Spanner: Google's Globally-Distributed Database

Google, Inc.

OSDI 2012

Presented by: Karen Ouyang

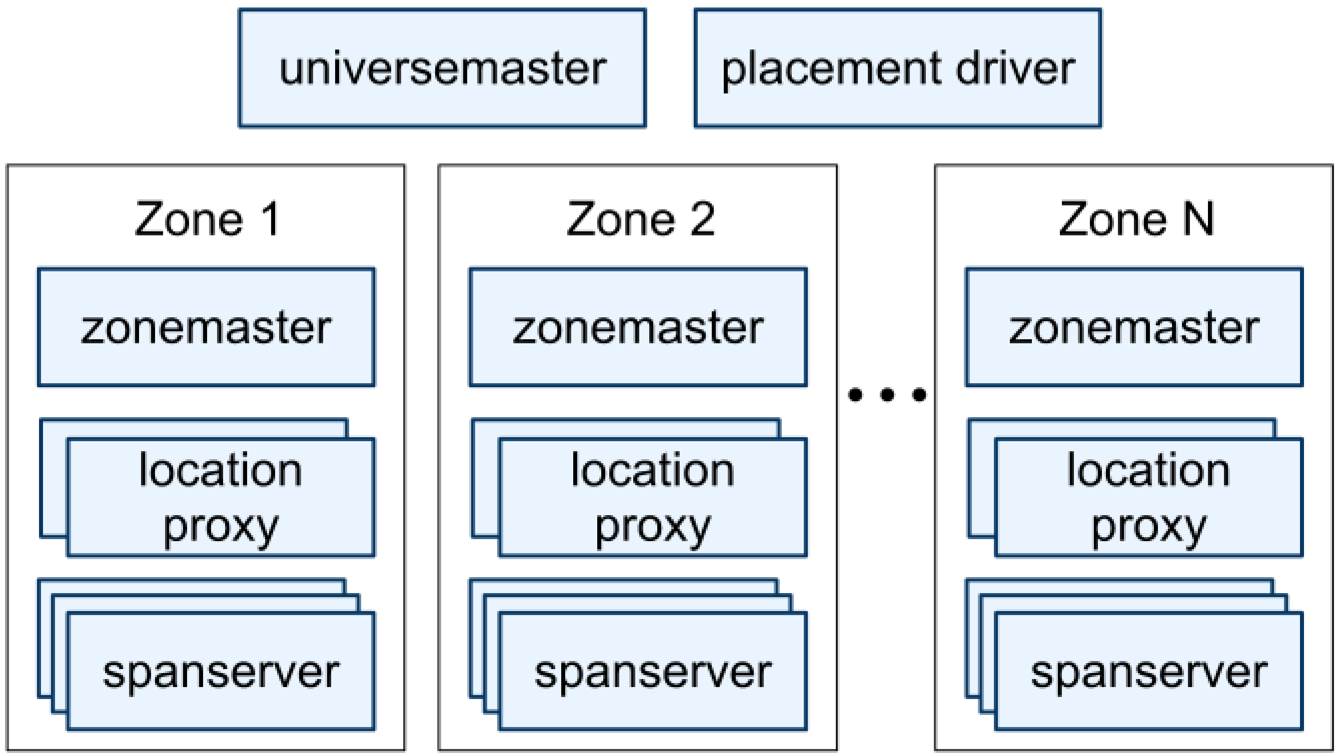
Problem Statement

- Distributed data system with high availability
- Support external consistency!

Key Ideas

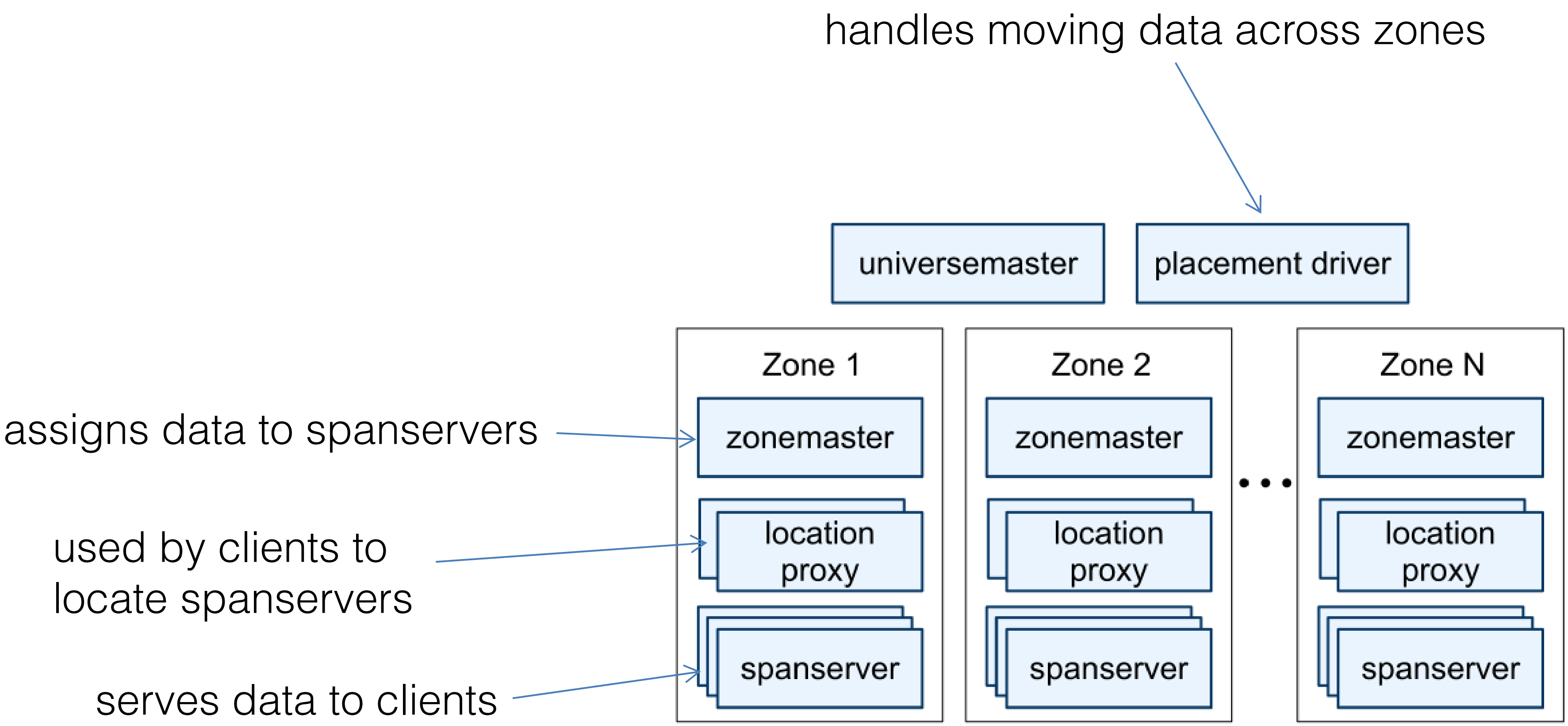
- Distributed data system with high availability
- Supports external consistency!
- Enabling technology: TrueTime API

