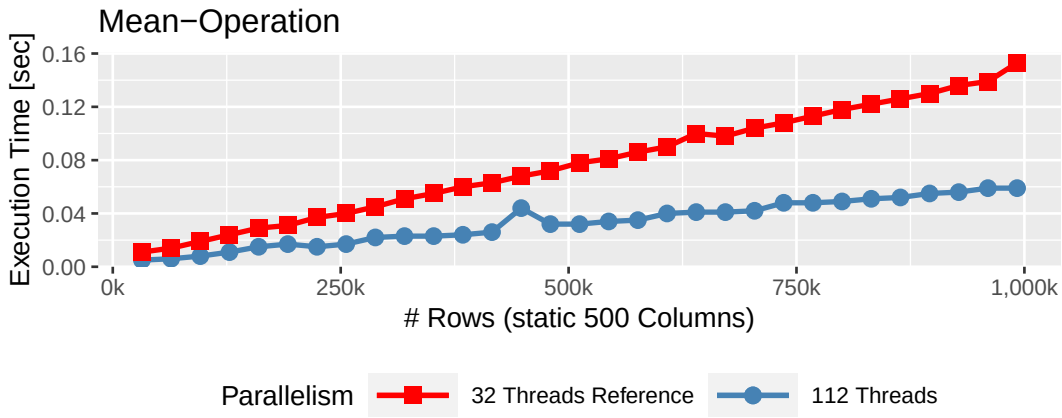
- Read federated matrix; broadcast 500×500 matrix
- Matrix multiplication and sum; get scalar

