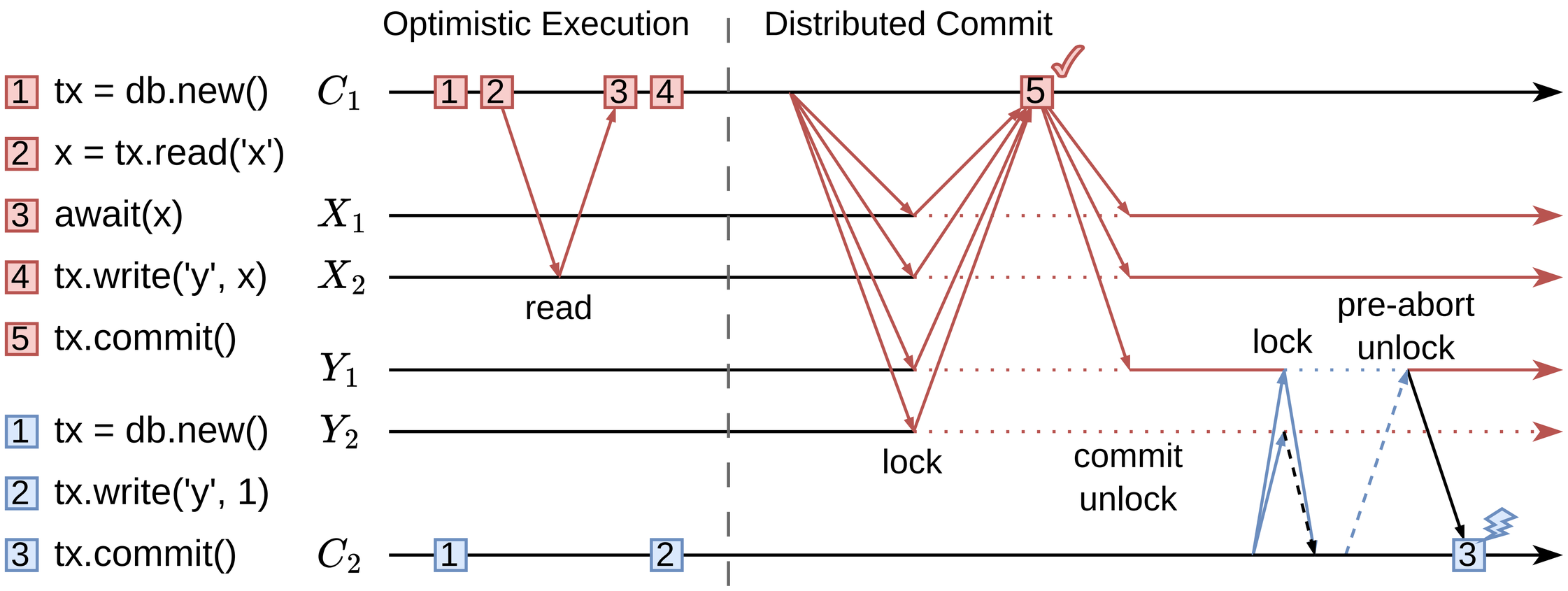
Unanimous 2PC: simple, fast and fault tolerant distributed transactions

Chris Jensen [CH], *Heidi Howard* [CA], *Antonios Katsarakis* [H], *Richard Mortier* [C]
[C]University of Cambridge, [H]Huawei Research, [A]Azure Research



Motivation

2-Phase-Commit (2PC) is *fast* but *not fault-tolerant*, while distributed commit protocols are *fault-tolerant* but *slow*.

Can we make 2PC both fast and fault tolerant? **Yes**

Protocol

Optimistic Execution Phase

U2PC transactions are first executed *optimistically*. Reads are performed at any replica in the shard, and writes are buffered until after commit.

Distributed Commit Phase

The commit protocol must validate that the optimistic execution was valid. The coordinator broadcasts `Lock` to **all** replicas. Then acts according to the following:

Message	Threshold	Action
`Lock-Ack`	**all** replicas in **all** shards	**commit**
`Lock-Nack`	**any** replica in **any** shard	`Pre-Abort-Unlock`
`Lock-Nack`	**all** replicas in **any** shard	**abort**

Recovery

To recover stop at least one replica per shard and read its lock state. Then for each transaction:

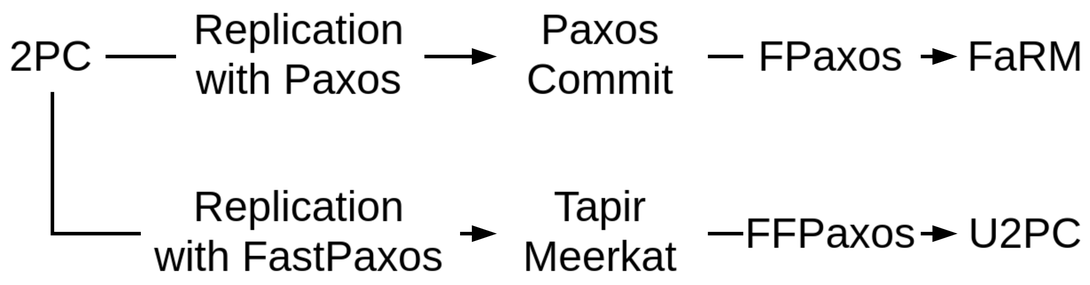
any `Commit`	**commit**
any `Abort`	**abort**
all `Lock`	**commit**
any not `Lock`	**abort**

Performance

	Cluster Size	Messages Read	Messages Write	RTT Commit	RTT Abort
FaRM	f+1	5+3f	2	3,4	1,2
FastPaxos	2f+1	3+6f	3+6f	1,2	1,2
U2PC	f+1	3+3f	2,3+3f[1]	1	1,2

U2PC scales better than FastPaxos based approaches such as TAPIR or Meerkat with lower latency than FaRM. However for multi-shard read-write transactions U2PC has higher overhead than FaRM.

Theory



Similarly to how FaRM applies *FlexiblePaxos* to PaxosCommit to reduce its shard overhead and improve scalability, U2PC applies *FastFlexiblePaxos* to Meerkat.

[1]Read-only transactions have a fast path of 2 messages.