Server Organization



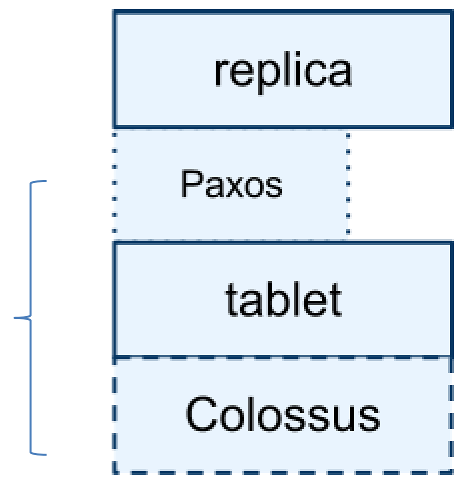
datacenters have one or more zones

Server Organization



Spanserver Stack

between 100 and 1000 instances

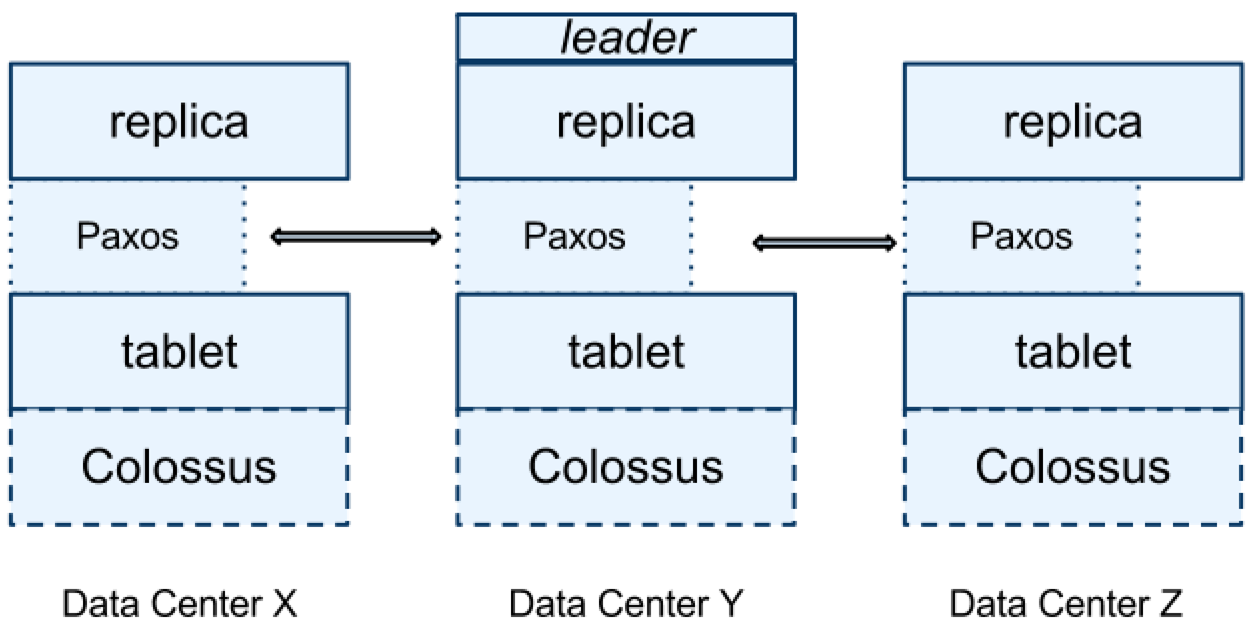


(key:string, timestamp:int64) → string

Spanserver Stack

set of replicas: *Paxos group*

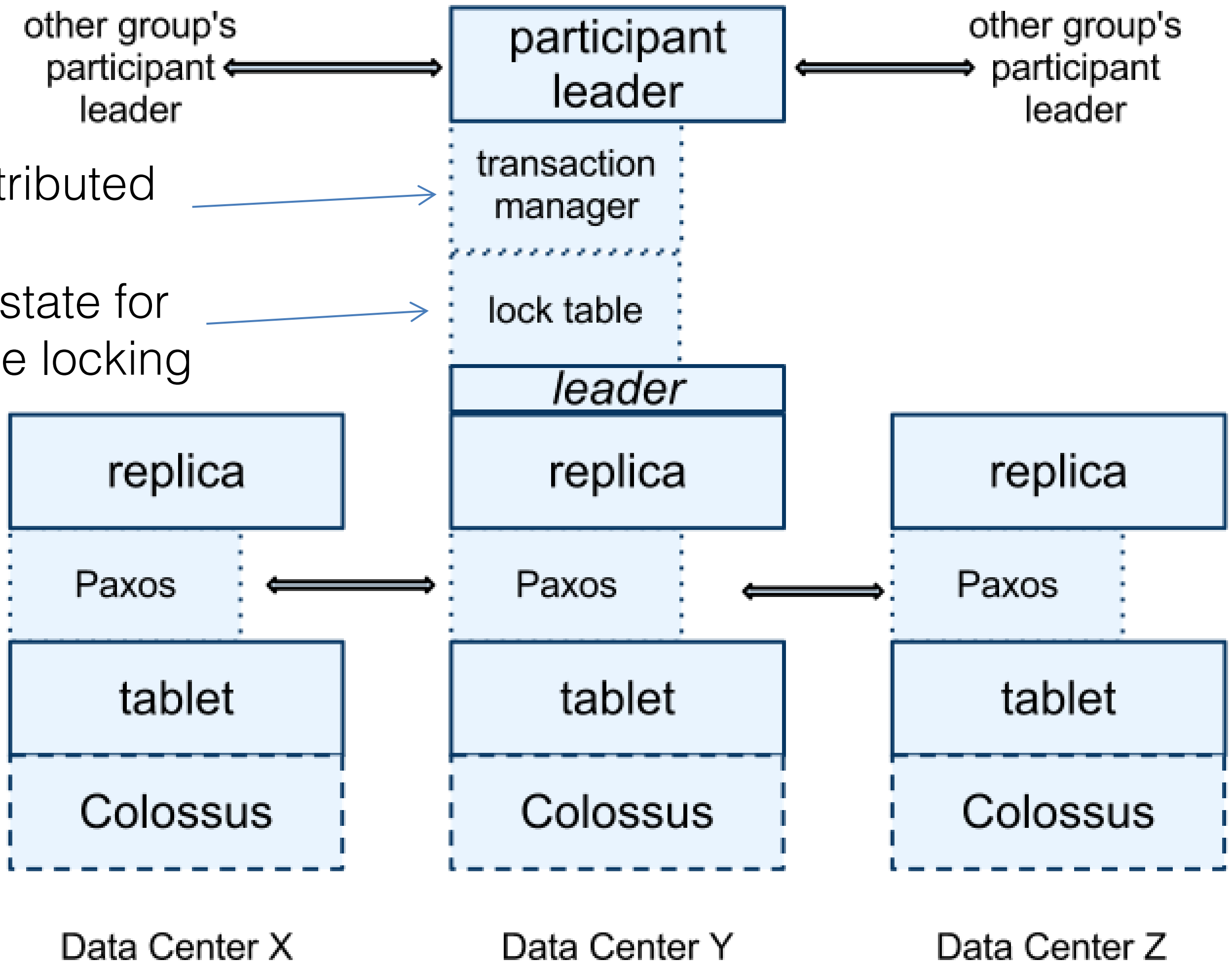
writes initiate Paxos protocol at leader;
reads from any sufficiently up-to-date replica



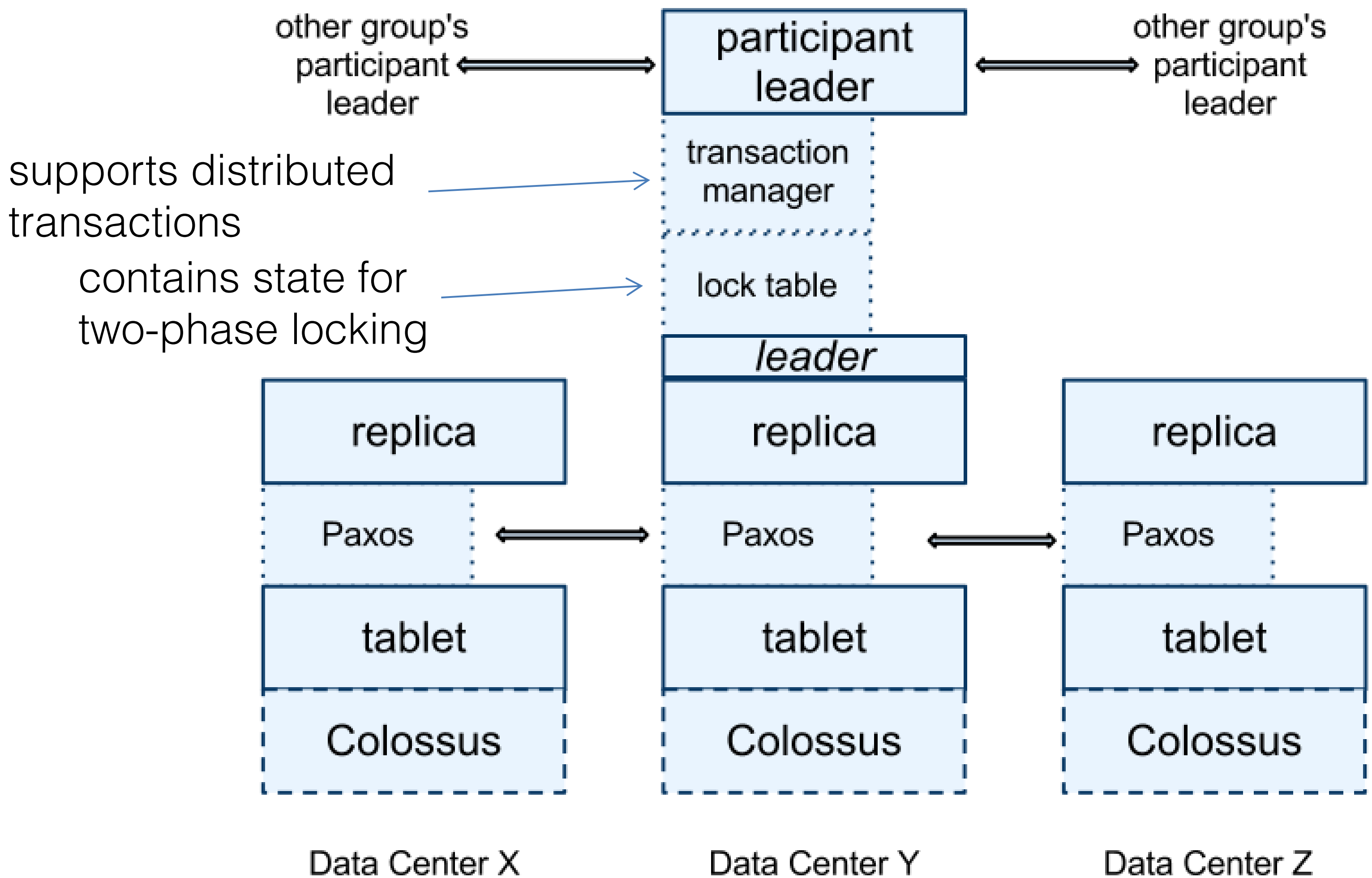
Spanserver Stack

supports distributed transactions

contains state for two-phase locking



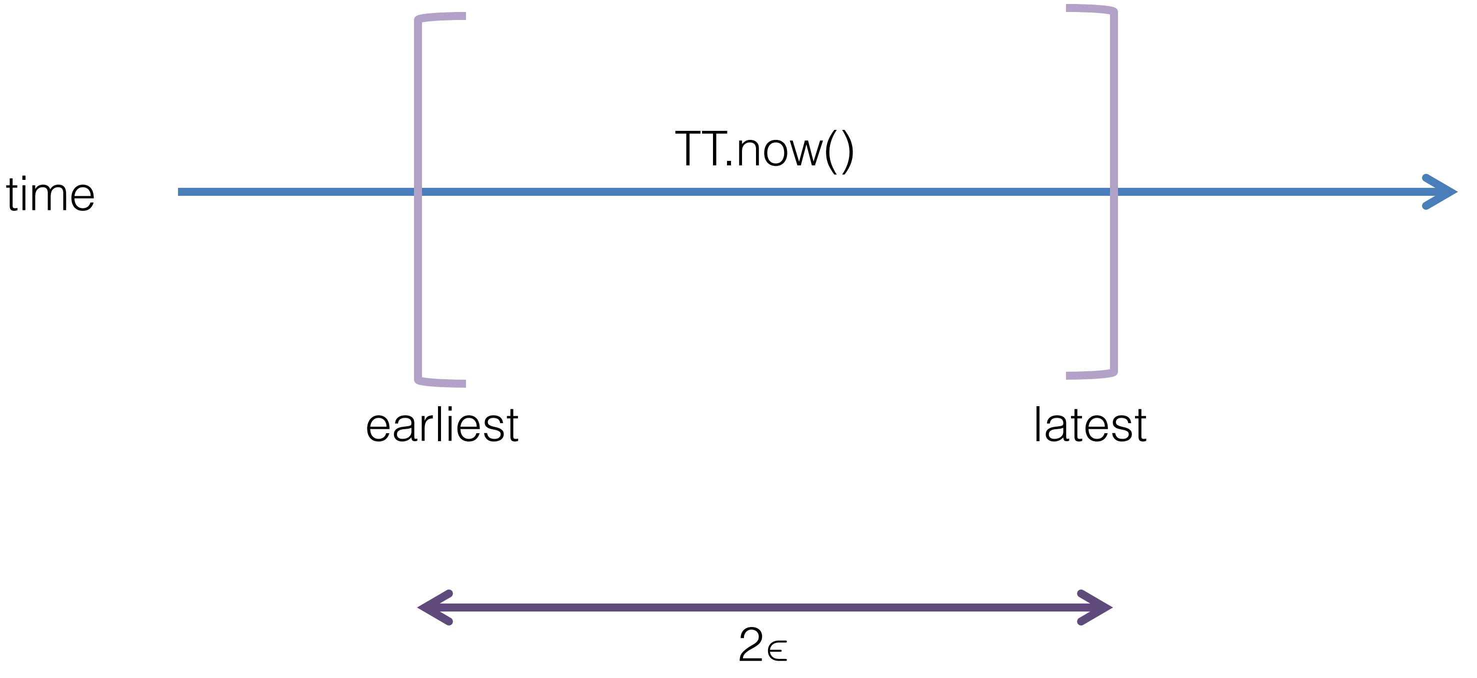
transactions with 1+ group: two-phase commit
select *coordinator leader* from participant leaders