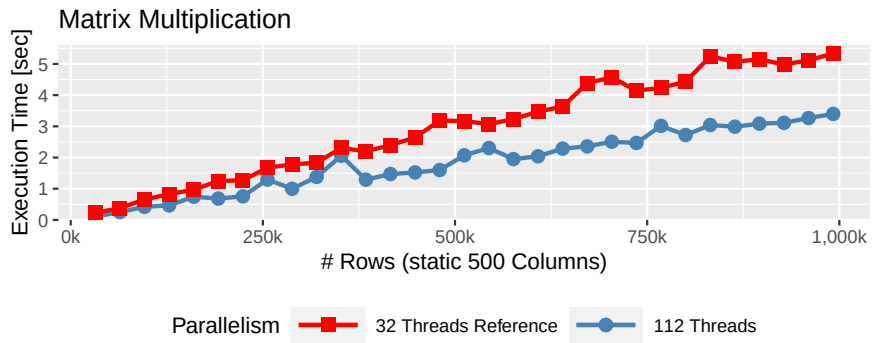


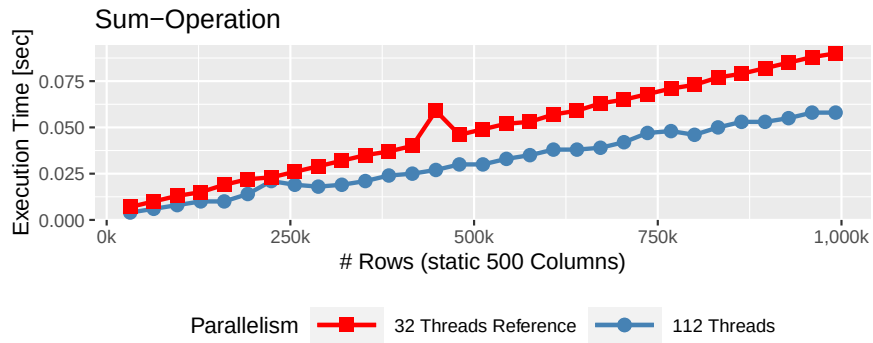
Multi-threaded Operator

LAN setup	1 coordinator: β-node	1 worker: α-node

- Perform mean, matrix multiplication, and sum on the federated matrix



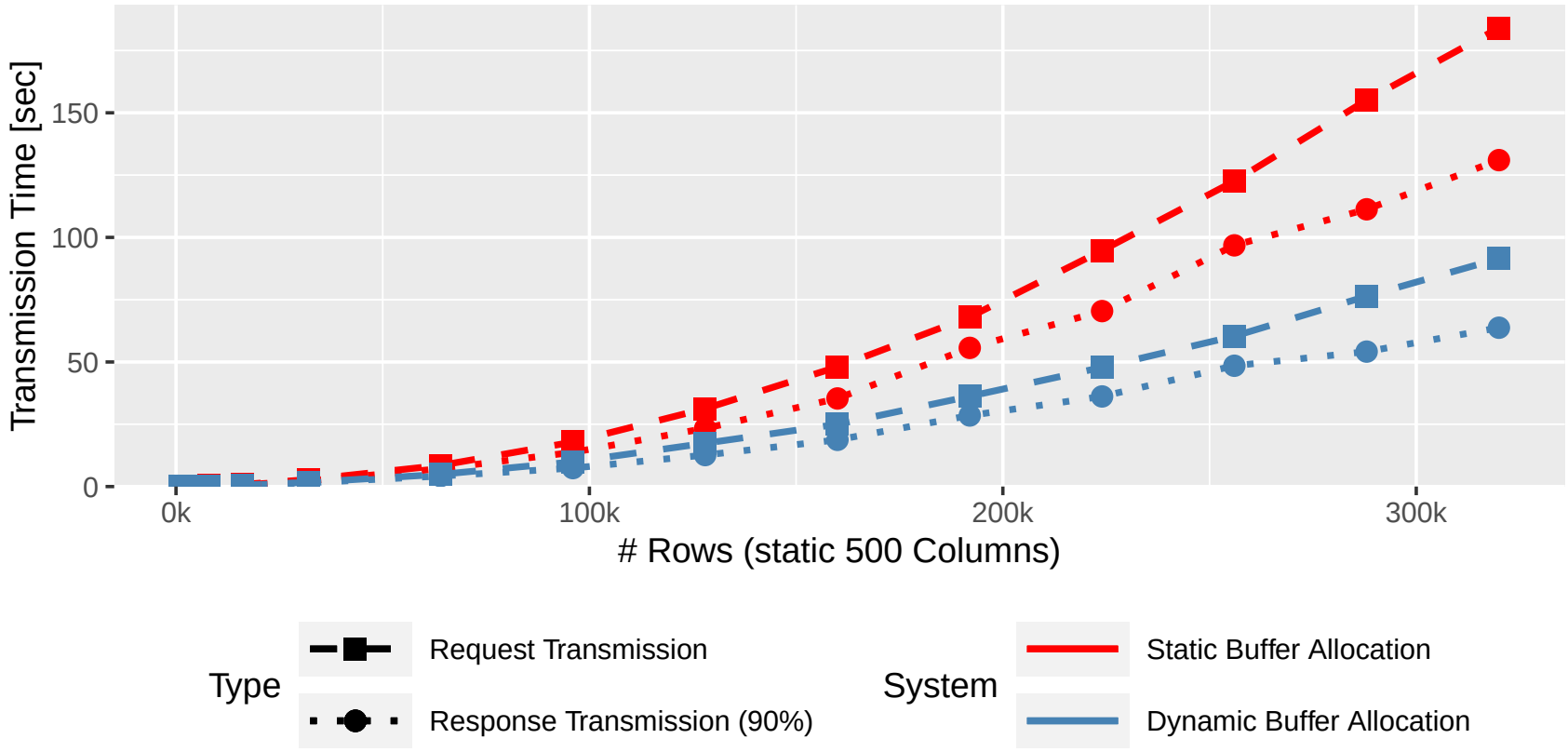




Netty Buffer Allocation

Local setup	1 coordinator and 1 worker: β-node

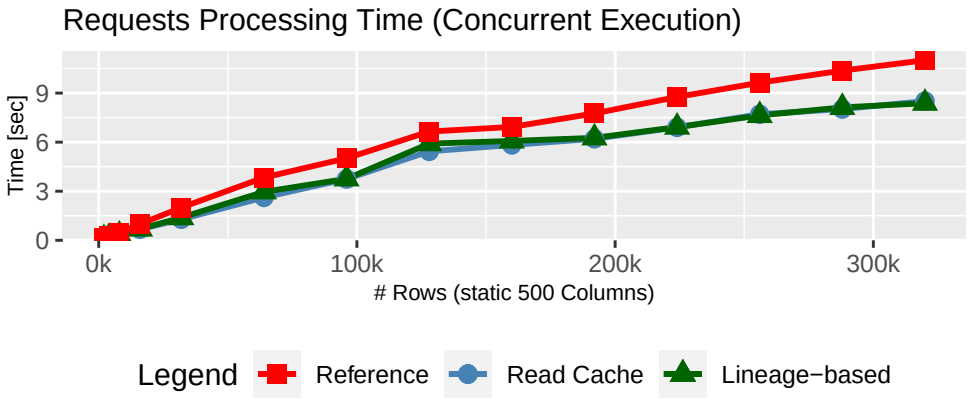
- Broadcast matrix to the worker
- Pull 90% back to the coordinator

