THE UNIVERSITY of EDINBURGH
informatics

Zeus

Locality-aware distributed transactions

A. Katsarakis[*], Y. Ma[†], Z. Tan[§], A. Bainbridge, M. Balkwill,
A. Dragojevic, B. Grot[*], B. Radunovic, Y. Zhang

[*]University of Edinburgh, [†]Fudan University, [§]UCLA, Microsoft Research

zeus-protocol.com

Thanks to: EPSRC Microsoft Research

Modern distributed datastores

Keep data

in-memory, **replicated**, **sharded**

across nodes of a datacenter

distributed datastore

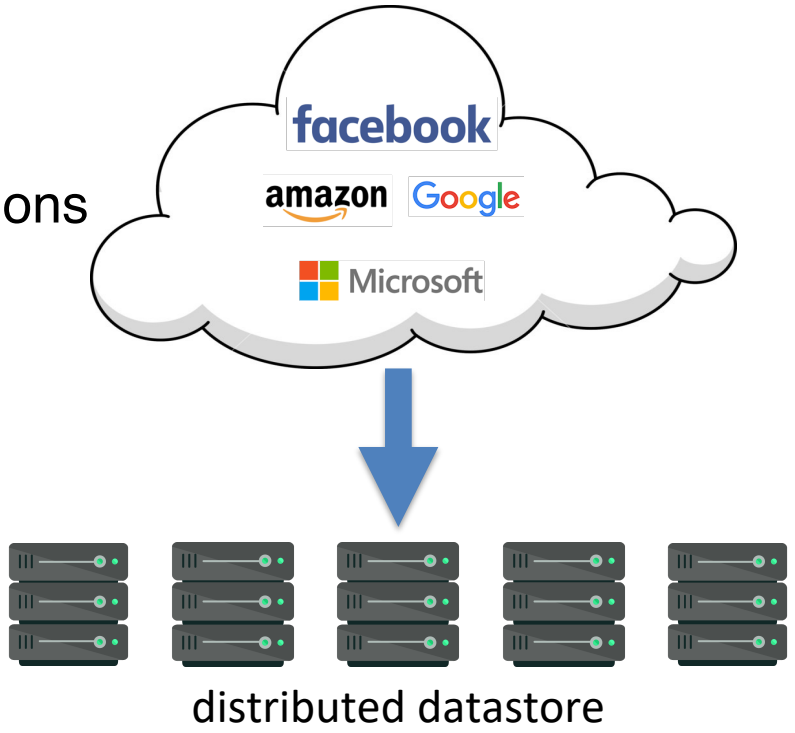
Modern distributed datastores

Keep data

in-memory, **replicated**, **sharded**

across nodes of a datacenter

Backbone of transactional cloud applications

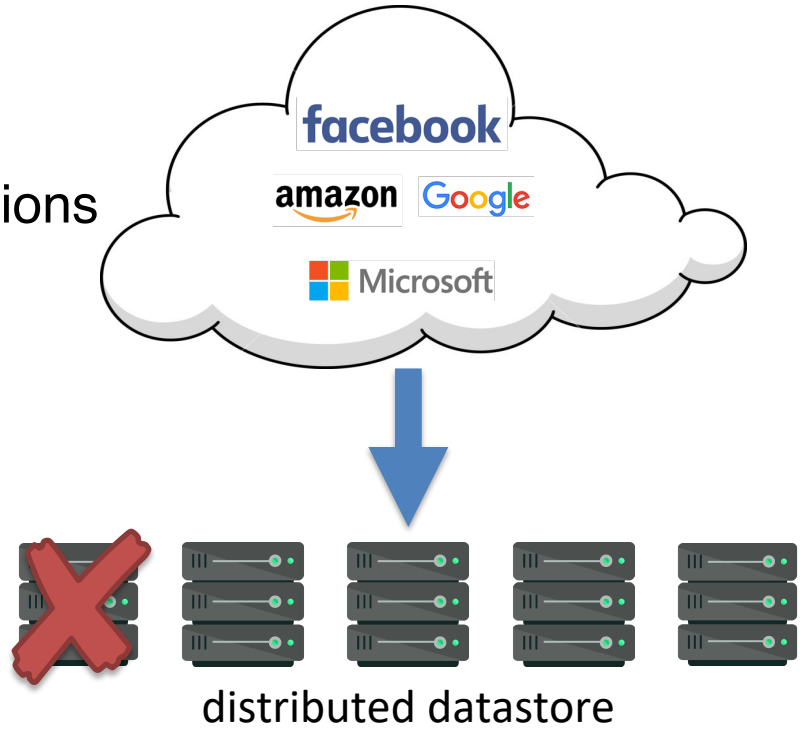


Modern distributed datastores

Keep data
in-memory, **replicated**, **sharded**
across nodes of a datacenter

Backbone of transactional cloud applications

Demand
distributed reliable transactions (txs)
strongly-consistent and fault-tolerant

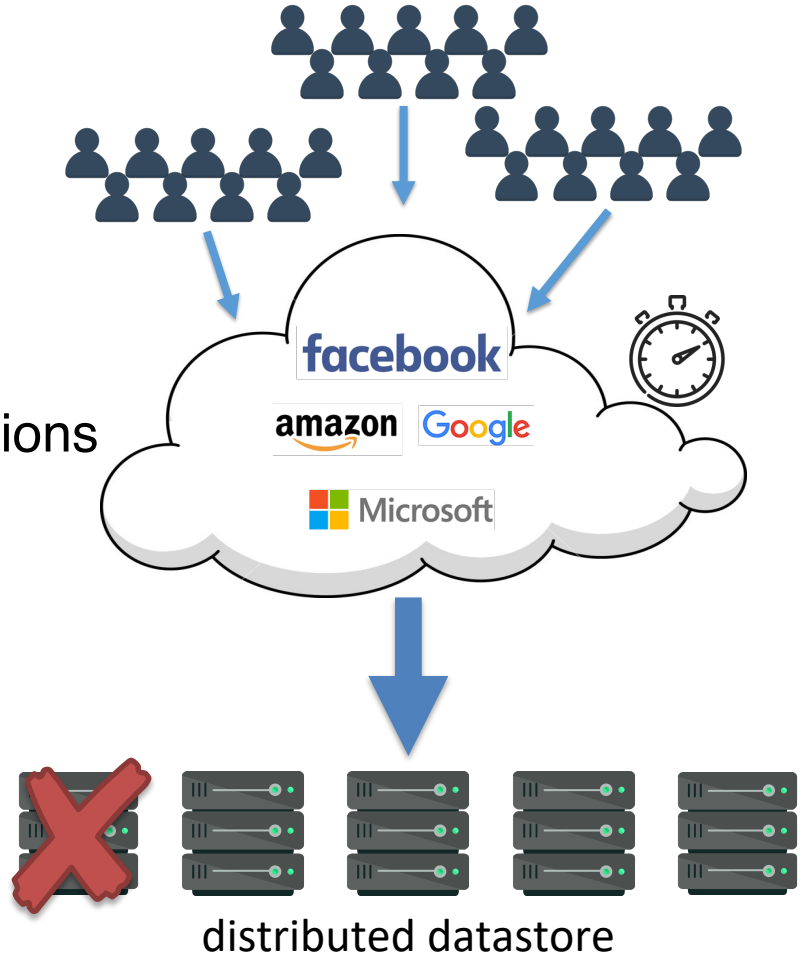


Modern distributed datastores

Keep data
in-memory, **replicated**, **sharded**
across nodes of a datacenter

Backbone of transactional cloud applications

Demand
distributed reliable transactions (txs)
strongly-consistent and fault-tolerant
high performance

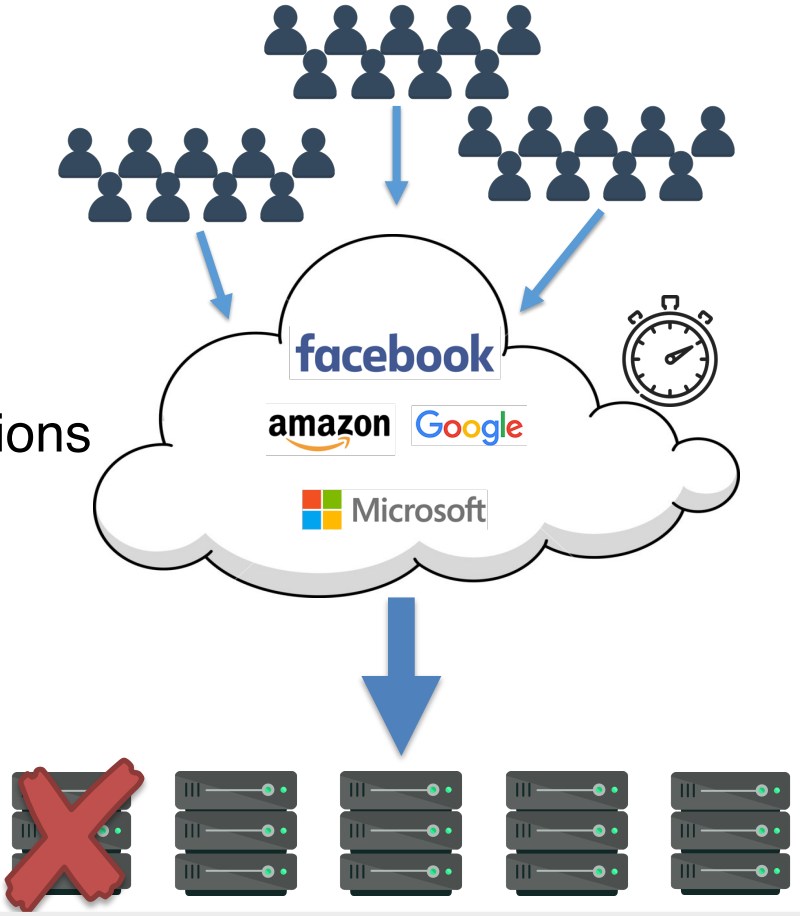


Modern distributed datastores

Keep data
in-memory, **replicated**, **sharded**
across nodes of a datacenter

Backbone of transactional cloud applications

Demand
distributed reliable transactions (txs)
strongly-consistent and fault-tolerant
high performance