TrueTime API

- Exposes clock uncertainty by expressing time as an interval
- Uses GPS and atomic clocks
- *Time master* machines per datacenter
- Client polls multiple masters to compute time interval

TrueTime API



Consistency

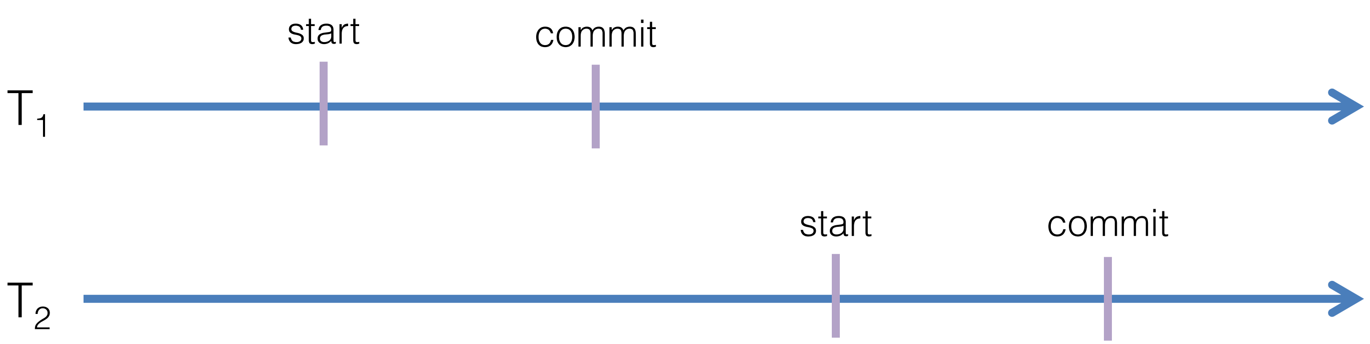
- Ensure external consistency by ensuring timestamp order
- All transactions are assigned timestamp
- Data written by T is timestamped with s

Read-Write Transactions

- Two-phase locking: assign timestamps at any time that locks are held
- Assign timestamps to Paxos writes in increasing order across leaders
  - A leader only assigns timestamps within its leader lease; leader leases are disjoint

Read-Write Transactions

- Transactions: two-phase commit
- Two transactions



- Assign commit timestamps with $s_1 < s_2$
- How?

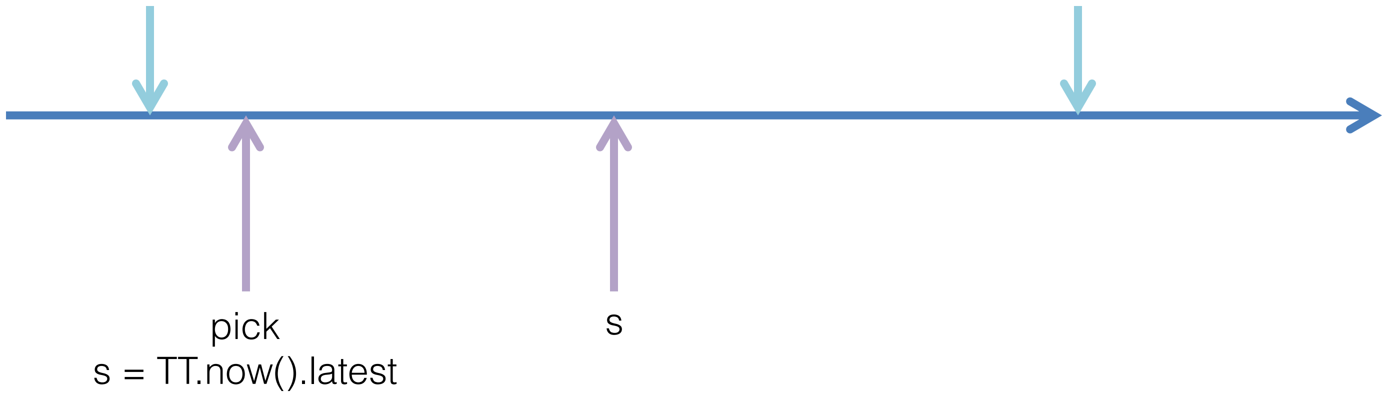
Read-Write Transactions

Start: commit timestamp is *after* time of commit request at server

- or: $t_{abs}(e_2^{server}) \leq s$

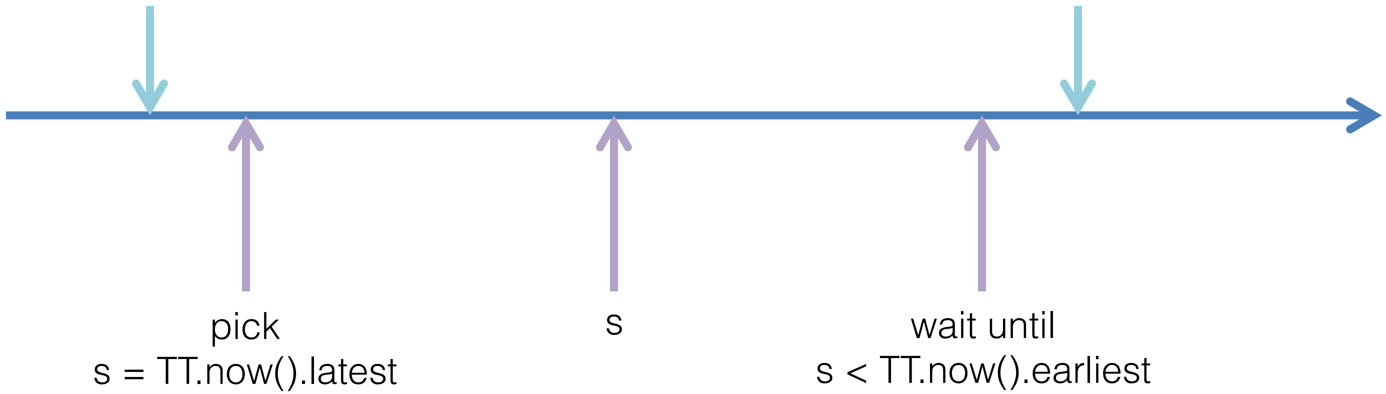
Read-Write Transactions

Commit wait: cannot see data committed by T until s (assigned timestamp) has passed



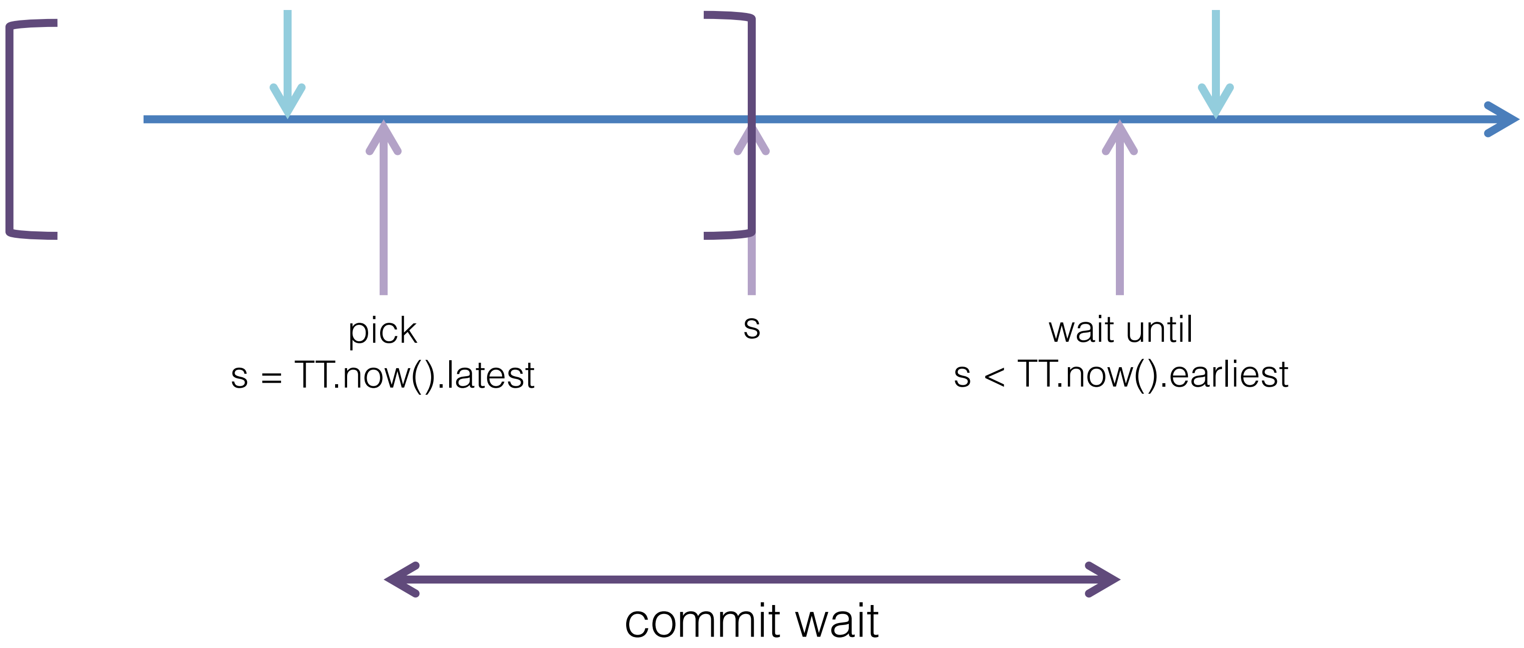
Read-Write Transactions

Commit wait: cannot see data committed by T until s (assigned timestamp) has passed