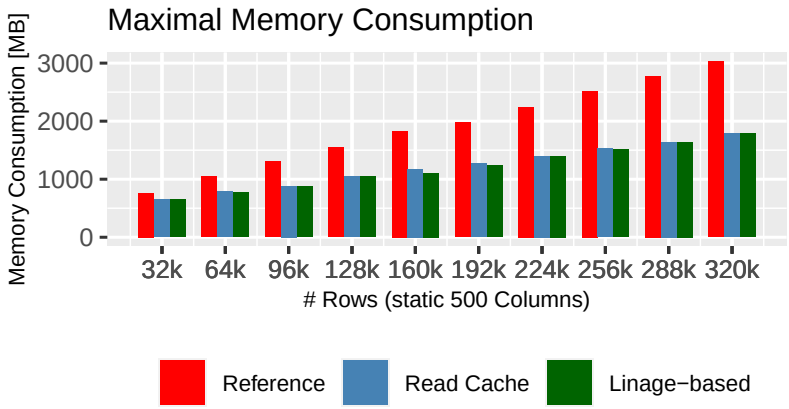


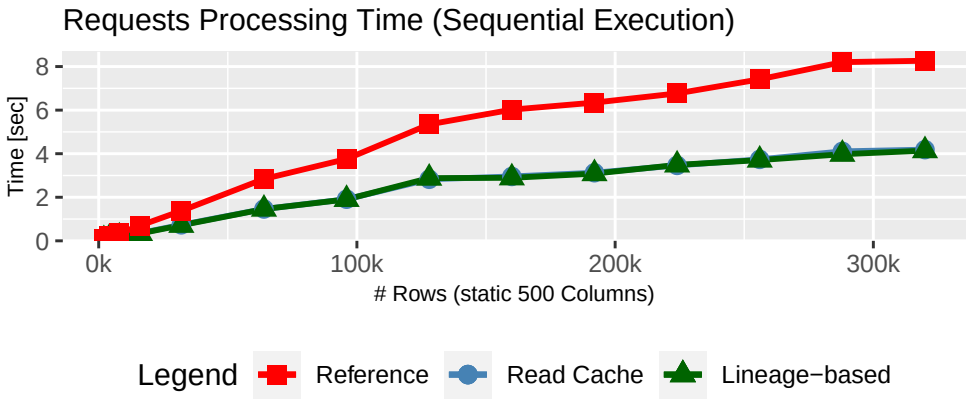
Federated Read Reuse

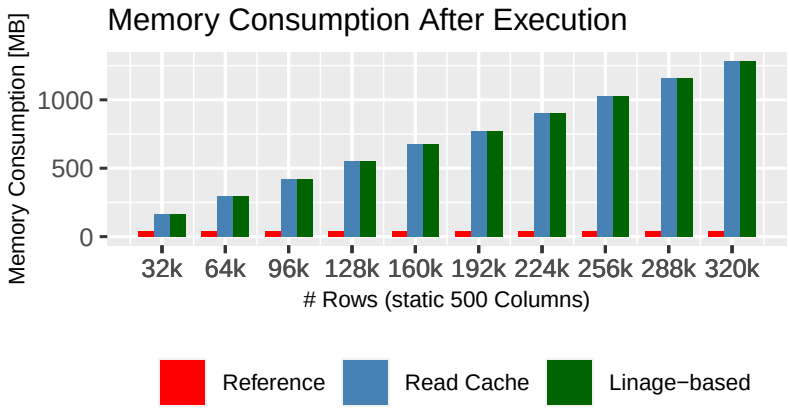
LAN setup	Coordinators: β-node	1 worker: β-node