Traditional distributed txs well-known as expensive

THE UNIVERSITY of EDINBURGH
informatics

Observation

Many tx applications exhibit **dynamic locality**

network functions, peer-to-peer payments …

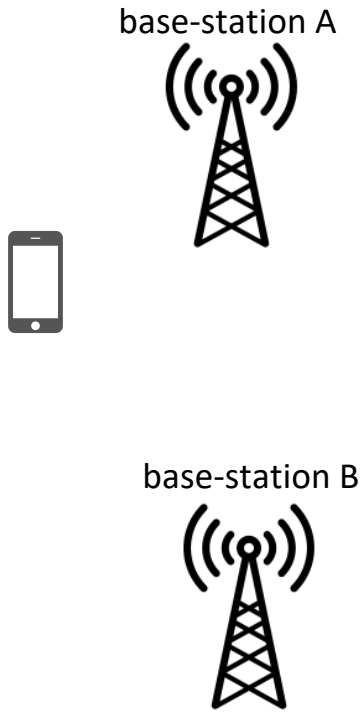
Observation

Many tx applications exhibit **dynamic locality**

network functions, peer-to-peer payments …

Example: cellular control plane

manages phone connectivity and handovers among base stations



Observation

Many tx applications exhibit **dynamic locality**

network functions, peer-to-peer payments …

Example: cellular control plane

manages phone connectivity and handovers among base stations

Locality

every phone user repeats txs:
same phone & nearest base-station

Observation

Many tx applications exhibit **dynamic locality**

network functions, peer-to-peer payments …

Example: cellular control plane

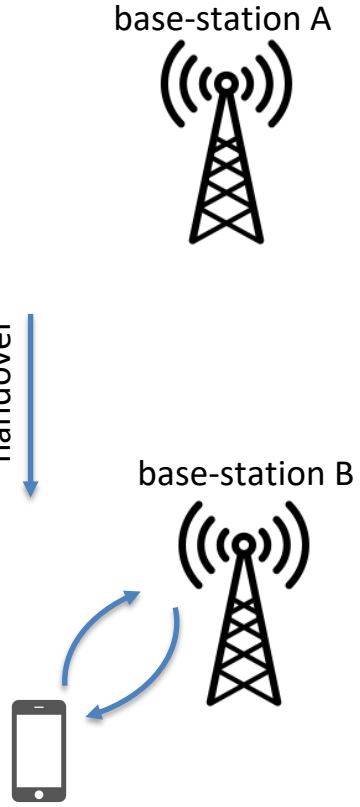
manages phone connectivity and handovers among base stations

Locality

every phone user repeats txs:
same phone & nearest base-station

But locality is dynamic

changes at run-time
e.g., user commutes → base-station changes



Observation

Many tx applications exhibit **dynamic locality**

network functions, peer-to-peer payments …

Example: cellular control plane

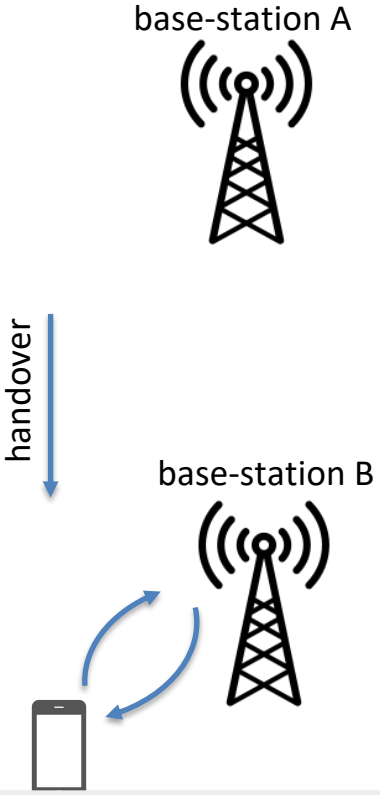
manages phone connectivity and handovers among base stations

Locality

every phone user repeats txs: same phone & nearest base-station

But locality is dynamic

changes at run-time



Can state-of-the-art datastores exploit dynamic locality?

State-of-the-art reliable datastores

Static sharding (e.g., consistent hashing)

Objects placed randomly on fixed nodes

- easy to locate and access objects
- reliable txs regardless of access pattern

State-of-the-art reliable datastores

Static sharding (e.g., consistent hashing)

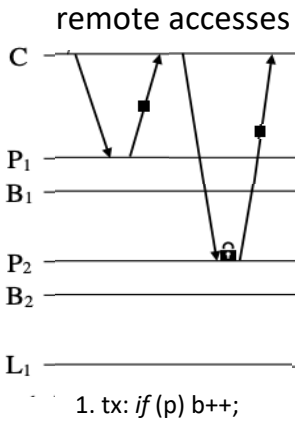
Objects placed randomly on fixed nodes

- easy to locate and access objects
- reliable txs regardless of access pattern

- **expensive reliable txs**

  mostly **remote accesses**

  some blocking (control flow, pointer chasing)



1. tx: *if* (p) b++;

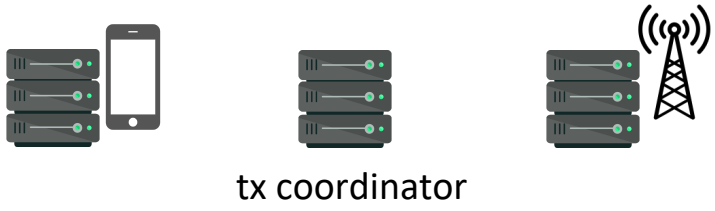
Adapted from *FaSST* [OSDI'16]

State-of-the-art reliable datastores

Static sharding (e.g., consistent hashing)

Objects placed randomly on fixed nodes

- easy to locate and access objects
- reliable txs regardless of access pattern



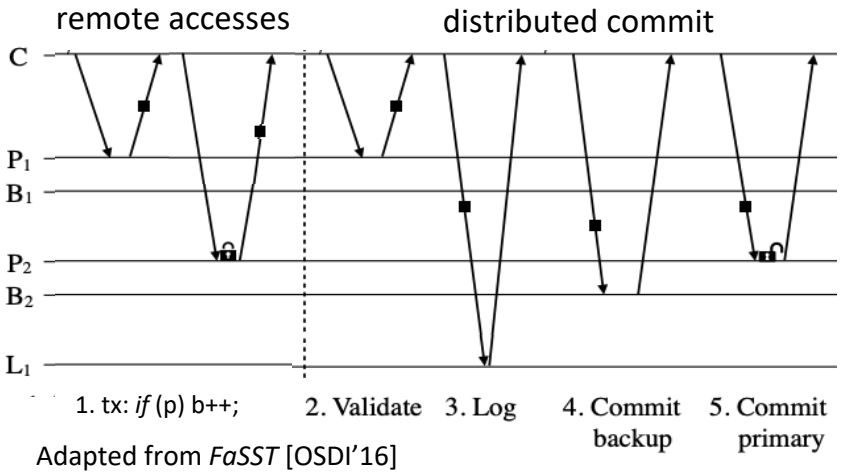
expensive reliable txs

mostly **remote accesses**

some blocking (control flow, pointer chasing)

related objects on different shards

costly distributed commit



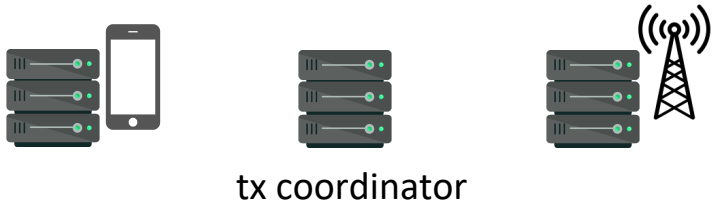
Adapted from *FaSST* [OSDI'16]

