Imperial College London

LSDS
Large-Scale Data & Systems Group

Spanner: Google's Globally Distributed Database

Peter Pietzuch
prp@doc.ic.ac.uk

Department of Computing
Imperial College London
http://lsds.doc.ic.ac.uk

Based on slides by Google

Autumn 2020

What is Spanner?

Next step from Bigtable in RDBMS evolution with strong time semantics: Distributed multi-version database

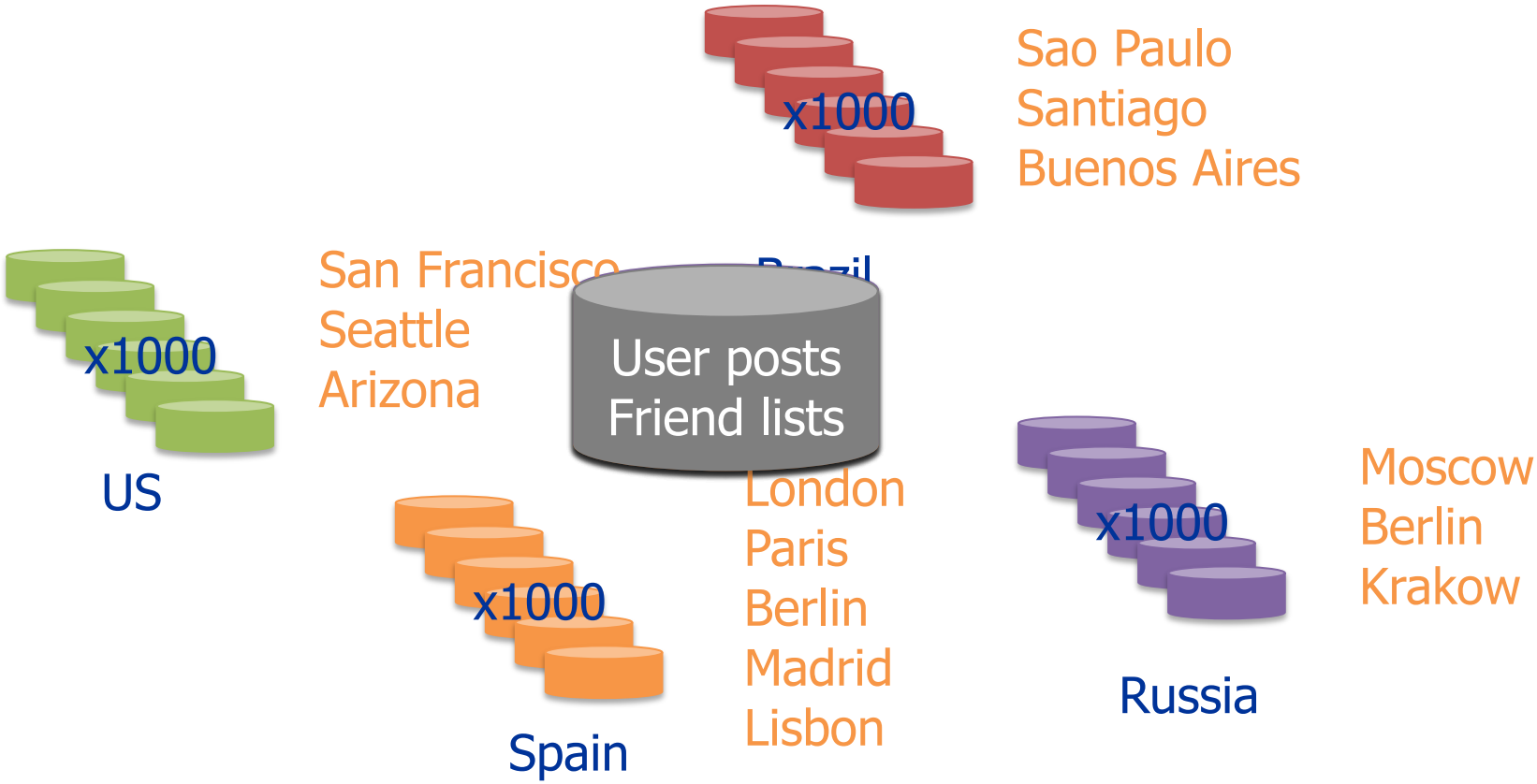
Key features of Spanner:

- General-purpose transactions (ACID)
  - Externally consistent global write-transactions with synchronous replication
  - Transactions across Datacenters
  - Lock-free read-only transactions
- Schematised, semi-relational (tabular) data model
  - SQL-like query interface

Running in production

- Storage for Google's advertisement data
- Replaced (manually) sharded MySQL database

Example: Social Network



Spanner Overview

Property: **External consistency** of distributed transactions

- First system at global scale
- Enables transaction serialization via global timestamps

Implementation: Integration of concurrency control, replication, and 2PC (Two phase commit)

- Correctness and performance
- Auto-sharding, auto-rebalancing, automatic failure response

Enabling technology: TrueTime

- Interval-based global time
- Acknowledges clock uncertainty and guarantees a bound on it
- Uses GPS devices and Atomic clocks to get accurate time

Read Transactions

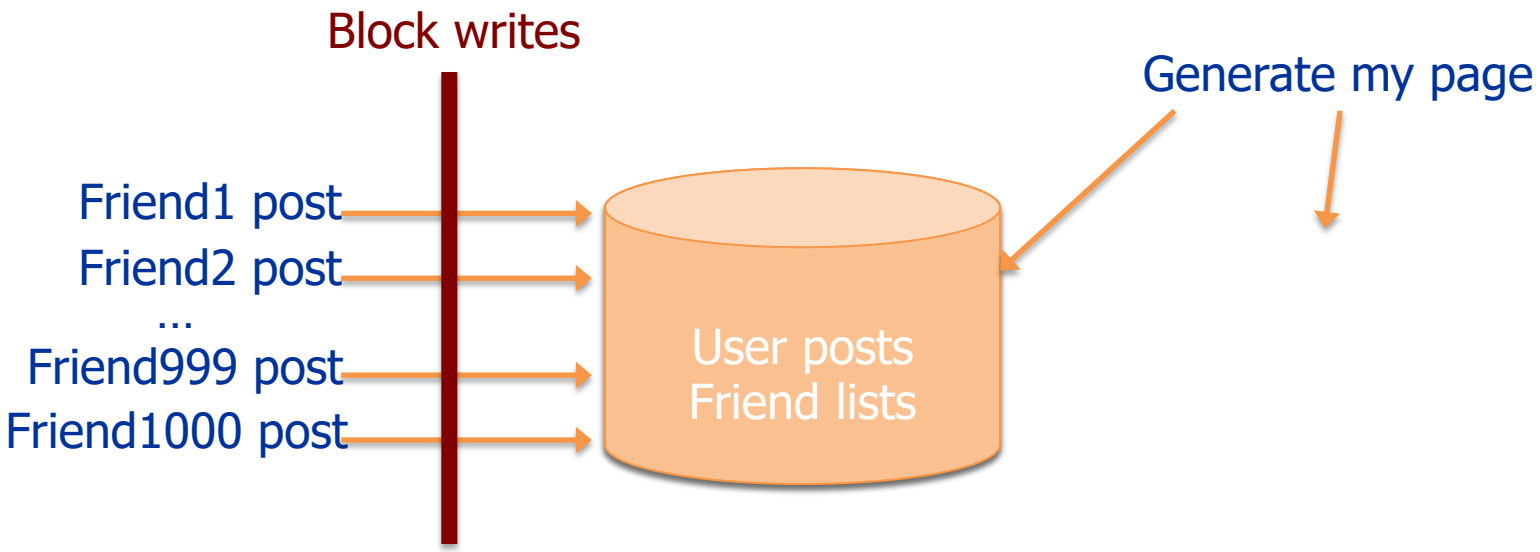
Generate page of friends' recent posts