- Sum-operation on the same federated matrix





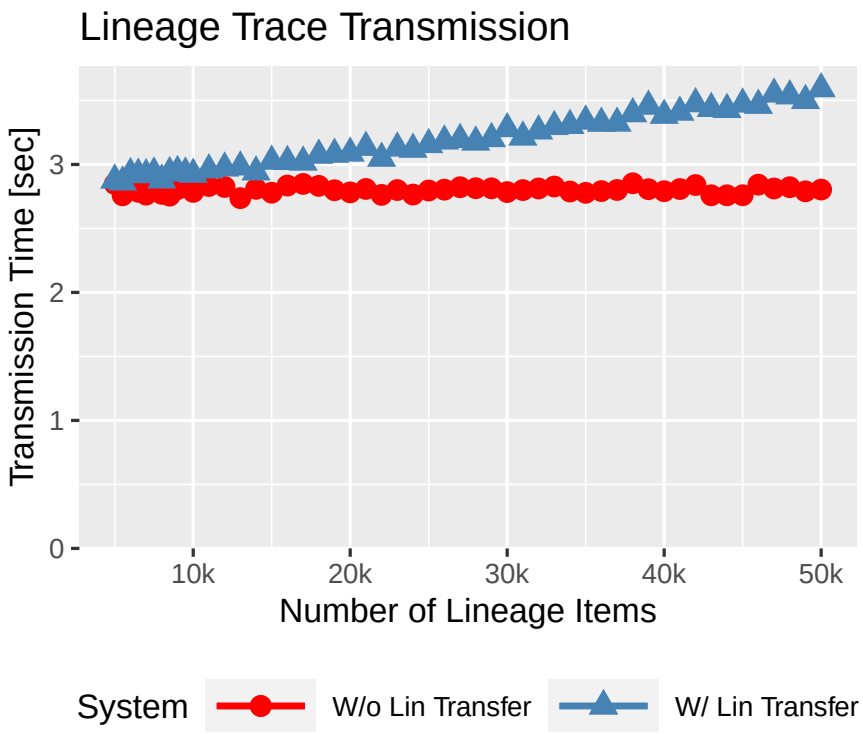


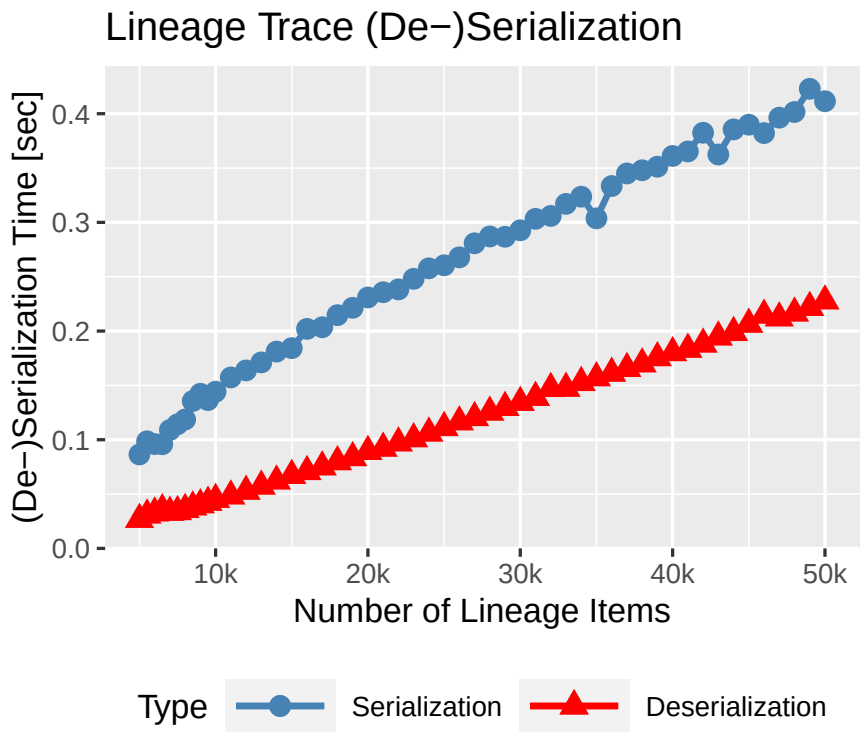


Lineage Trace Transfer

WAN setup	1 coordinator: β-node	1 worker: β-node

- Put matrix with lineage trace to the worker