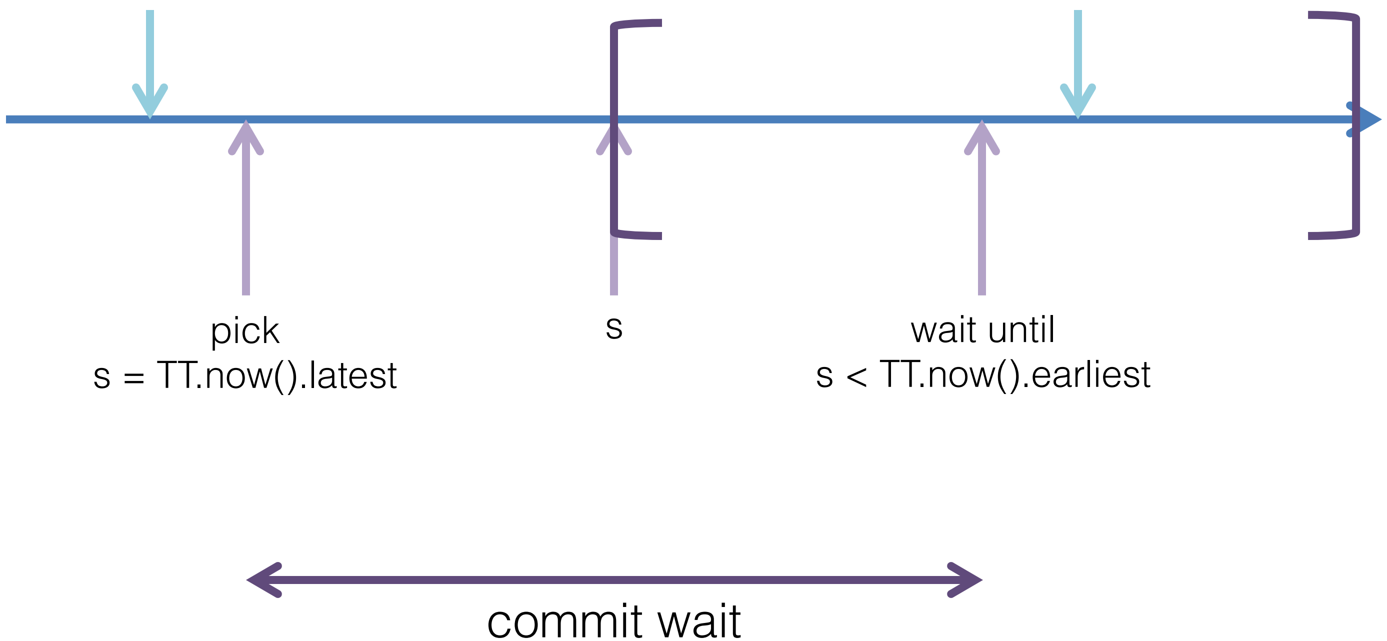
Read-Write Transactions

Commit wait: cannot see data committed by T until s (assigned timestamp) has passed



Read-Write Transactions

Commit wait: cannot see data committed by T until s (assigned timestamp) has passed



Read-Write Transactions

$s_1 < t_{abs}(e_1^{commit})$

$t_{abs}(e_1^{commit}) < t_{abs}(e_2^{start})$

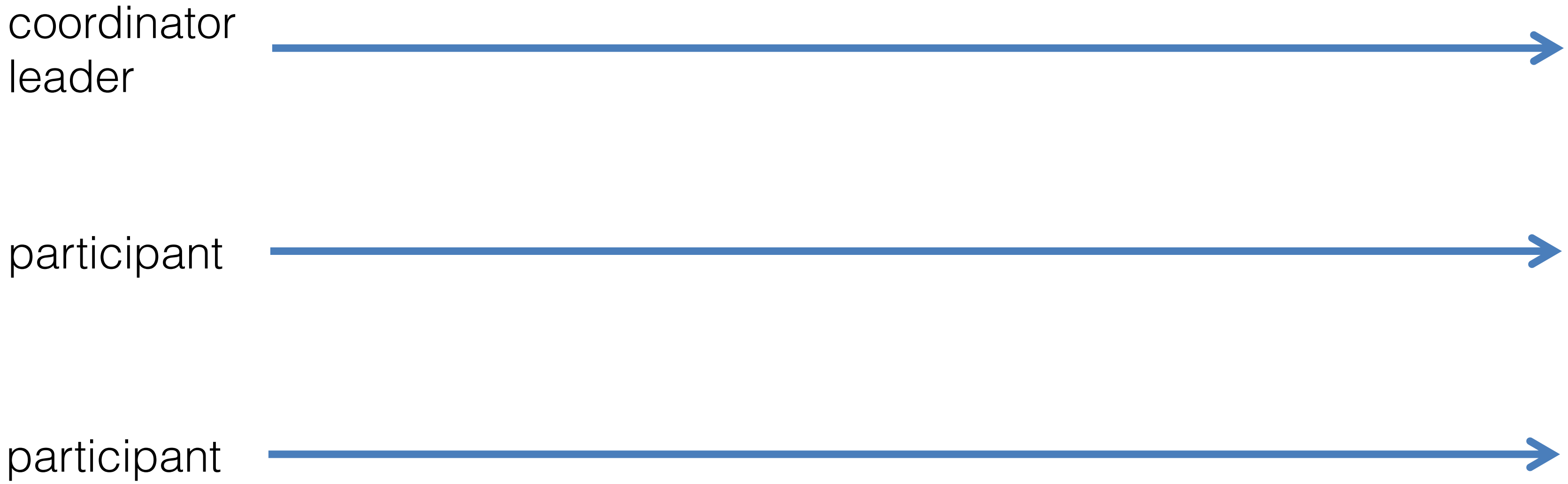
$t_{abs}(e_2^{start}) < t_{abs}(e_2^{server})$