- Consistent view of friend list and their posts

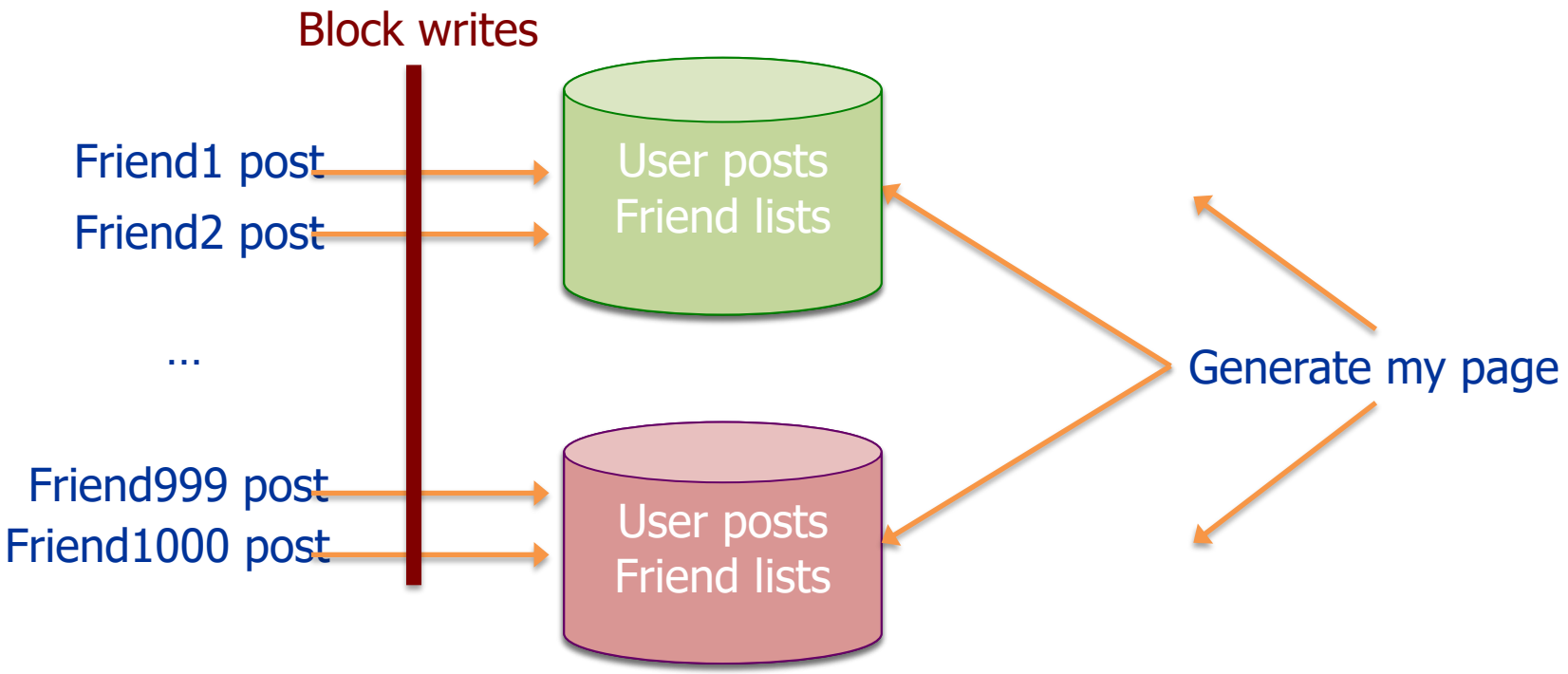
Why consistency matters

1. Remove untrustworthy person X as friend
2. Post P: "My government is repressive…"

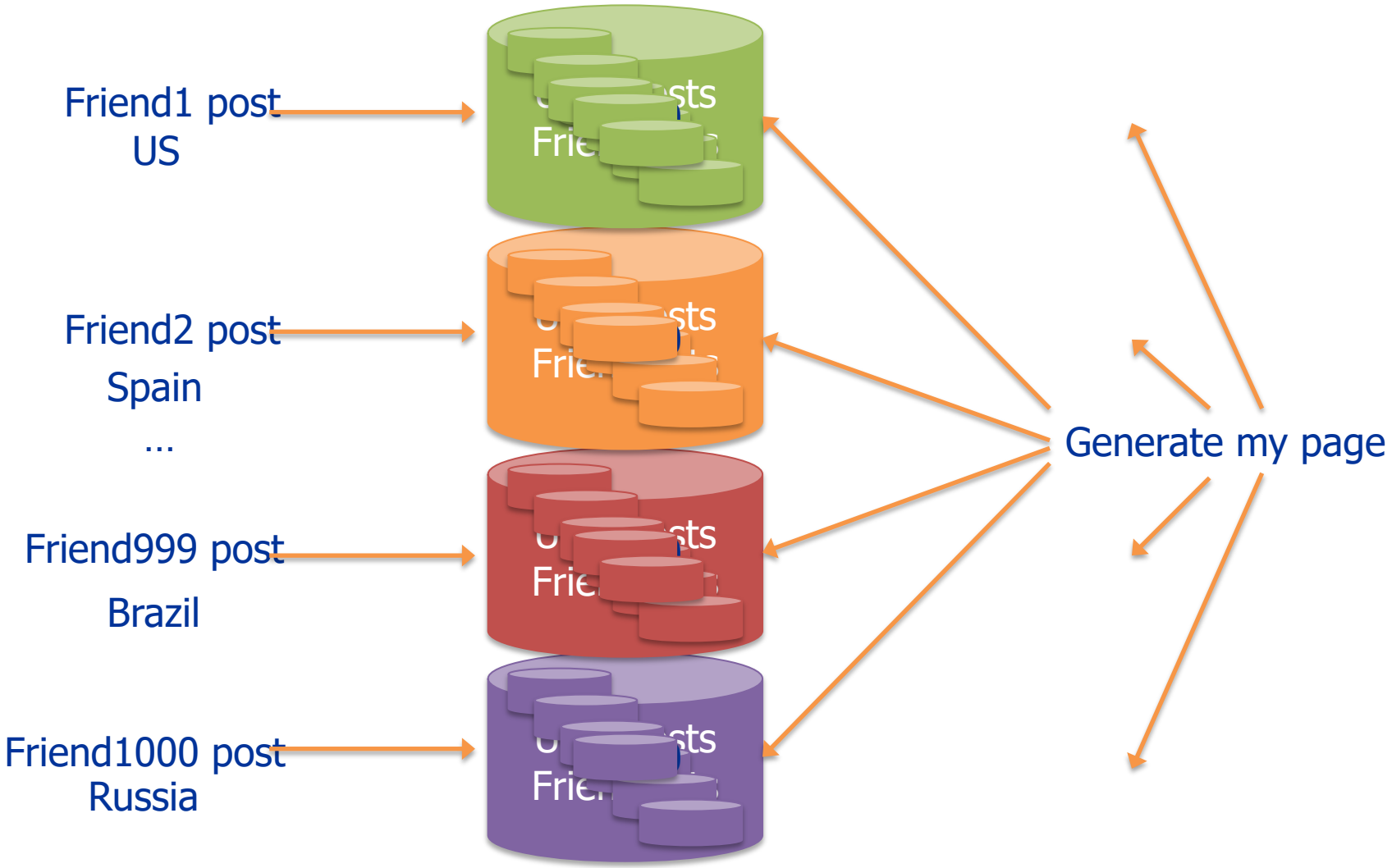
Single Machine



Multiple Machines



Multiple Data Centres



External Consistency

Consistent view

- Synchronised snapshot read of database
- Effect of past transactions should be seen and effect of future transactions should not be seen across data centres