- Vary number of lineage items

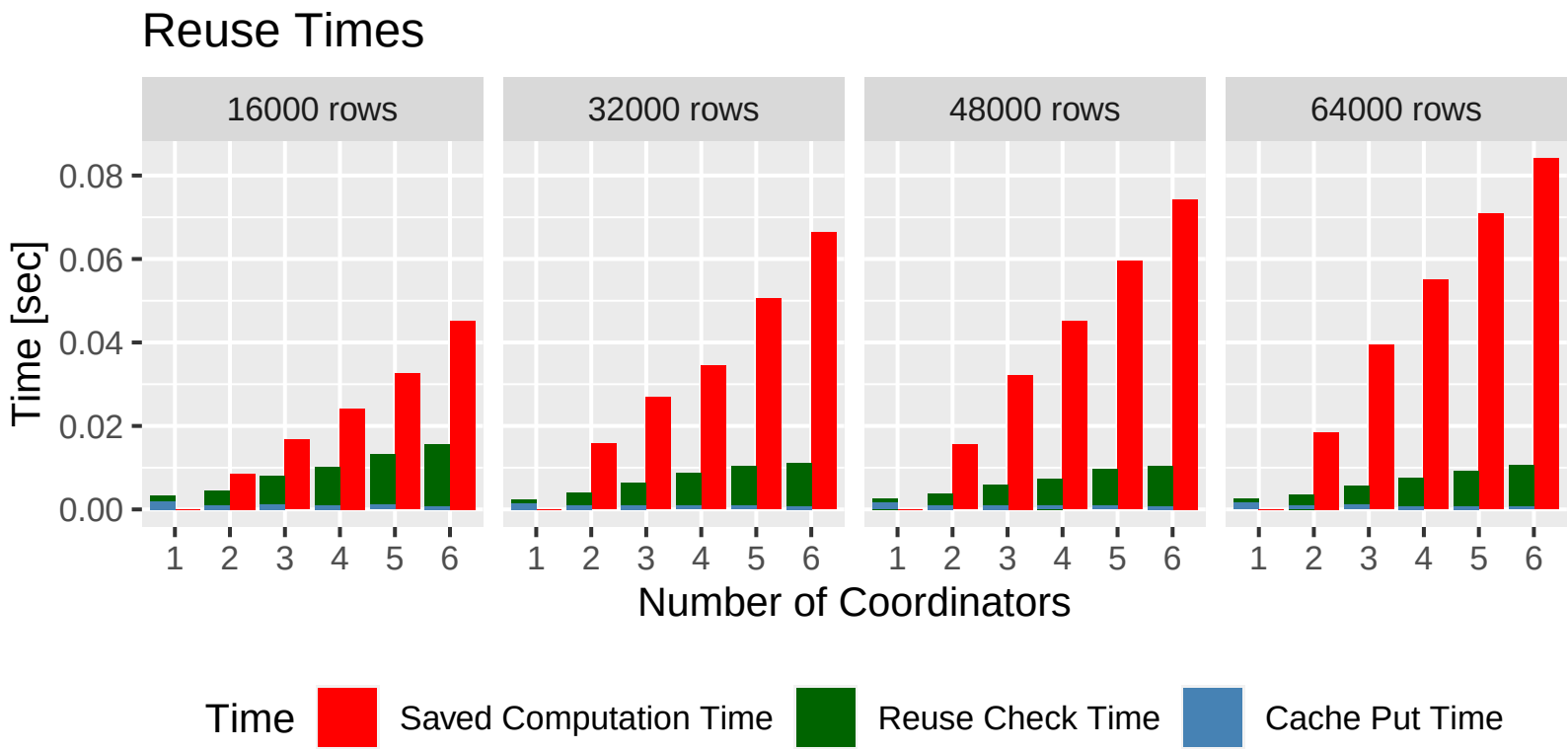




Lineage Reuse at the Federated Worker

Local setup	Coordinators and 1 worker: β-node

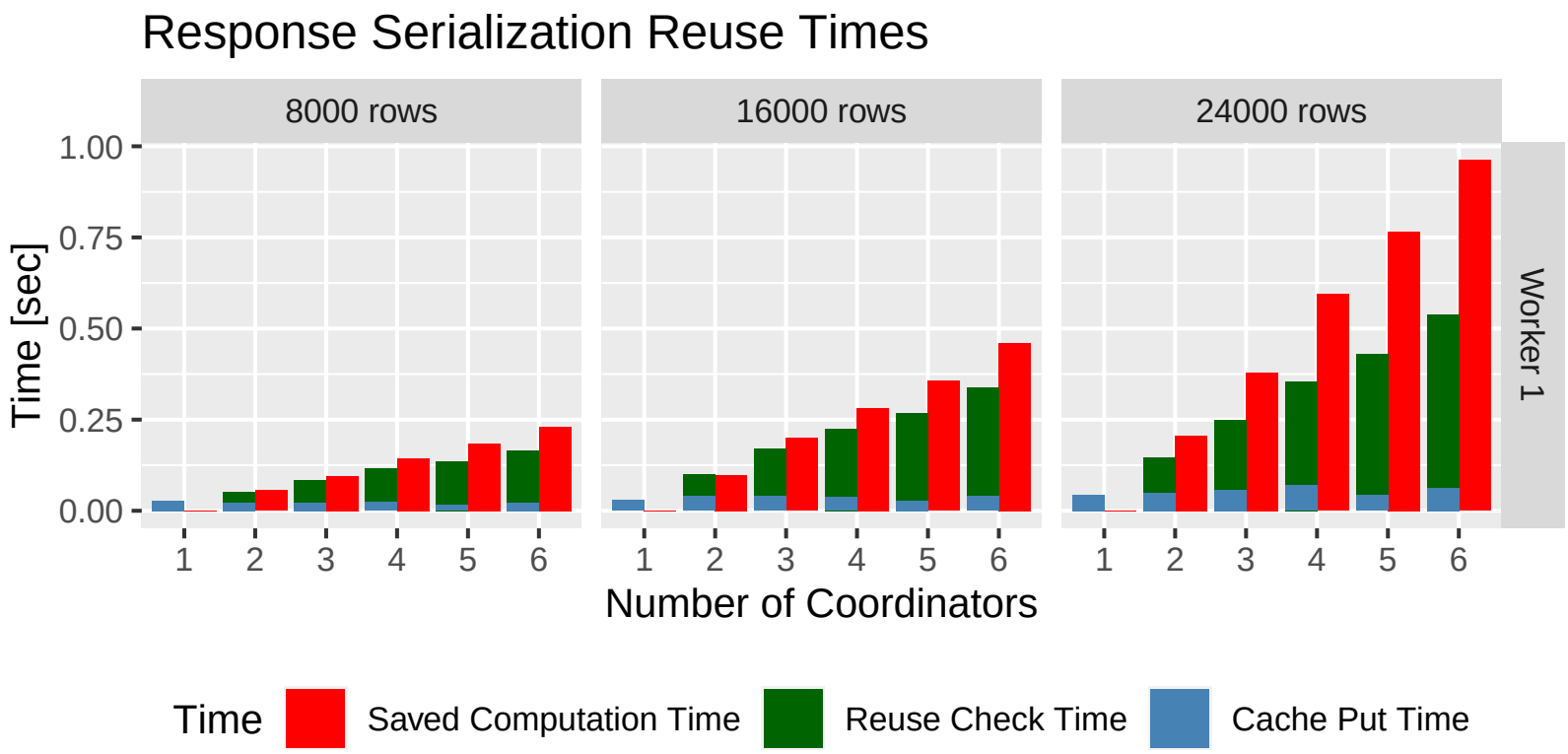
- Train linear regression model on federated data
- Vary the number of coordinators



Response Serialization Reuse

Local setup	Coordinators and 2 workers: α-node

- PCA followed by DBScan with a federated matrix
- Vary the number of coordinators