State-of-the-art reliable datastores

Static sharding (e.g., consistent hashing)

Objects placed randomly on fixed nodes

- easy to locate and access objects
- reliable txs regardless of access pattern



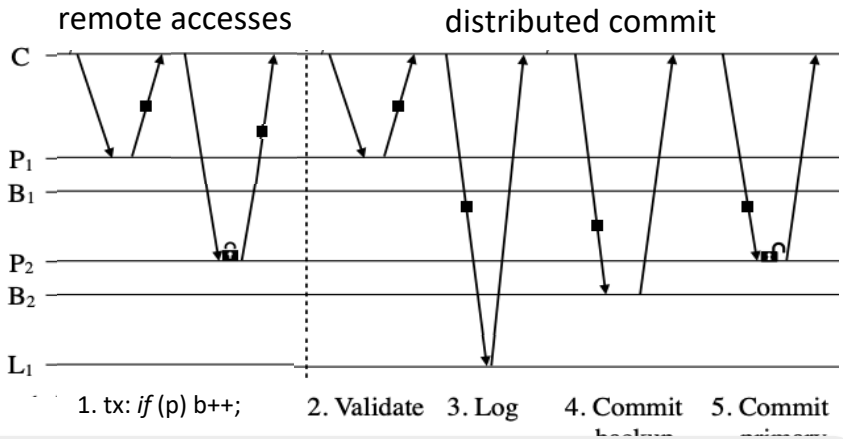
expensive reliable txs

mostly **remote accesses**

some blocking (control flow, pointer chasing)

related objects on different shards

costly distributed commit



Cannot exploit locality → expensive reliable txs

Enter Zeus

Distributed datastore: exploits locality for fast reliable txs

Inspired by multiprocessor's hardware transactional memory

Enter Zeus

Distributed datastore: exploits locality for fast reliable txs
Inspired by multiprocessor's hardware transactional memory

Basic idea 

Each object has a single node **owner** = **data** + **exclusive write access**
the **owner changes dynamically** and is tracked by replicated directory

Enter Zeus

Distributed datastore: exploits locality for fast reliable txs
Inspired by multiprocessor's hardware transactional memory

Basic idea 

Each object has a single node **owner** = **data** + **exclusive write access**
the **owner changes dynamically** and is tracked by replicated directory

Coordinator executes a tx by **acquiring ownership** of all its objects
→ **single-node commit**

Enter Zeus

Distributed datastore: exploits locality for fast reliable txs
Inspired by multiprocessor's hardware transactional memory

Basic idea 

Each object has a single node **owner** = **data** + **exclusive write access**
the **owner changes dynamically** and is tracked by replicated directory

Coordinator executes a tx by **acquiring ownership** of all its objects
→ **single-node commit**

Ownership stays with coordinator
→ future txs on these objects enjoy **local accesses**

THE UNIVERSITY of EDINBURGH
informatics

Enter Zeus

Distributed datastore: exploits locality for fast reliable txs
Inspired by multiprocessor's hardware transactional memory

Basic idea 

Each object has a single node **owner** = **data** + **exclusive write access**
the **owner changes dynamically** and is tracked by replicated directory

Coordinator executes a tx by **acquiring ownership** of all its objects
→ **single-node commit**

Ownership stays with coordinator
→ future txs on these objects enjoy **local accesses**

What are the exact steps?

THE UNIVERSITY of EDINBURGH
informatics

Locality-aware txs in Zeus

1.

2.

3.

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

2.

3.

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

How to get ownership reliably?

2.

3.

Ownership protocol

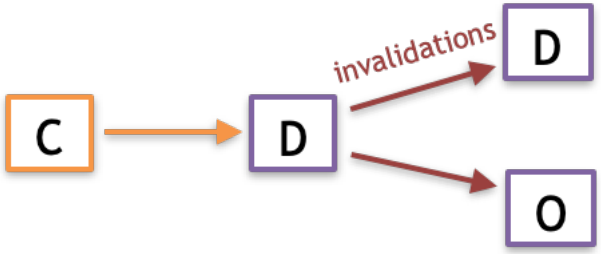
1) Coordinator gets ownership from current owner

Ownership protocol

1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

Ownership protocol

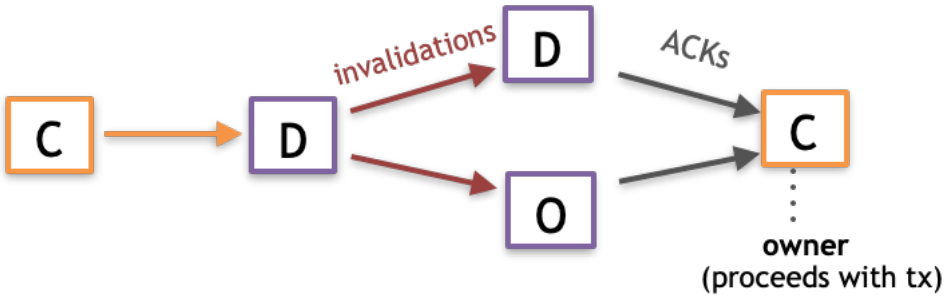


1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters

Ownership protocol

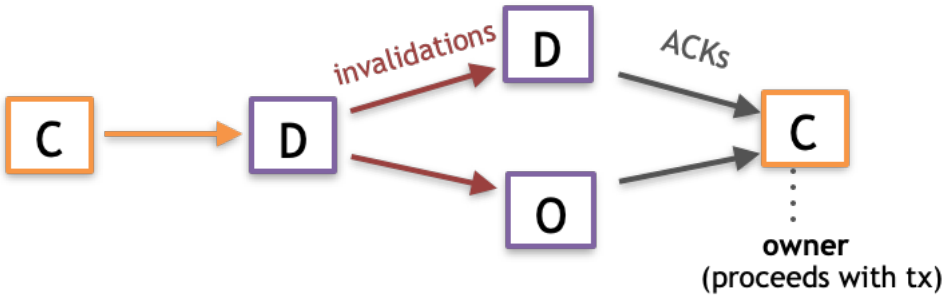


1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters
2. Arbiters **acknowledge** the coordinator directly

Ownership protocol



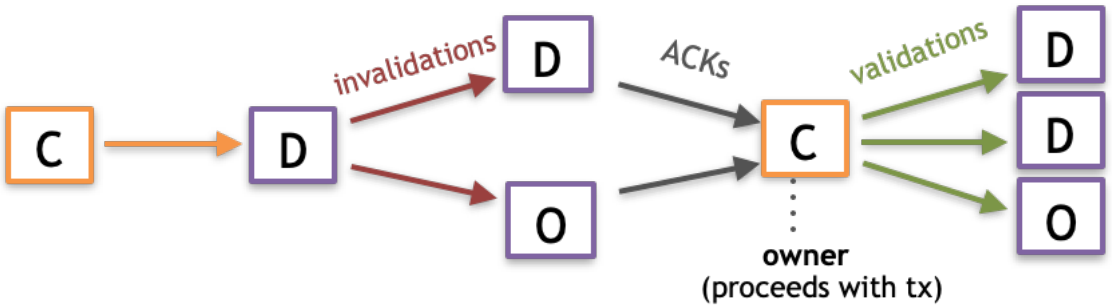
1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters
2. Arbiters **acknowledge** the coordinator directly

Ownership is acquired & coordinator proceeds with tx ✓

Ownership protocol

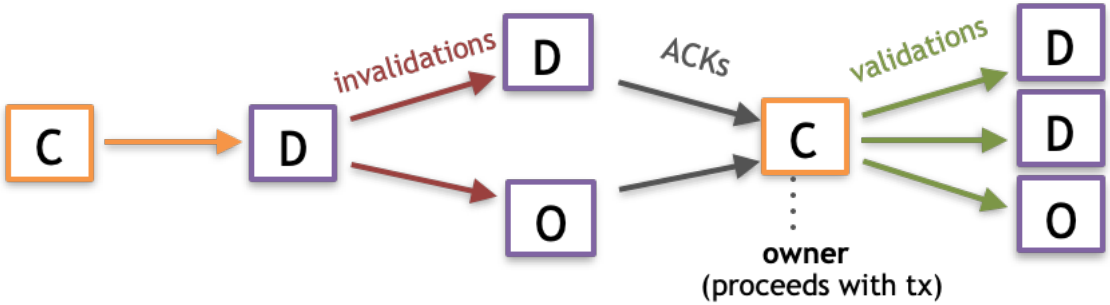


1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters
2. Arbiters **acknowledge** the coordinator directly
3. Coordinator sends **validations** informing arbiters for acquisition