$t_{abs}(e_2^{server}) \leq s_2$

$s_1 < s_2$

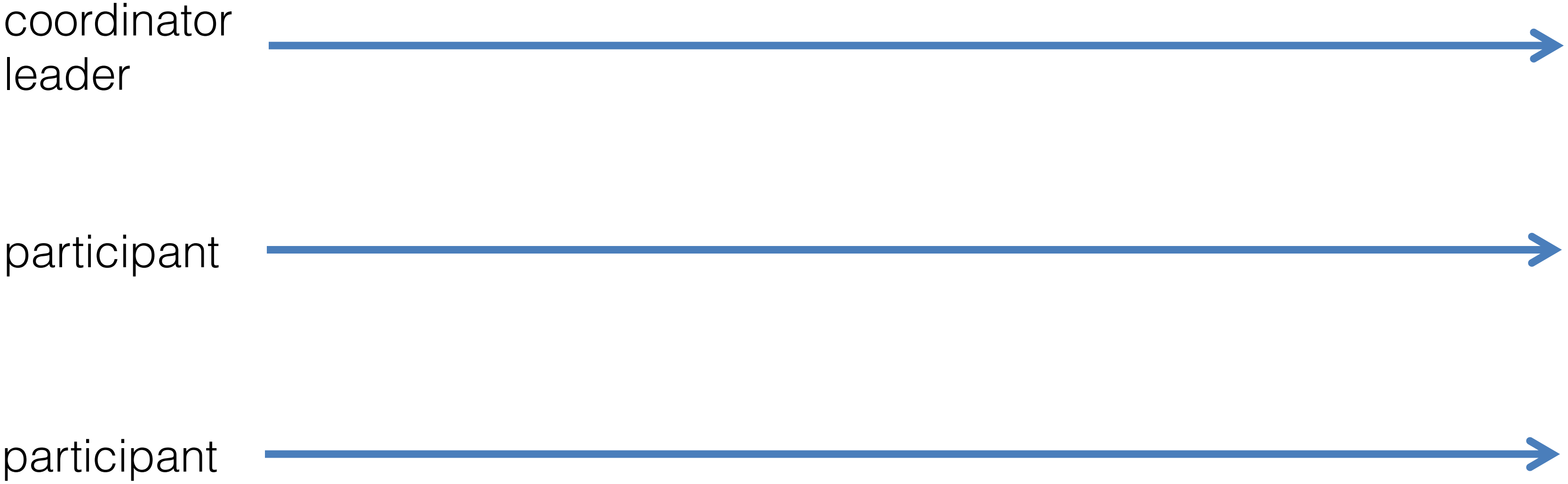
Read-Write Transactions

Two-phase commit



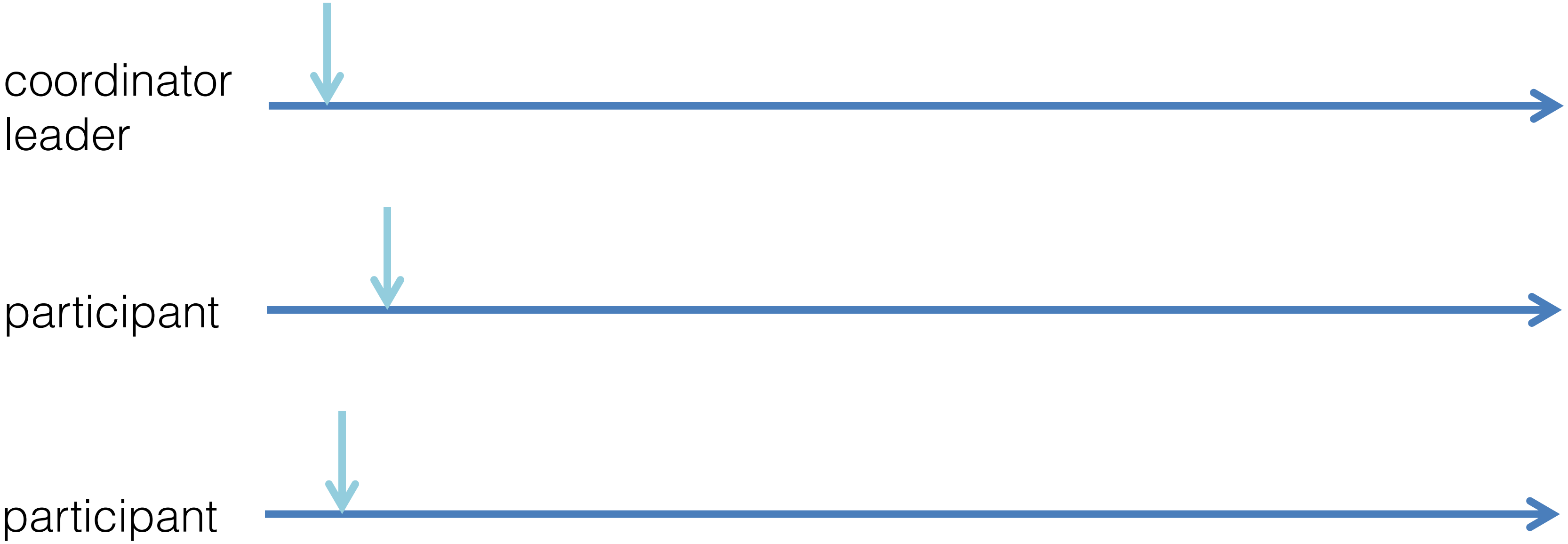
Read-Write Transactions

Two-phase commit: client begins



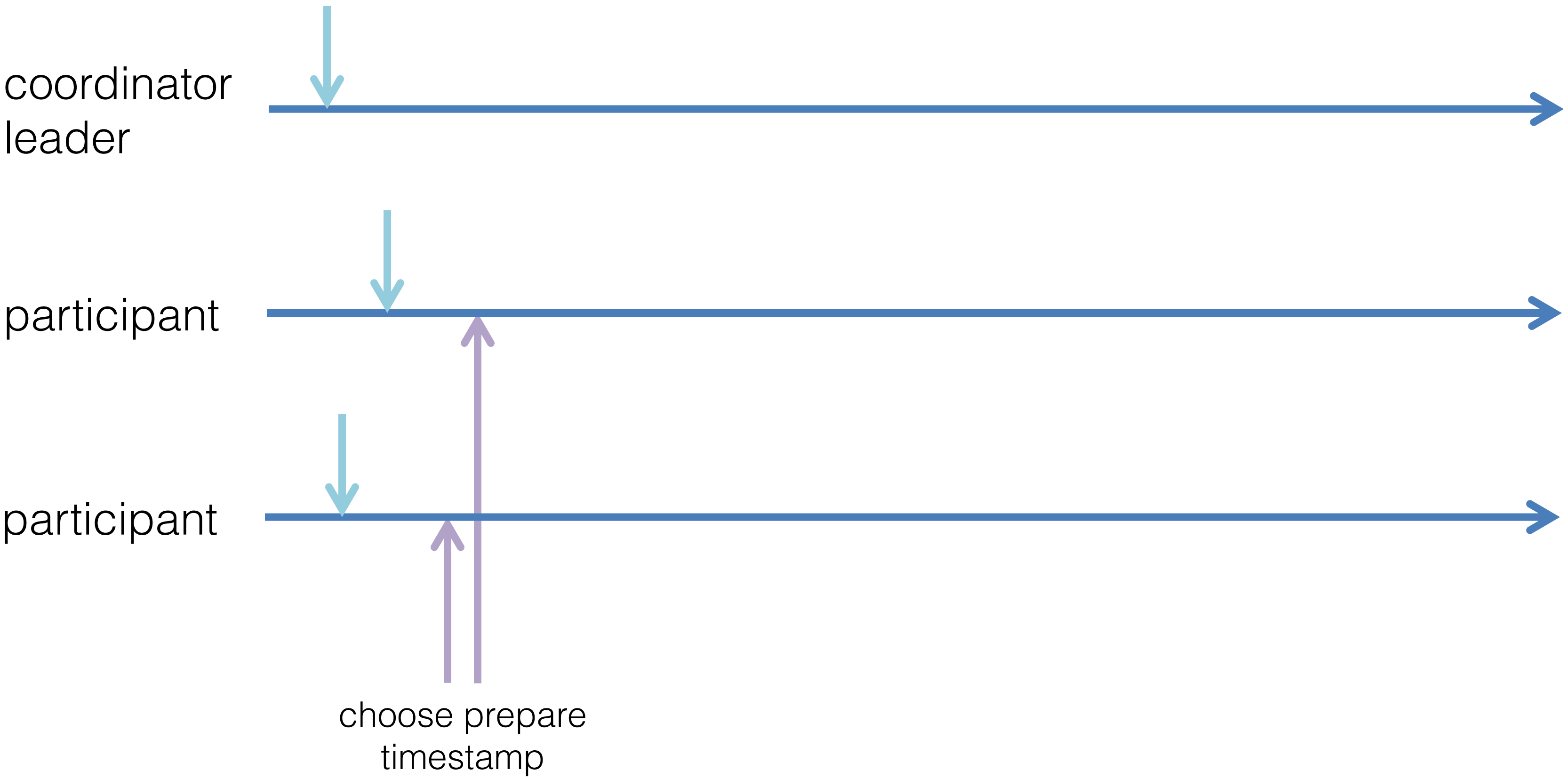
Read-Write Transactions

Two-phase commit



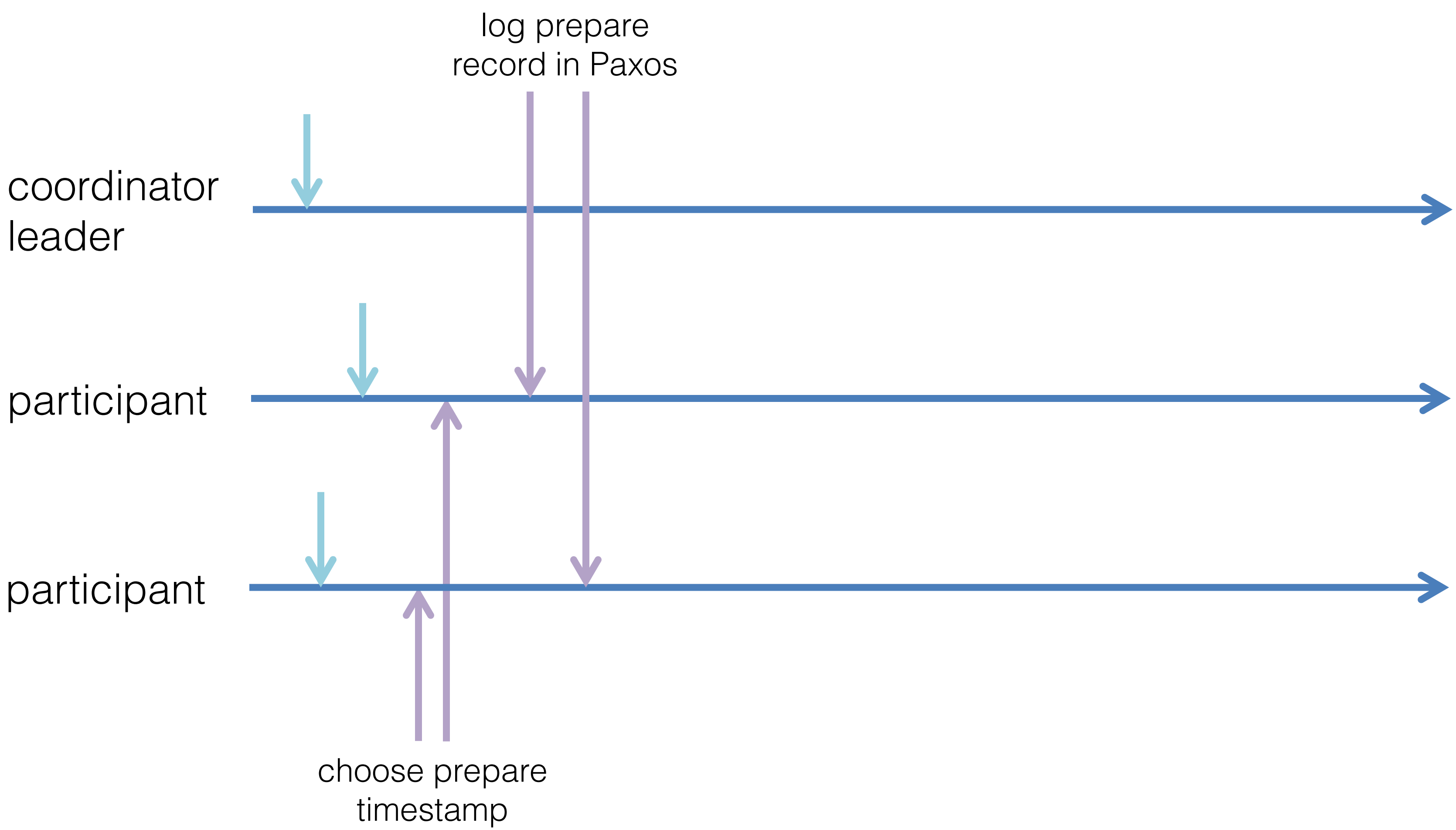
Read-Write Transactions

Two-phase commit



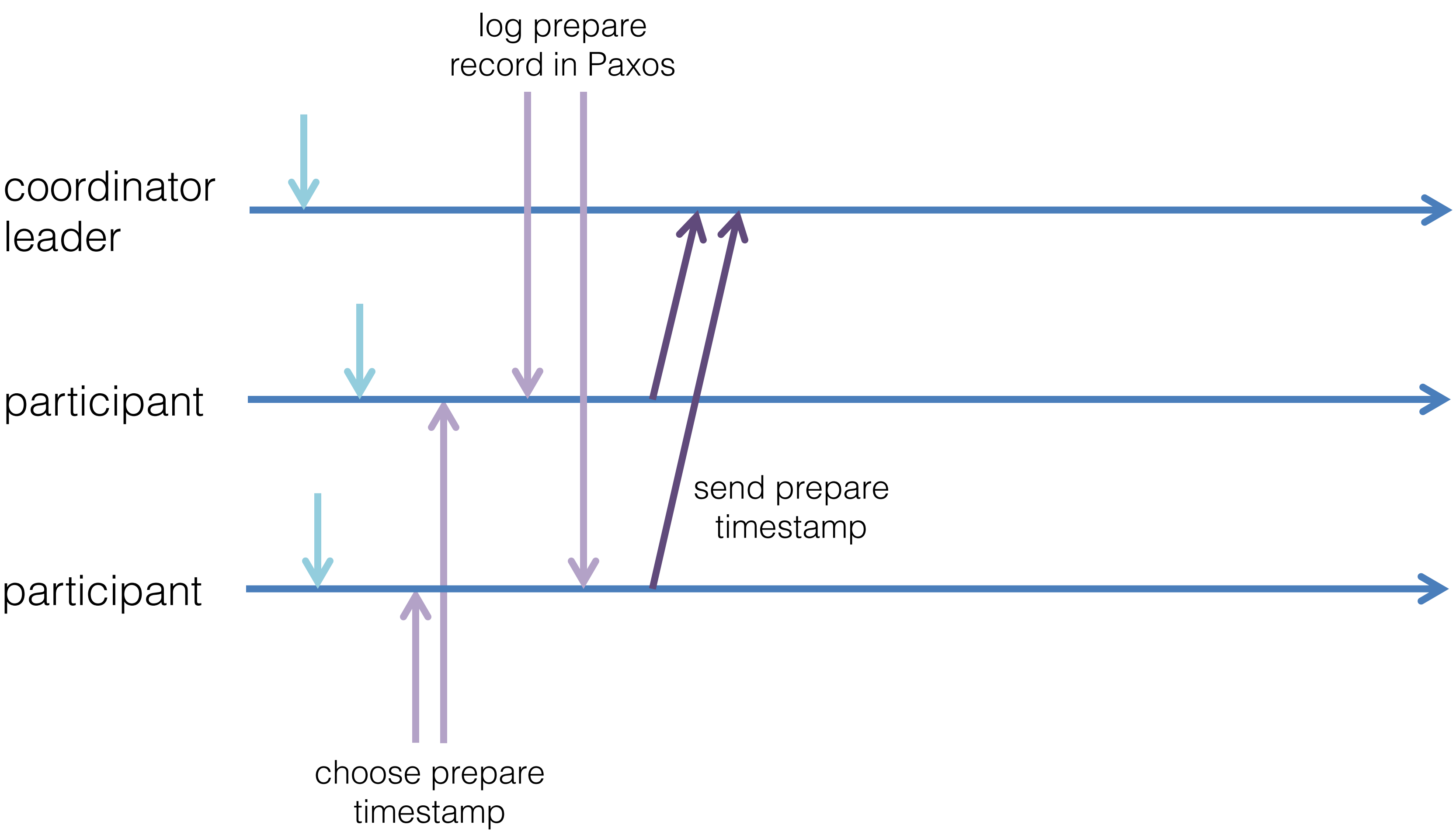
Read-Write Transactions

Two-phase commit



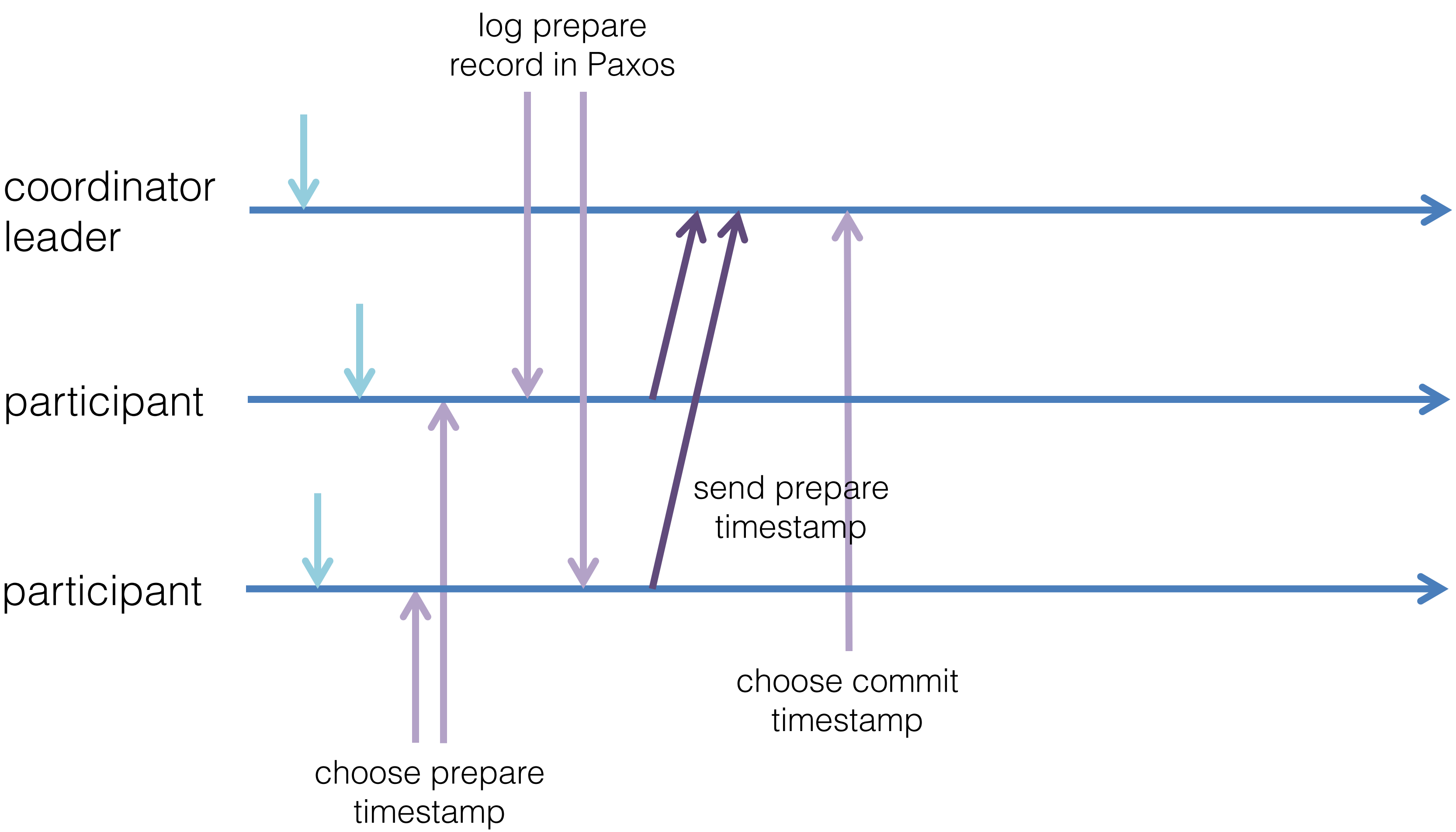
Read-Write Transactions

Two-phase commit