Hyper-parameter Tuning Performance

LAN setup	4 coordinators: β-node	1 worker: β-node

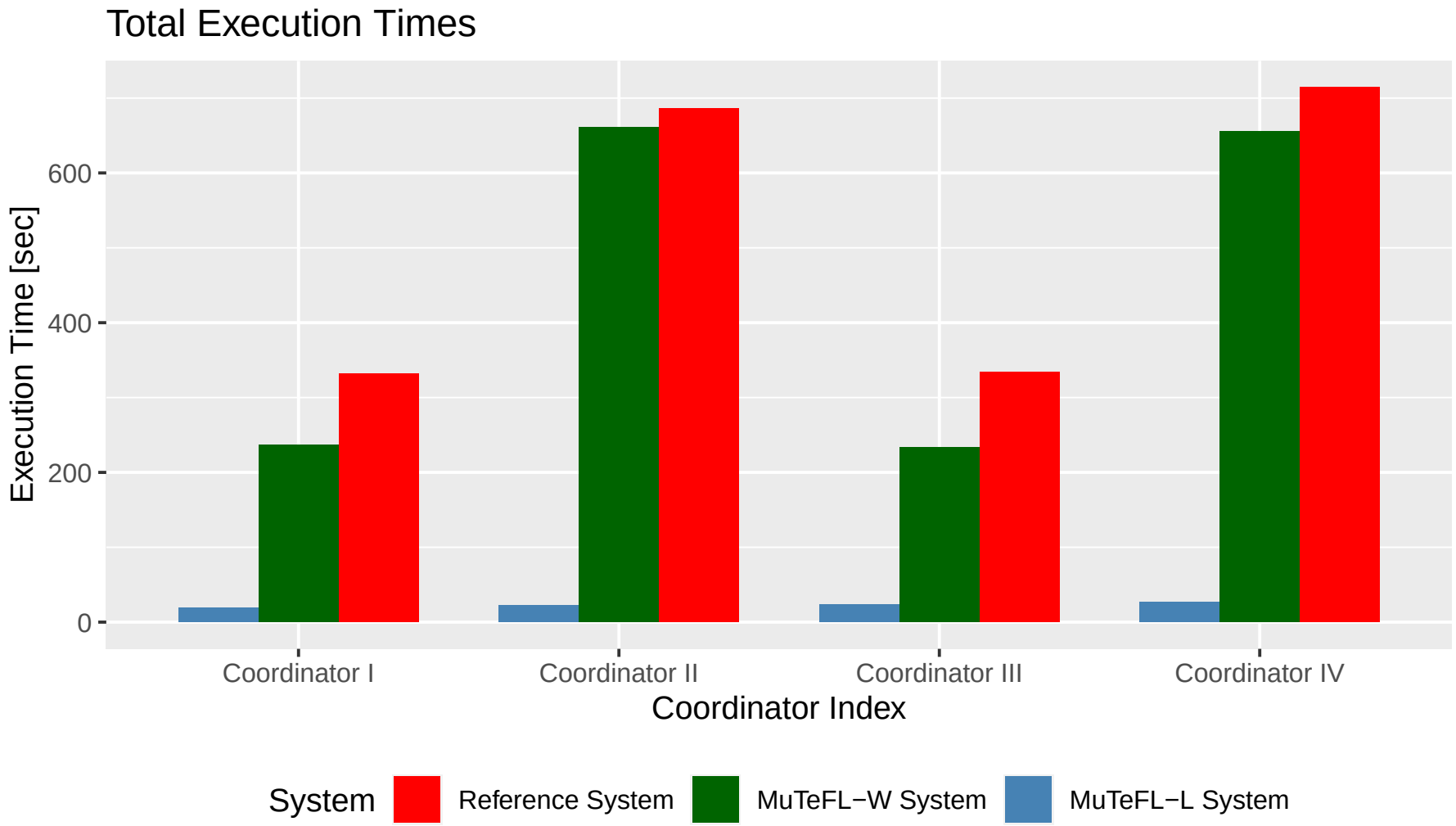
- Hyper-parameter tuning of a linear regression model

Coordinator	λ	intercept	tolerance	k (PCA)
I	$[10^{-9}, 10^{-2}]$	$[0, 6]$	$[10^{-12}, 10^{-6}]$	N/A
II	$[10^{-3}, 10^{0}]$	$[0, 2]$	$[10^{-12}, 10^{-9}]$	$\{300, 400, 500\}$
III	$[10^{-7}, 10^{0}]$	$[-3, 3]$	$[10^{-10}, 10^{-4}]$	N/A
IV	$[10^{-5}, 10^{-2}]$	$[0, 2]$	$[10^{-10}, 10^{-7}]$	$\{250, 500, 750\}$