Ownership protocol



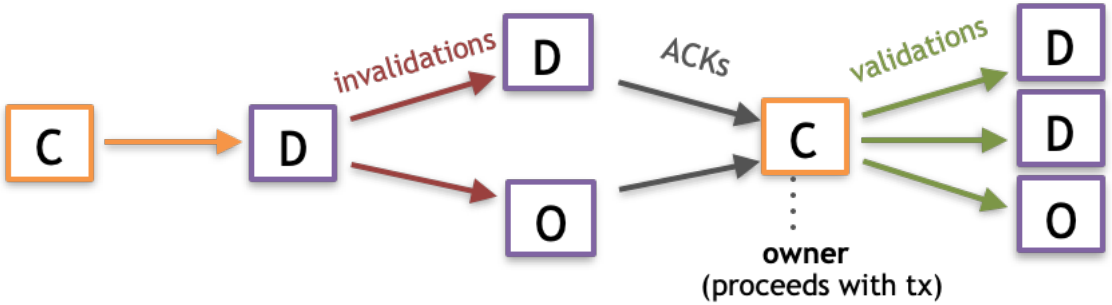
1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters
2. Arbiters **acknowledge** the coordinator directly
3. Coordinator sends **validations** informing arbiters for acquisition

Conflicts: logical timestamps, **fault tolerance**: idempotent replays
as in Hermes [ASPLOS'20]

Ownership protocol



1) Coordinator gets ownership from current owner
2) Keeps consistent directory replicas

arbiters

1. Coordinator sends object ownership **invalidations** (through a directory replica) to all arbiters
2. Arbiters **acknowledge** the coordinator directly
3. Coordinator sends **validations** informing arbiters for acquisition

Conflicts: logical timestamps, **fault tolerance**: idempotent replays

✓ Correctness verified under conflicts and faults

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

Locality + ownership stays with coordinator

Locality-aware txs in Zeus

1. **Execute as the owner**
   a) at object access: **if (not owner) get ownership**
   b) **local access**

common case

Locality + ownership stays with coordinator

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

b) **local access**

common case

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

b) **local access**

common case

2. Local commit

commits tx: traditional single-node commit

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

b) **local access**

common case

2. Local commit

commits tx: traditional single-node commit

3. Reliable commit

completes tx: updating replicas

Locality-aware txs in Zeus

1. Execute as the owner

a) at object access: **if (not owner) get ownership**

b) **local access**

common case

2. Local commit

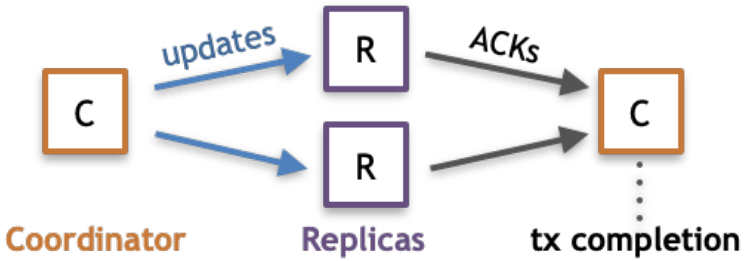
commits tx: traditional single-node commit

3. Reliable commit

completes tx: updating replicas

Great! But how efficient is reliable commit?

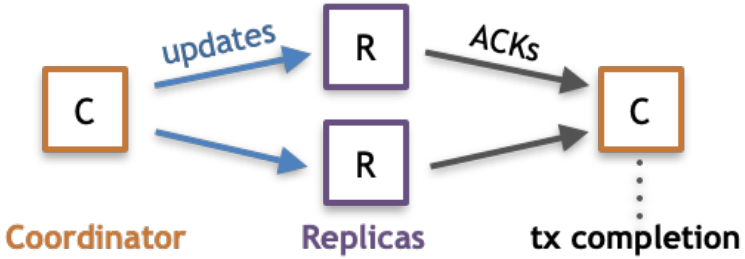
Reliable commit



1. Committed tx → no conflicts → fast tx completion

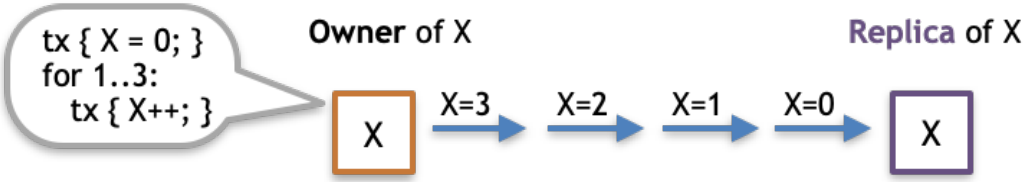
- coordinator sends **updates** to replicas and waits for **ACKs**
- read-only txs: no updates → no reliable commit

Reliable commit



1. Committed tx → no conflicts → fast tx completion

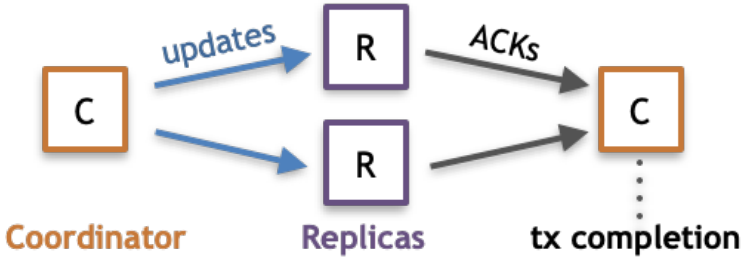
- coordinator sends **updates** to replicas and waits for **ACKs**
- read-only txs: no updates → no reliable commit



2. No conflicts → no aborts → pipelined txs (no waiting for replication)

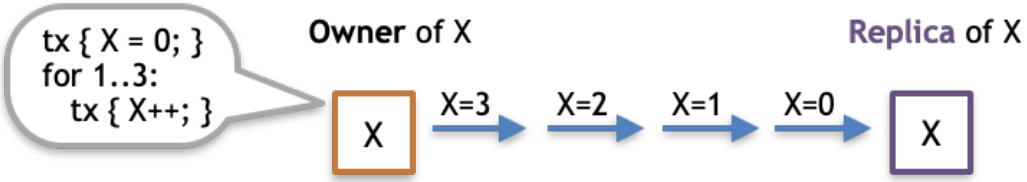
- subsequent txs use local state with certainty & issue **updates**
- coordinator sequences **updates**, which replicas apply in order

Reliable commit



1. Committed tx → no conflicts → fast tx completion

- coordinator sends **updates** to replicas and waits for **ACKs**
- read-only txs: no updates → no reliable commit

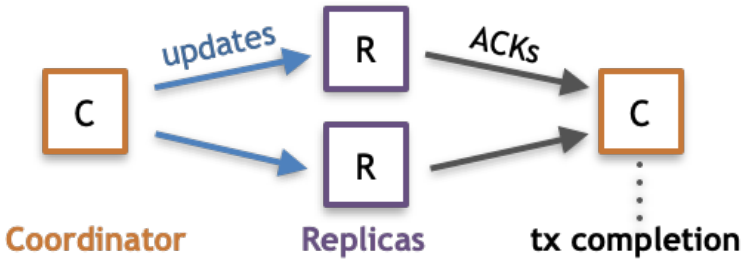


2. No conflicts → no aborts → pipelined txs (no waiting for replication)

- subsequent txs use local state with certainty & issue **updates**
- coordinator sequences **updates**, which replicas apply in order

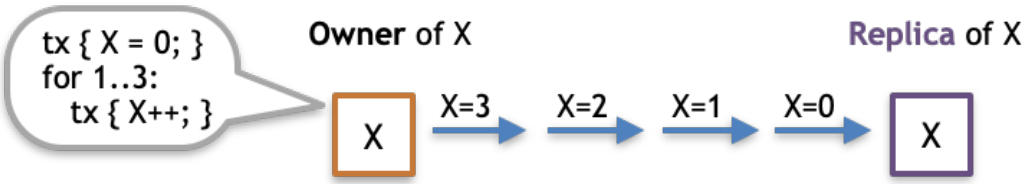
Fault tolerance: idempotent replays

Reliable commit



1. Committed tx → no conflicts → fast tx completion

- coordinator sends **updates** to replicas and waits for **ACKs**
- read-only txs: no updates → no reliable commit



2. No conflicts → no aborts → pipelined txs (no waiting for replication)

- subsequent txs use local state with certainty & issue **updates**
- coordinator sequences **updates**, which replicas apply in order