Equivalent to linearizability

If transaction T1 commits before another transaction T2 starts, then T1's commit timestamp is smaller than T2's

Any read that sees T2 must see T1

Strongest consistency guarantee that can be achieved in practice

Version Management

Transactions that write use strict 2PL (Two phase locking)

- Each transaction T assigned timestamp s
- Data written by T timestamped with s

Time	<8	8	15
My friends	[X]	[]	
My posts			[P]
X's friends	[me]	[]	

Synchronising Snapshots

Global wall-clock time

==

External Consistency:

Commit order respects global wall-time order

==

Timestamp order respects global wall-time order

given

timestamp order == commit order

Read-Write Transactions

Use read locks on all data items that are read

- Acquired at leader
- Read latest version, not based on timestamp

Writes buffered, and acquire write locks at commit time (when prepare is done)

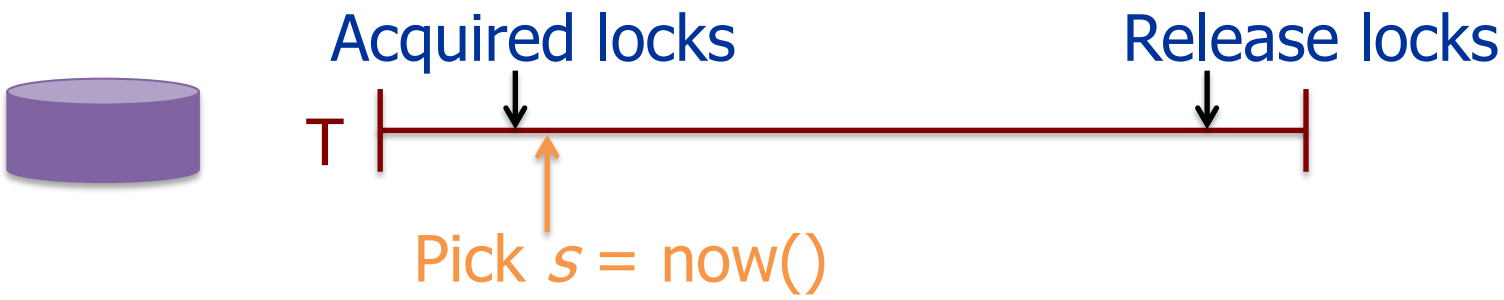
Timestamp assigned at commit time

- Data version written with commit timestamp

Timestamps, Global Clock

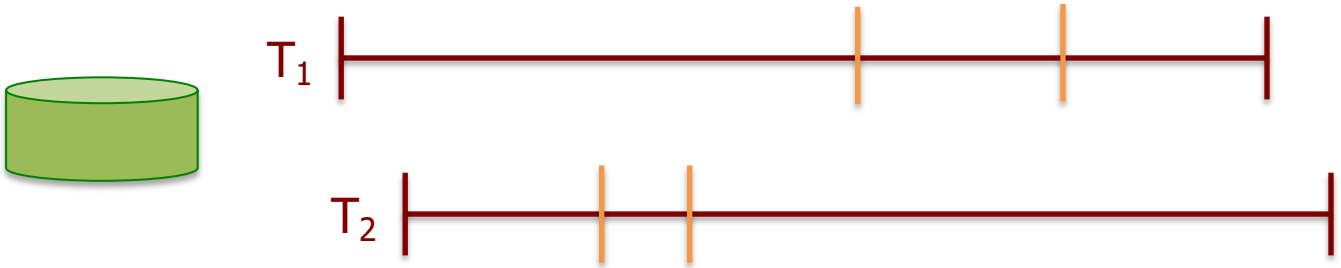
Strict two-phase locking for write transactions