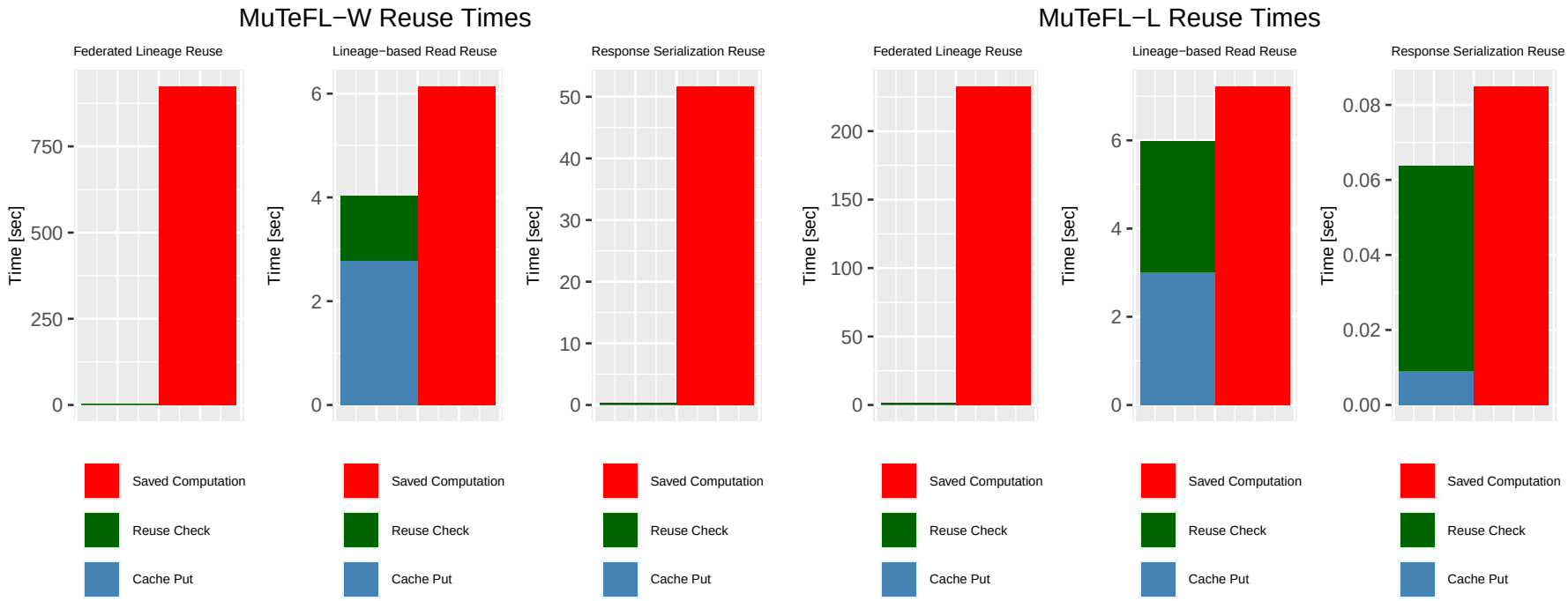
Baselines:

1. *Reference System*: Disable the introduced techniques, except the Netty Buffer Allocation
2. *MuTeFL-W System*: Lineage-based reuse activated only on the federated worker
3. *MuTeFL-L System*: Lineage-based reuse activated on the coordinators and on the federated worker

Hyper-parameter Tuning Performance cont.



Hyper-parameter Tuning Performance cont.



	# Transfers	Serialization	Deserialization
MuTeFL-W	5056	0.7874 sec	1.8708 sec
MuTeFL-L	2191	0.5906 sec	0.8914 sec

Conclusions

- Summary:
  - Tenant Distinction
  - Multi-threaded Event Loop
  - Federated Read Reuse
  - Federated Lineage Reuse
  - Federated Lookup Table
  - Multi-threaded Operator
  - Lineage Trace Transfer
  - Response Serialization Reuse
- Conclusion:
  - Long-running, server-like federated workers
  - Applications with multiple coordinators
  - Experiments show feasibility of a MuTeFL system and usefulness of the introduced optimizations