✓Very efficient! ✓Correctness verified under faults

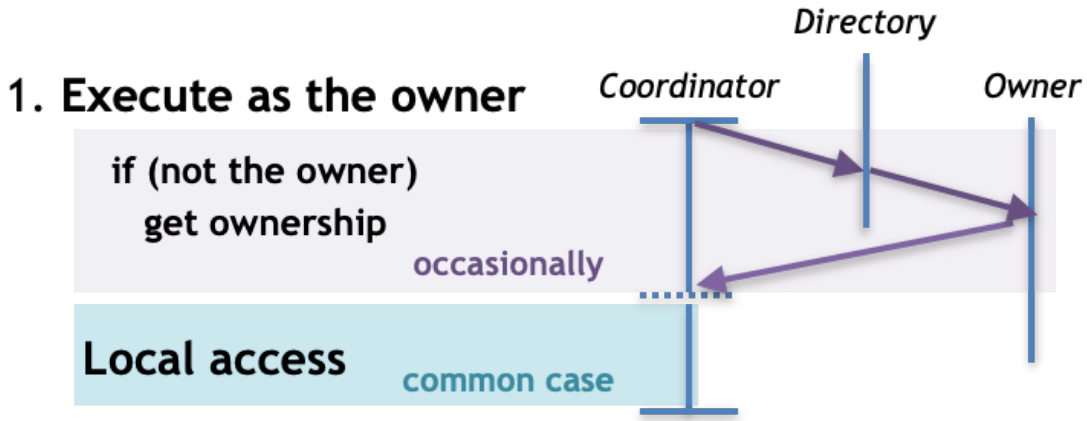
THE UNIVERSITY of EDINBURGH
informatics

Recap: txs in Zeus

Locality-aware, distributed and reliable

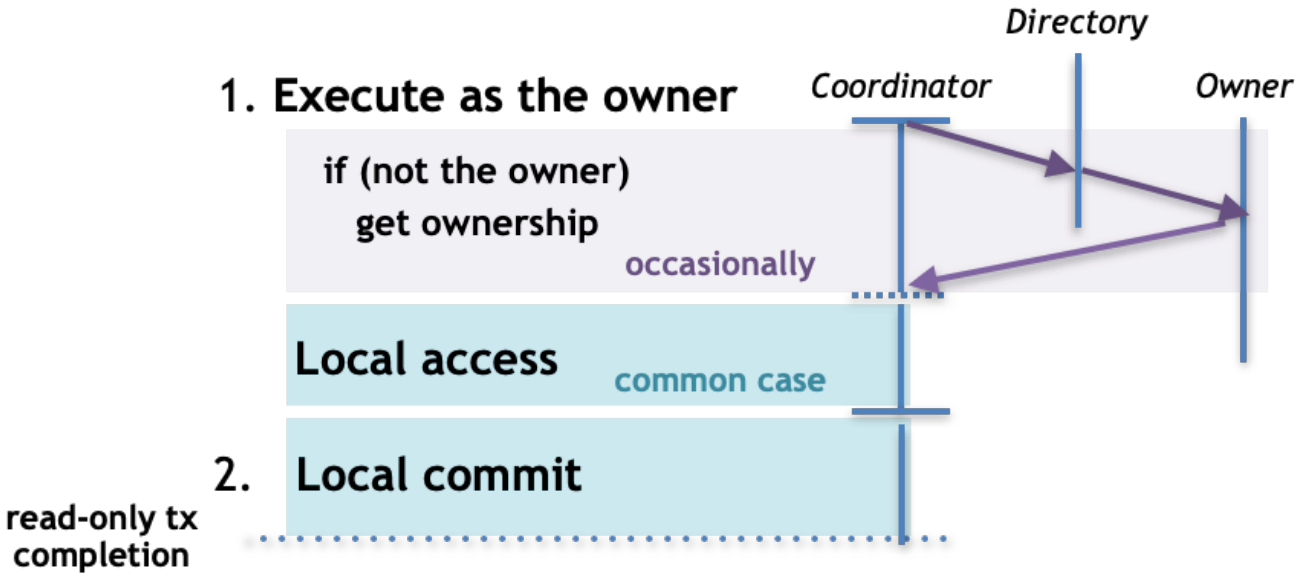
Recap: txs in Zeus

Locality-aware, distributed and reliable



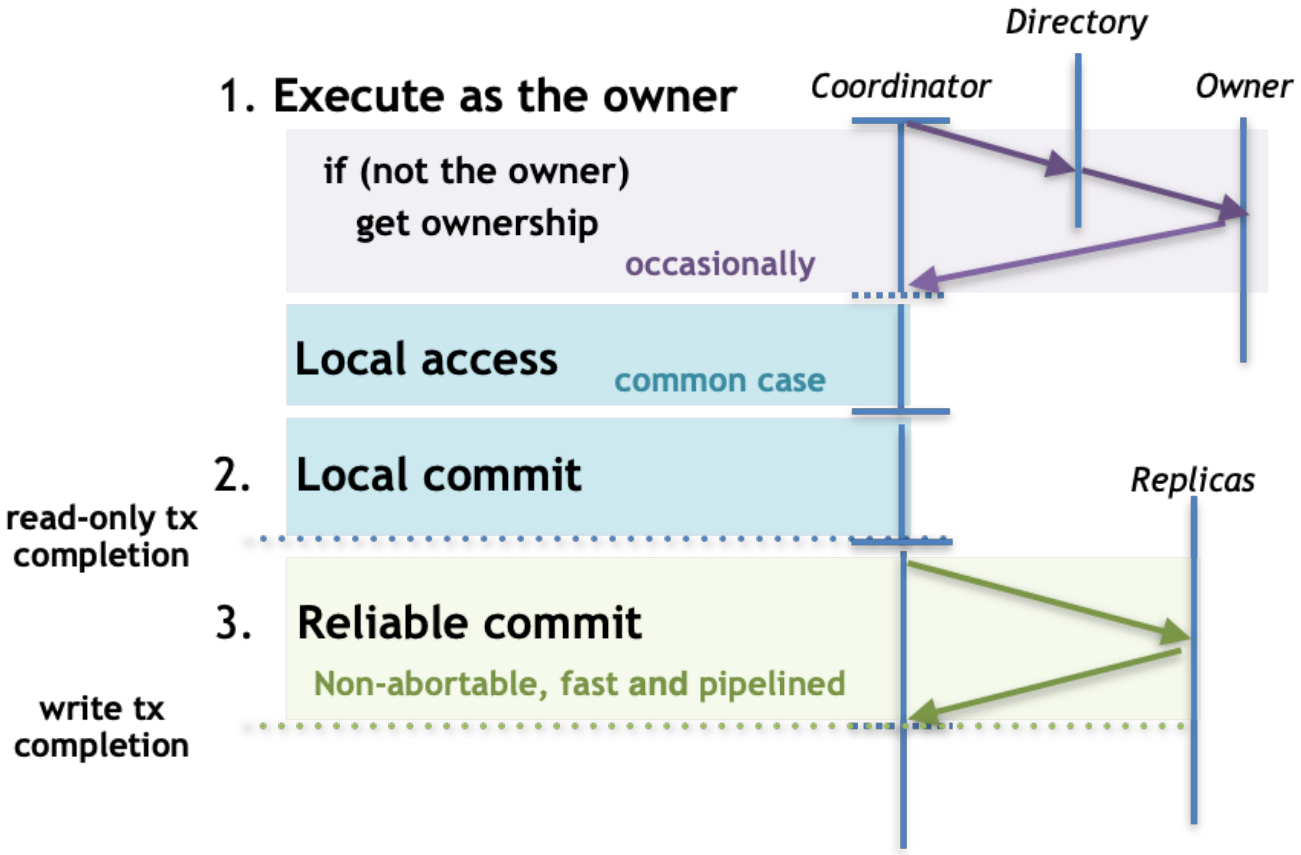
Recap: txs in Zeus

Locality-aware, distributed and reliable



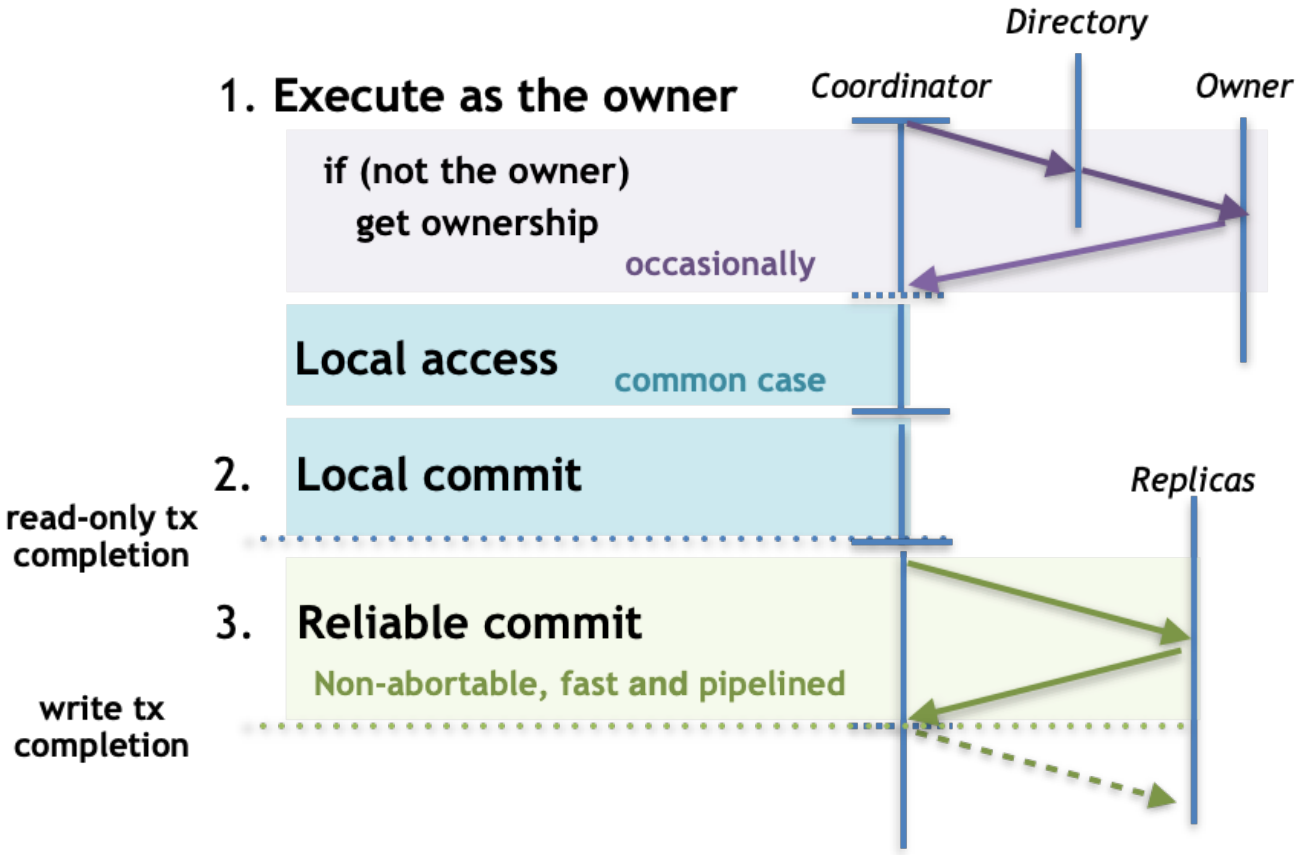
Recap: txs in Zeus

Locality-aware, distributed and reliable



Recap: txs in Zeus