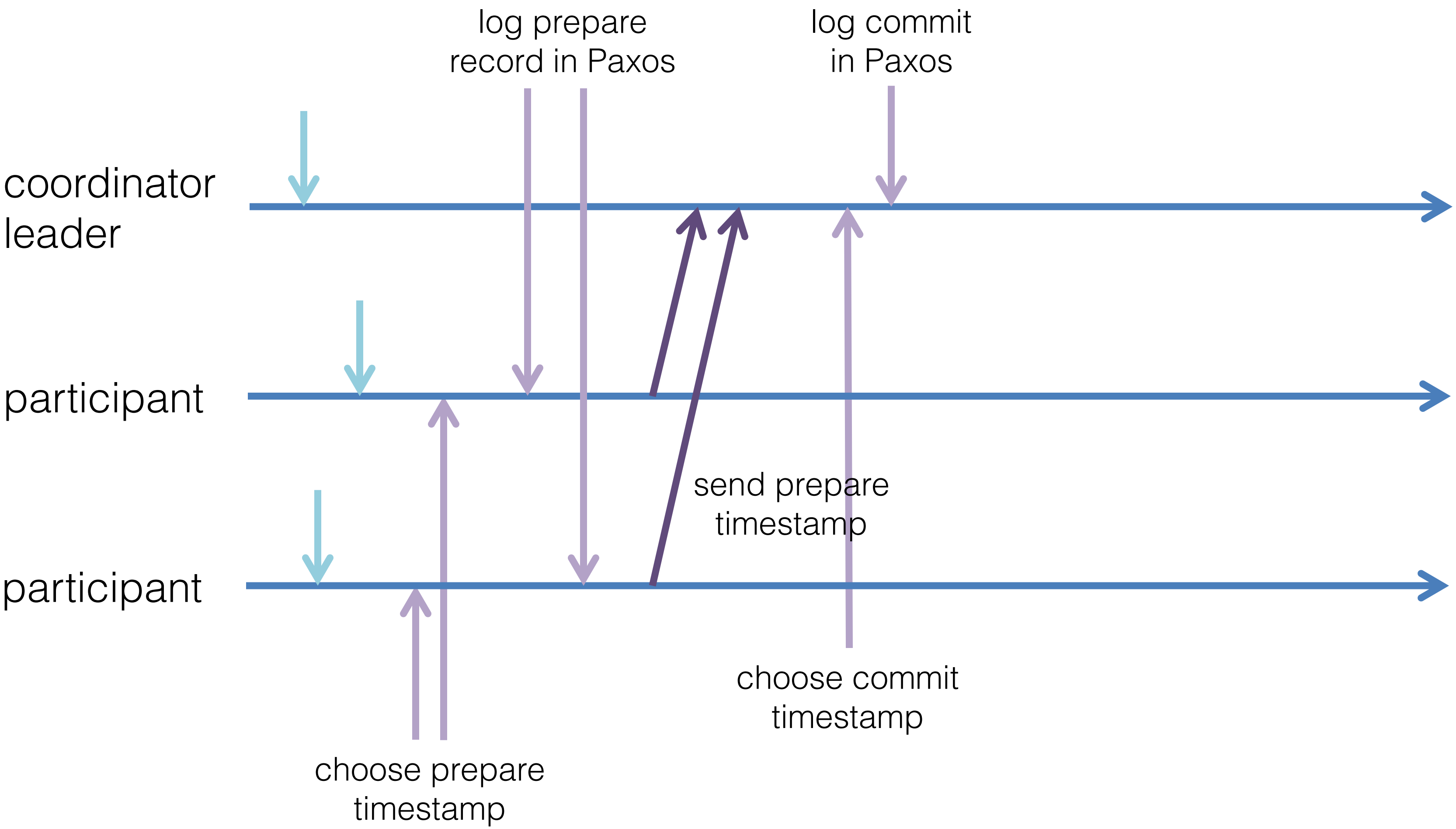
Read-Write Transactions

Two-phase commit



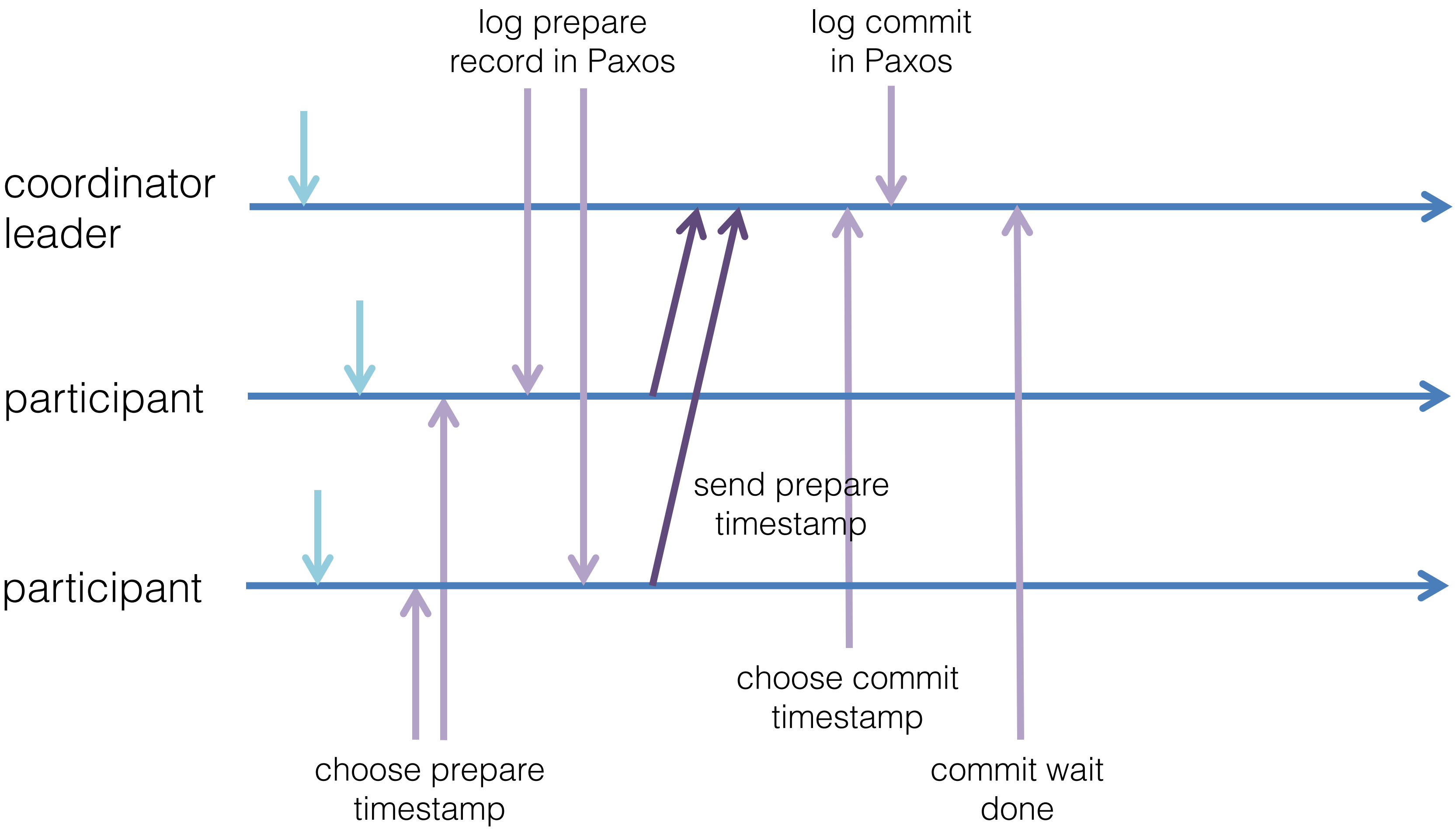
Read-Write Transactions

Two-phase commit



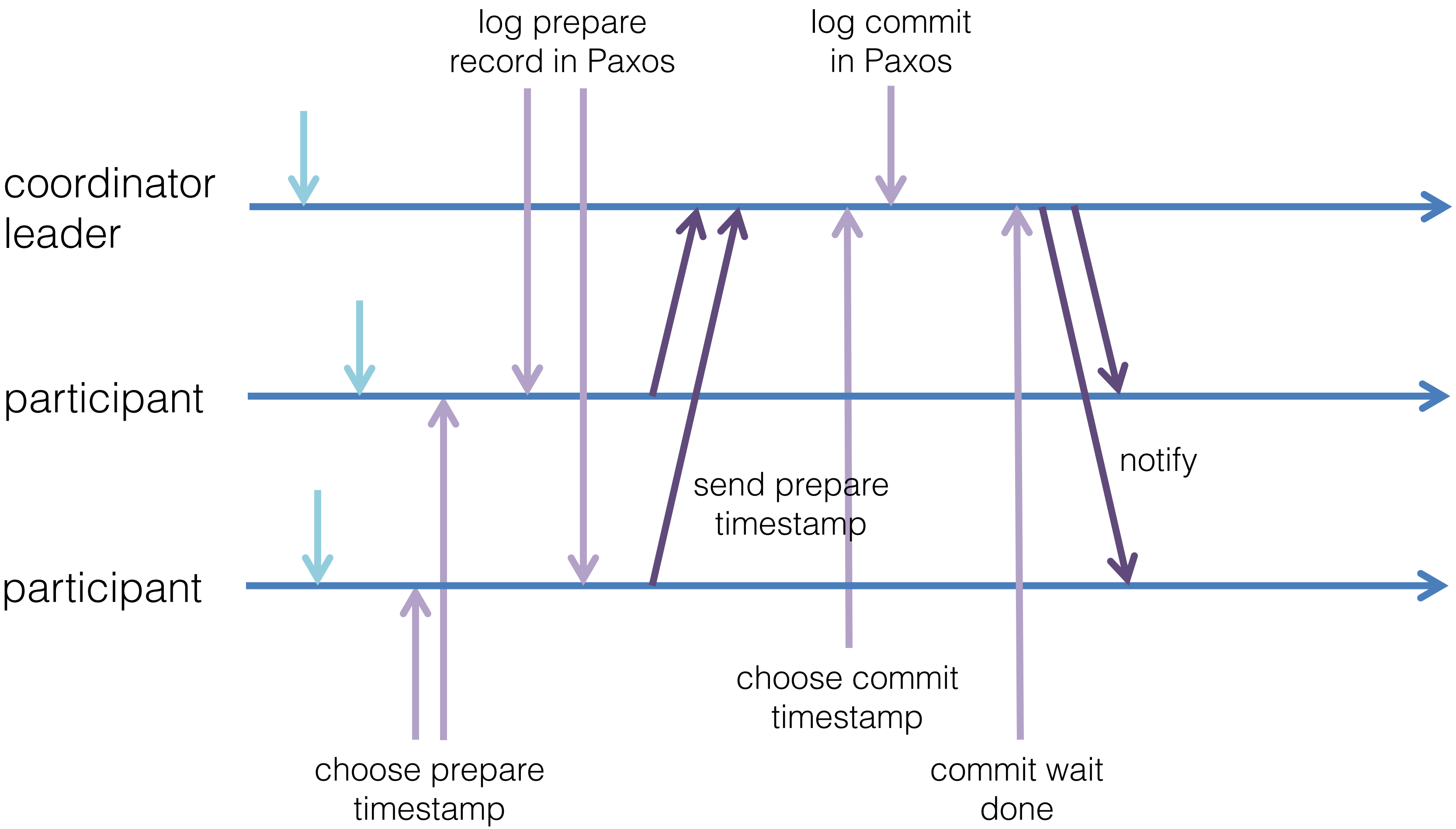
Read-Write Transactions

Two-phase commit



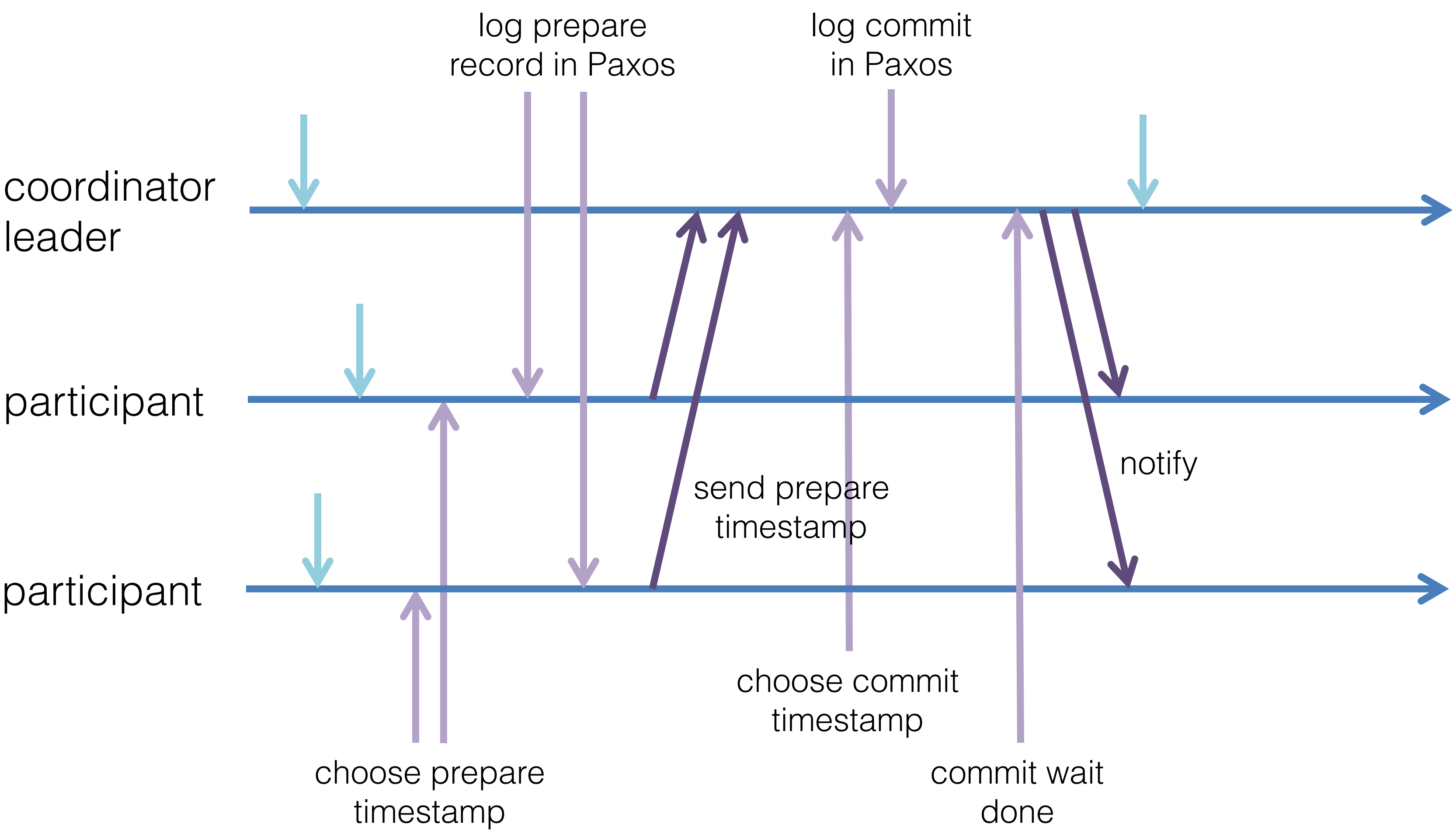
Read-Write Transactions

Two-phase commit



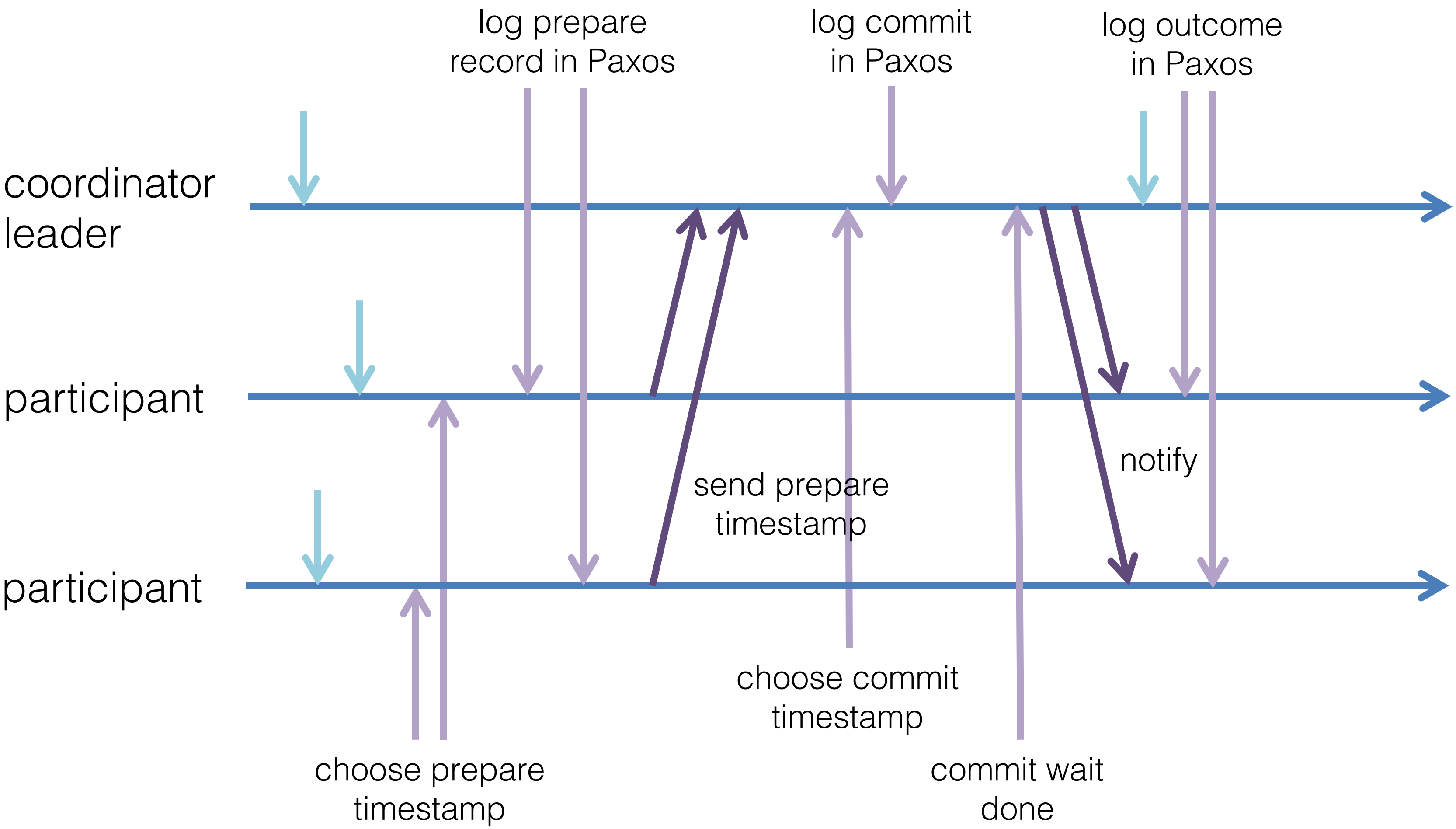
Read-Write Transactions

Two-phase commit



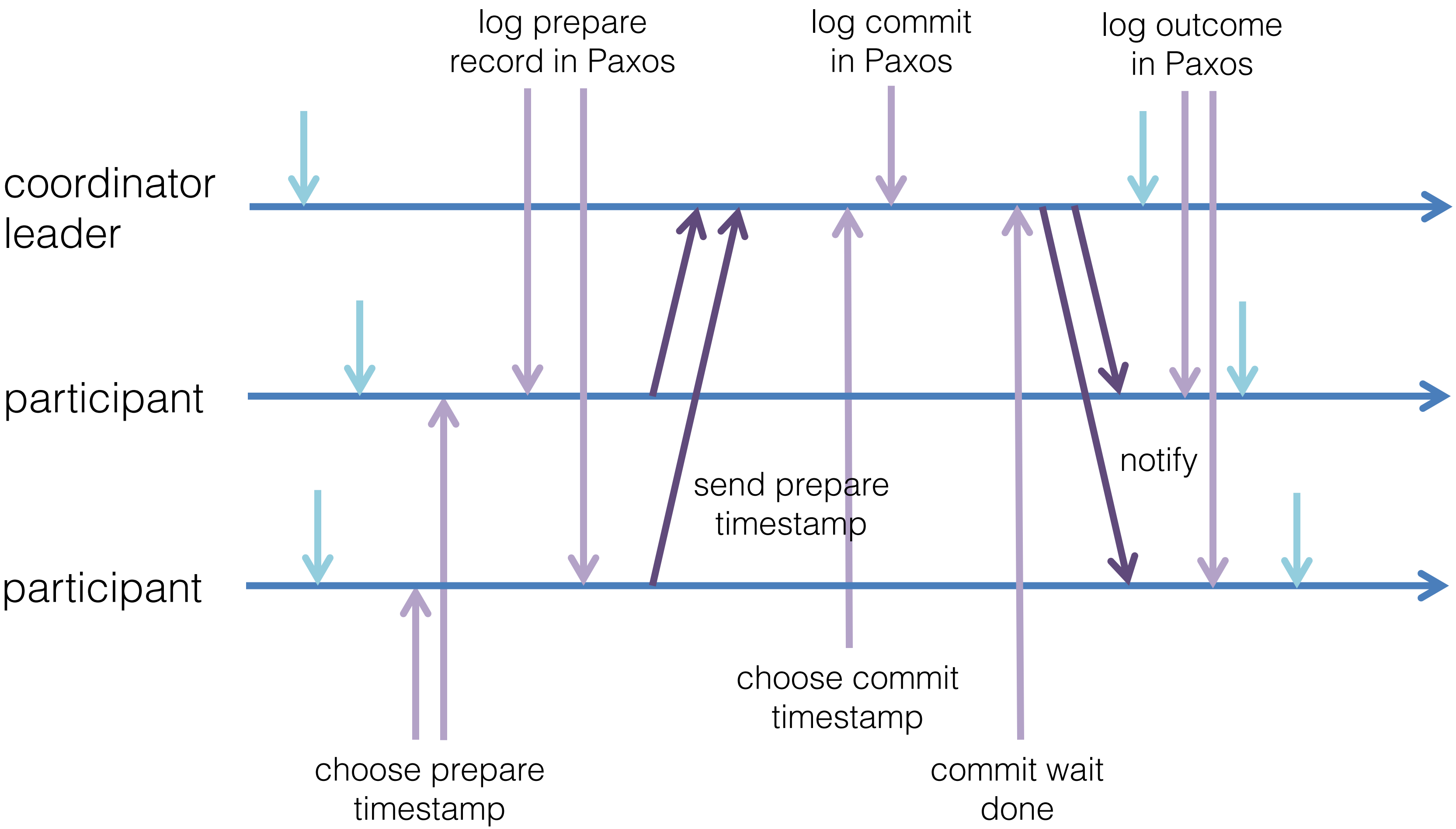
Read-Write Transactions

Two-phase commit



Read-Write Transactions

Two-phase commit



Read-Only Transactions