Assign timestamp while locks are held

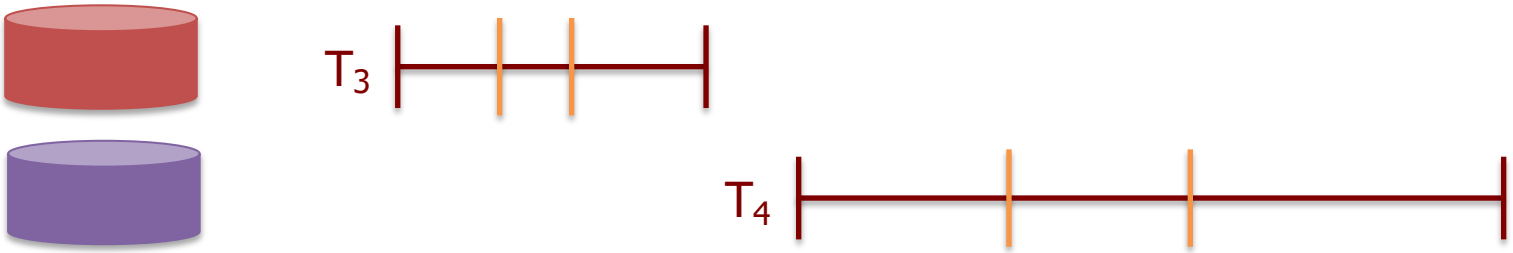


Timestamp Invariants

Timestamp order == commit order



Timestamp order respects global wall-time order

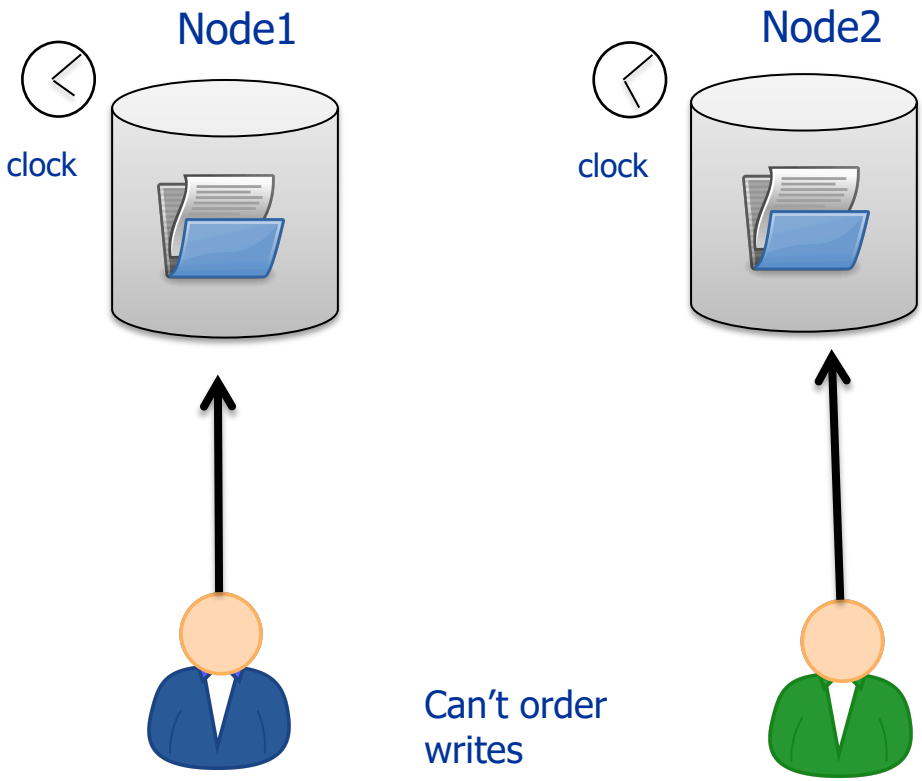


Clock Skew in Distributed Systems

Typically there is no global clock in distributed system

Individual nodes have local clocks with clock skew

- Means that timestamps lead to partial order



TrueTime I

Novel API behind Spanner's core innovation

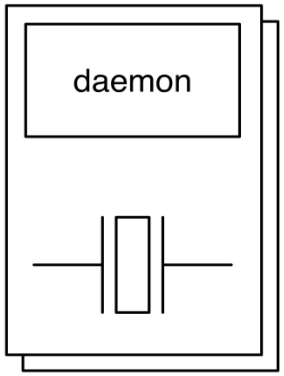
TrueTime

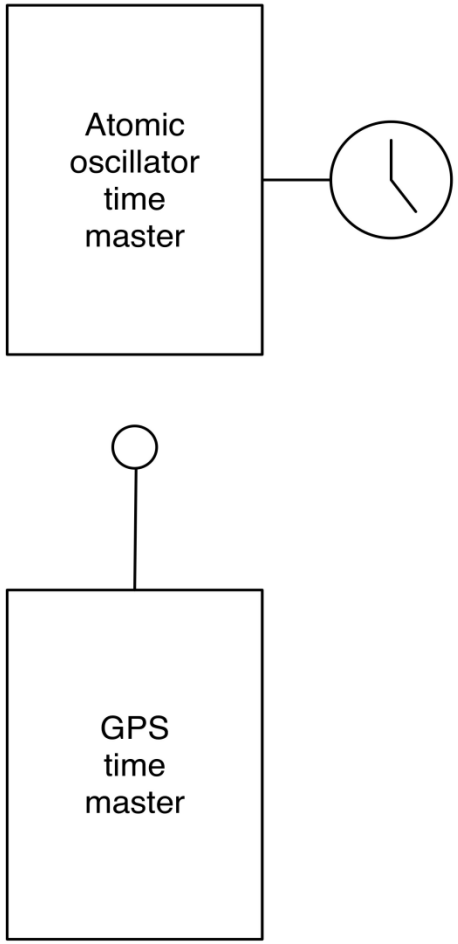
- Leverages hardware features like GPS and Atomic Clocks
- Implemented via TrueTime API