Locality-aware, distributed and reliable

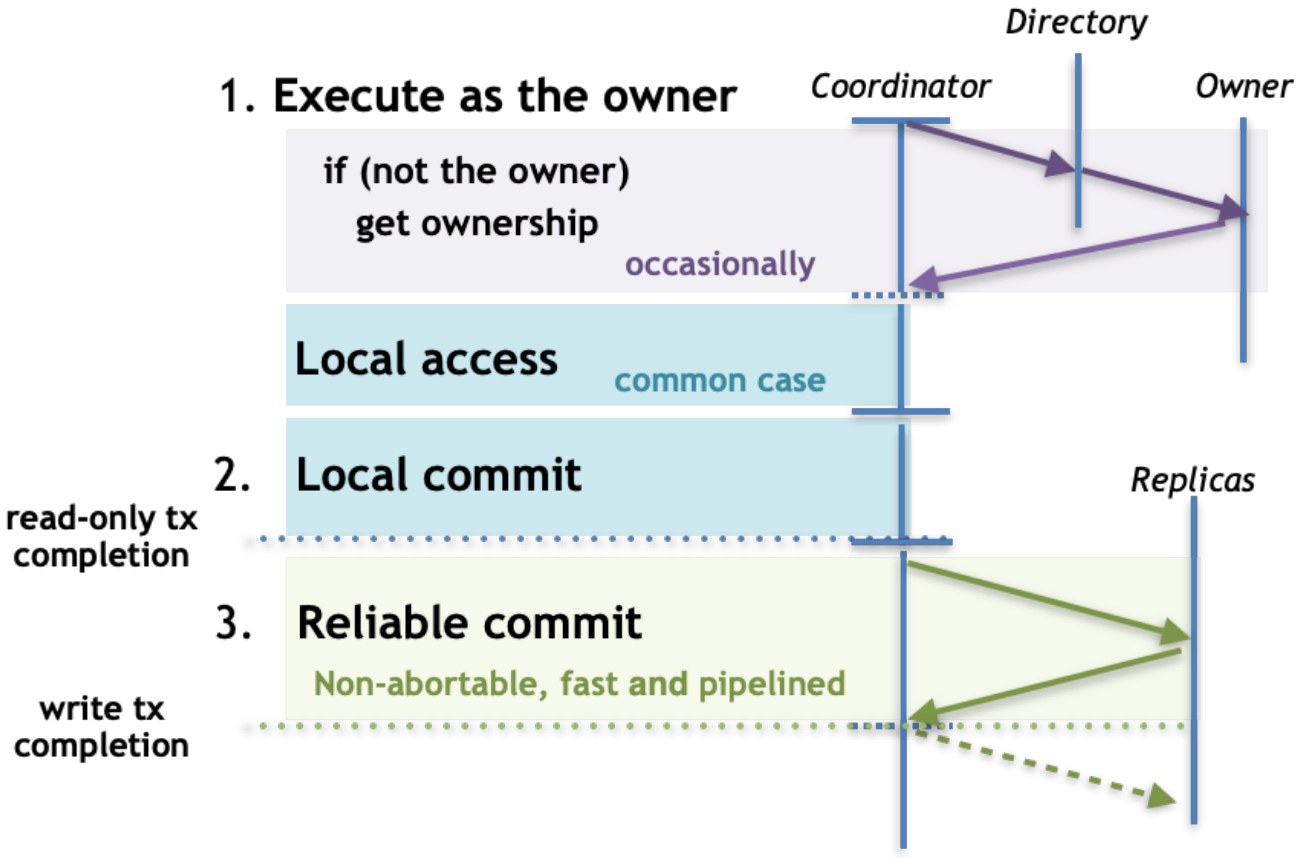


Local read-only txs at all replicas


Zeus: Locality-aware Distributed Transactions

Recap: txs in Zeus

Locality-aware, distributed and reliable



Local read-only txs at all replicas


Zeus: Locality-aware Distributed Transactions

Awesome! Does it translate into performance?

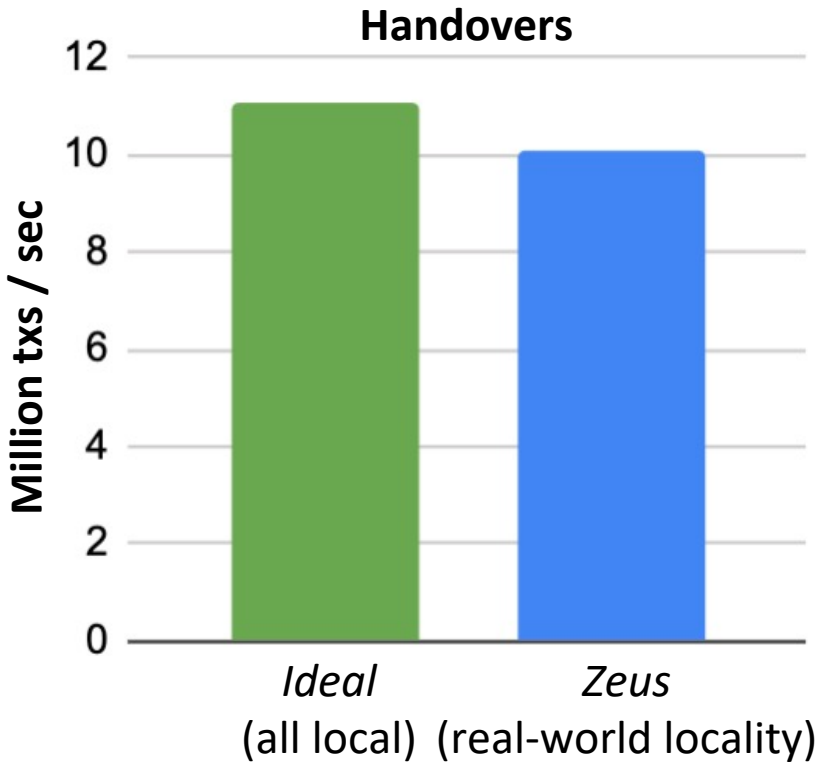
THE UNIVERSITY of EDINBURGH
informatics

Performance

6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)