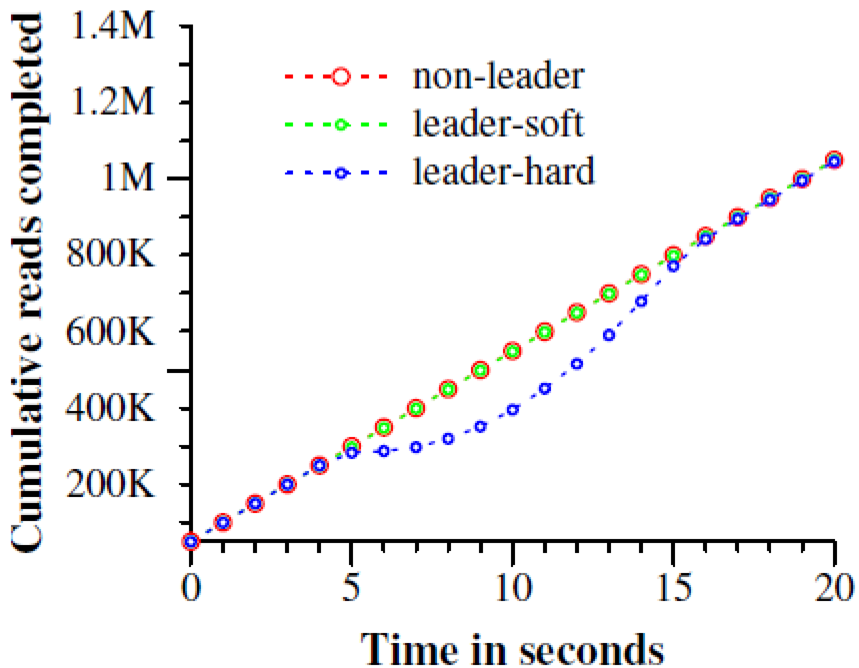
- Serving reads at a timestamp
  - Replica tracks safe time t_{safe}: can read $t \leq t_{safe}$
  - Define $t_{safe} = \min(t^{Paxos}, t^{TM})$
- Assigning timestamps to RO transactions
  - Simplest: assign s_{read} = TT.now().latest
  - May block; should assign oldest timestamp that preserves external consistency

Microbenchmarks

participants	latency (ms)	
	mean	99th percentile
1	17.0 ±1.4	75.0 ±34.9
2	24.5 ±2.5	87.6 ±35.9
5	31.5 ±6.2	104.5 ±52.2
10	30.0 ±3.7	95.6 ±25.4
25	35.5 ±5.6	100.4 ±42.7
50	42.7 ±4.1	93.7 ±22.9