Performance

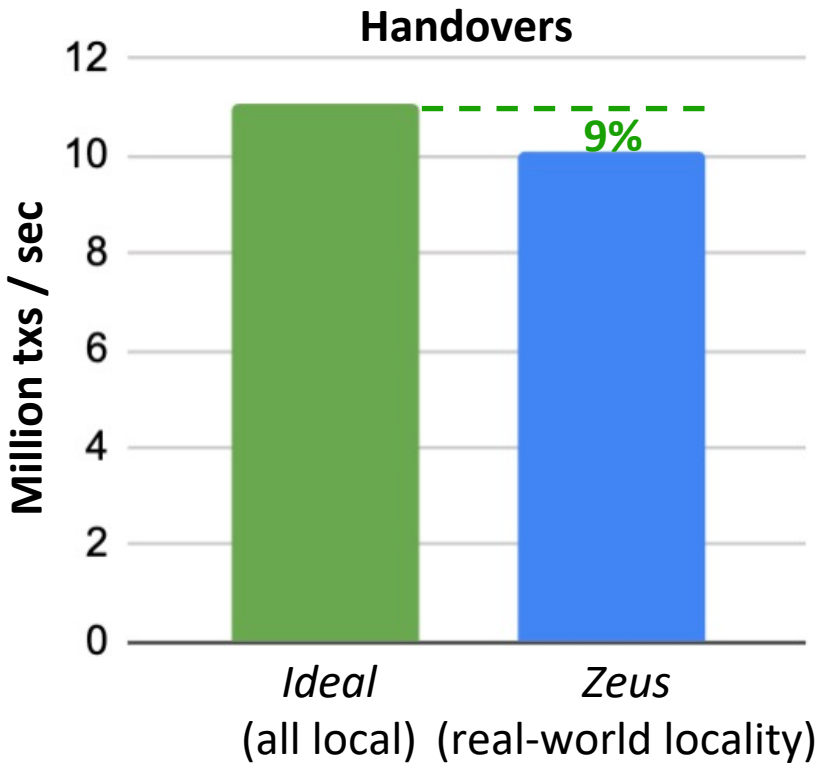
6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)



THE UNIVERSITY of EDINBURGH
informatics

Performance

6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)

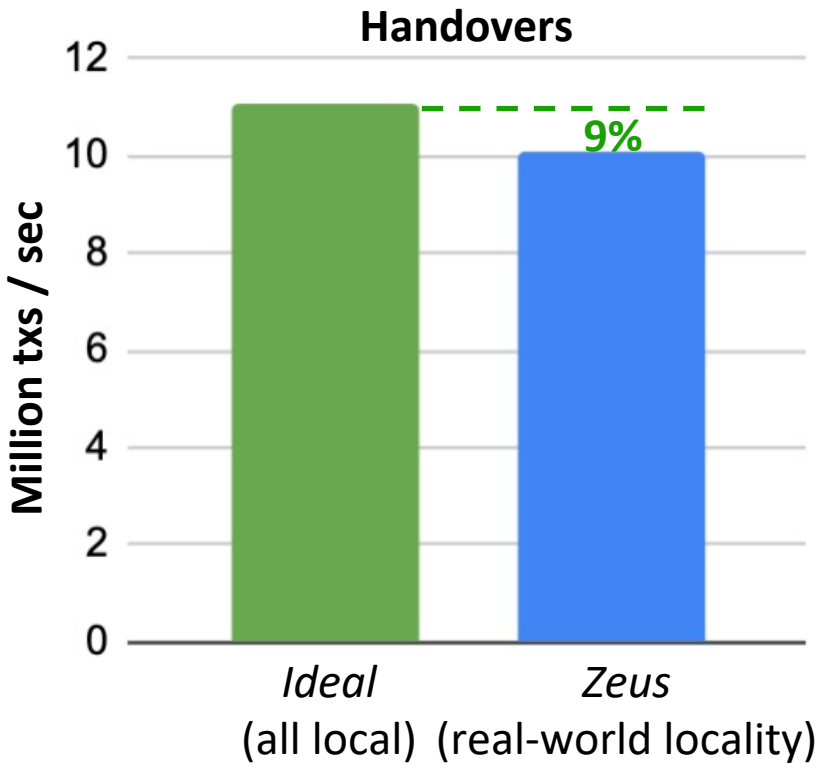


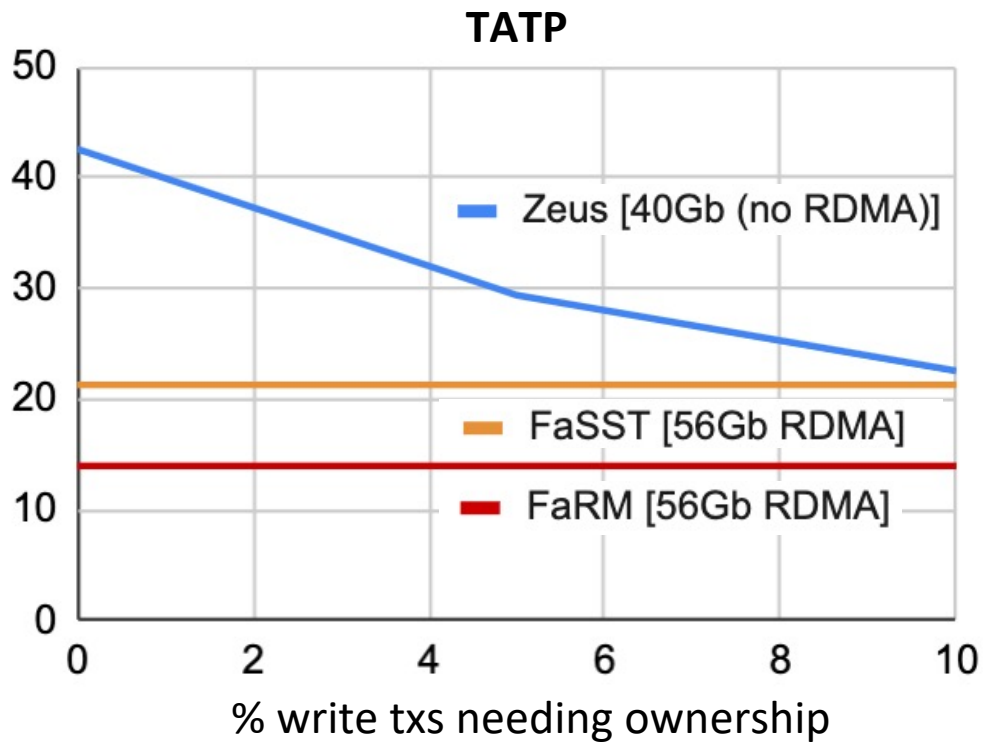
Zeus: **within 9% of ideal**

THE UNIVERSITY of EDINBURGH
informatics

Performance

6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)

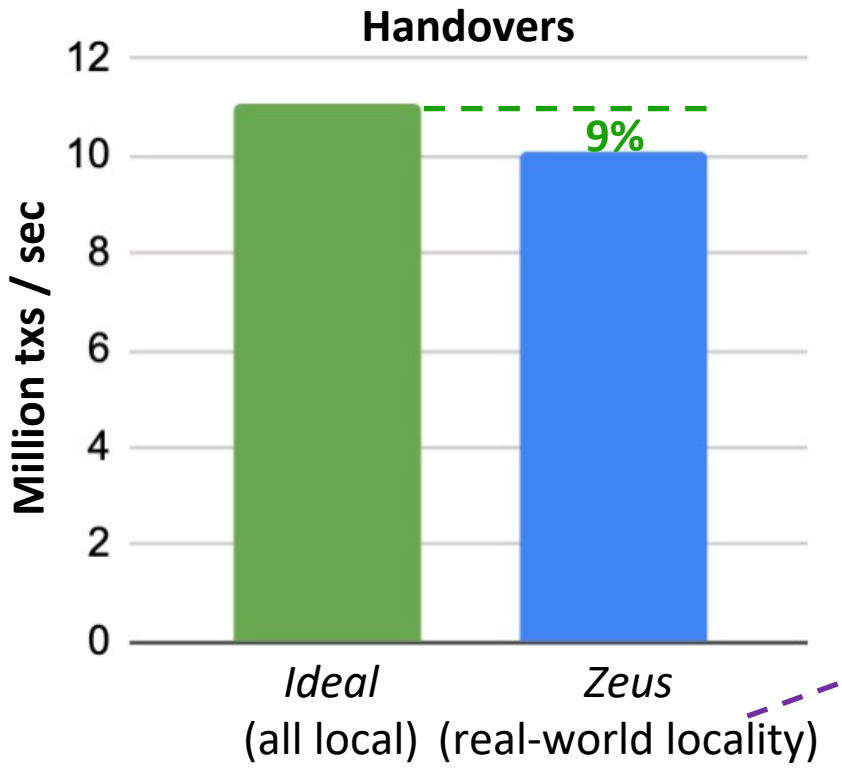


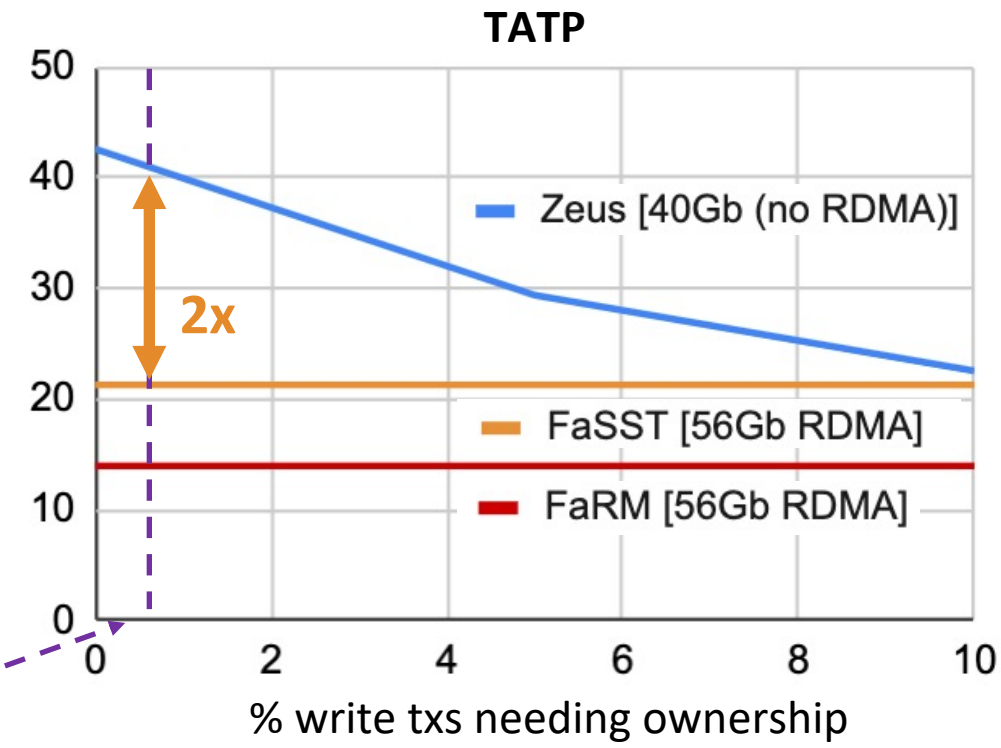


Zeus: **within 9% of ideal**

Performance

6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)





Zeus: **within 9% of ideal**

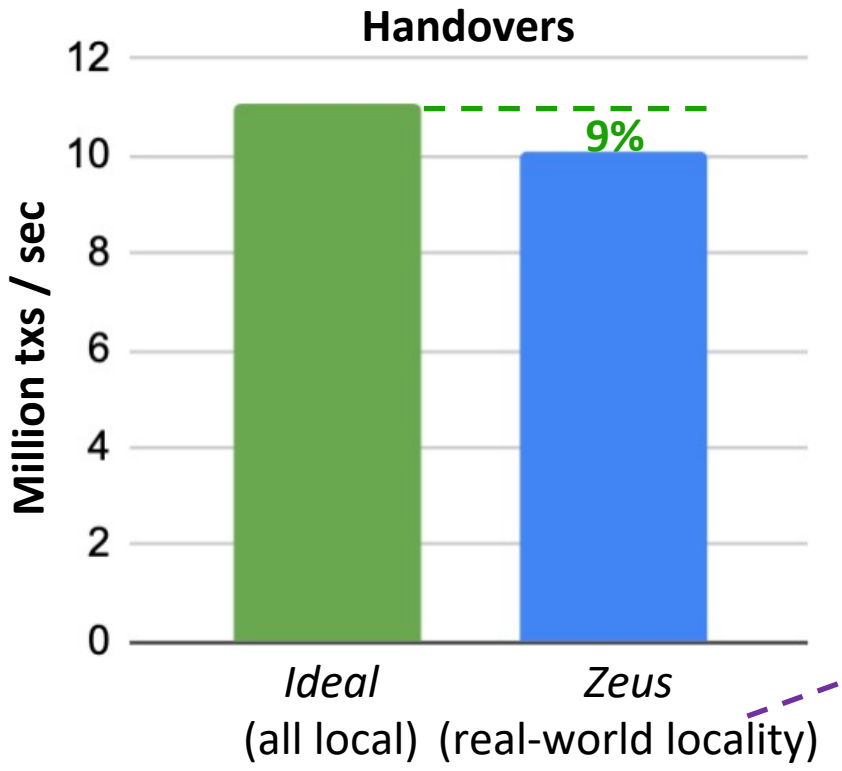
Up to 40M.tx/s and **2x state-of-the-art**

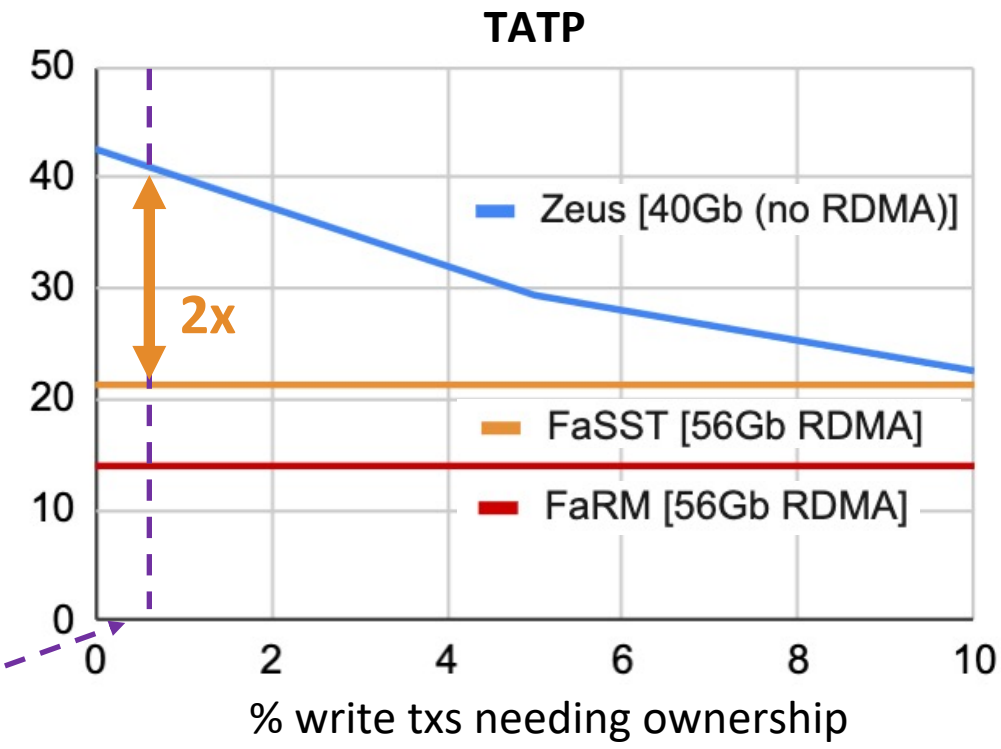
FaSST [OSDI'16], *FaRM* [SOSP'15]
which use 56Gb RDMA

THE UNIVERSITY of EDINBURGH
informatics

Performance

6 nodes, 3-way replication,
Zeus 40Gb (no RDMA)





Zeus: **within 9% of ideal**

Up to 40M.tx/s and **2x state-of-the-art**

Paper: more benchmarks, ownership, latency ...

THE UNIVERSITY of EDINBURGH
informatics

Conclusion

State-of-the-art reliable txs over **static sharding**:

cannot exploit dynamic locality

remote accesses

costly **distributed commit**

Conclusion

State-of-the-art reliable txs over **static sharding**:

cannot exploit dynamic locality

remote accesses

costly **distributed commit**

Zeus' reliable txs exploit locality via **dynamic ownership**:

local accesses in the common case

single-node commit

- **local** for read-only txs
- **fast and pipelined** for write txs

zeus-protocol.com

TLA+ specification, Q&A …

Conclusion

State-of-the-art reliable txs over **static sharding**:

cannot exploit dynamic locality

remote accesses

costly **distributed commit**

Zeus' reliable txs exploit locality via **dynamic ownership**:

local accesses in the common case

single-node commit

- **local** for read-only txs
- **fast and pipelined** for write txs

Performance 10s millions txs/second

up to **2x state-of-the-art**

zeus-protocol.com

TLA+ specification, Q&A …

THE UNIVERSITY of EDINBURGH
informatics

Conclusion

State-of-the-art reliable txs over **static sharding**:

cannot exploit dynamic locality

remote accesses

costly **distributed commit**

Zeus' reliable txs exploit locality via **dynamic ownership**:

local accesses in the common case

single-node commit

- **local** for read-only txs
- **fast and pipelined** for write txs

Performance 10s millions txs/second

up to **2x state-of-the-art**

Bonus: programmability!


Zeus: Locality-aware Distributed Transactions

Artifacts Available acm

zeus-protocol.com

TLA+ specification, Q&A …

THE UNIVERSITY of EDINBURGH
informatics

Conclusion

State-of-the-art reliable txs over **static sharding**:

cannot exploit dynamic locality

remote accesses

costly **distributed commit**

Zeus' reliable txs exploit locality via **dynamic ownership**:

local accesses in the common case

single-node commit

- **local** for read-only txs
- **fast and pipelined** for write txs

Performance 10s millions txs/second

up to **2x state-of-the-art**

Bonus: programmability!

zeus-protocol.com

TLA+ specification, Q&A …

Reliable txs with locality? Use Zeus!