100	71.4 ±7.6	131.2 ±17.6
200	150.5 ±11.0	320.3 ±35.1

Two-phase commit scalability

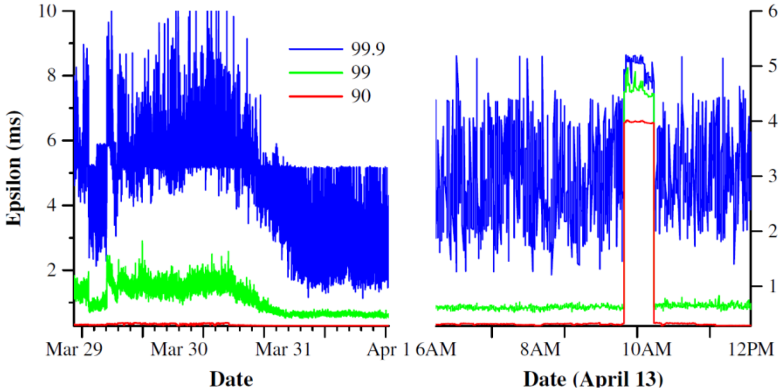
Microbenchmarks



Effect of killing servers on throughput

Performance

- TrueTime



- F1, Google's advertising backend
  - Automatic failover ☺
  - High standard deviation for latency?

Final Thoughts

- Implemented at a large scale (F1)!
- Commit wait is pretty clever
- Very dependent on clocks
- Security?

References

- Corbett et al. “Spanner: Google’s Globally-Distributed Database.” *Proc. Of OSDI.* 2012.
- http://research.google.com/archive/spanner.html