  - Key method is now(): not only returns current system time but also value (ε), which tells maximum uncertainty

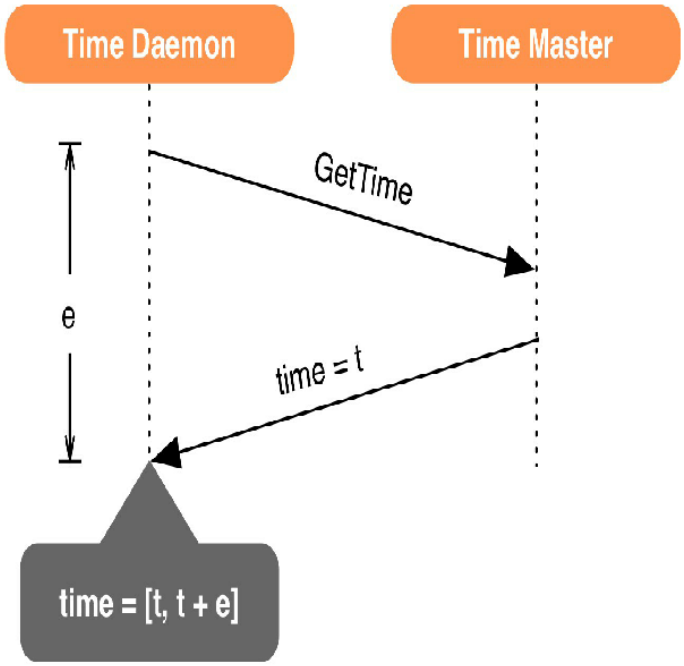
Set of time master servers per data centre and time slave daemon per machines

- Majority of time masters GPS fitted; few others atomic clock fitted (Armageddon masters)
- Daemon polls variety of masters and reaches consensus about correct timestamp

TrueTime II


daemon


Atomic
oscillator
time
master
GPS
time
master


Time Daemon
Time Master
GetTime
e
time = t
time = [t, t + e]

TrueTime III

TrueTime uses both GPS and Atomic clocks since they are different failure rates and scenarios

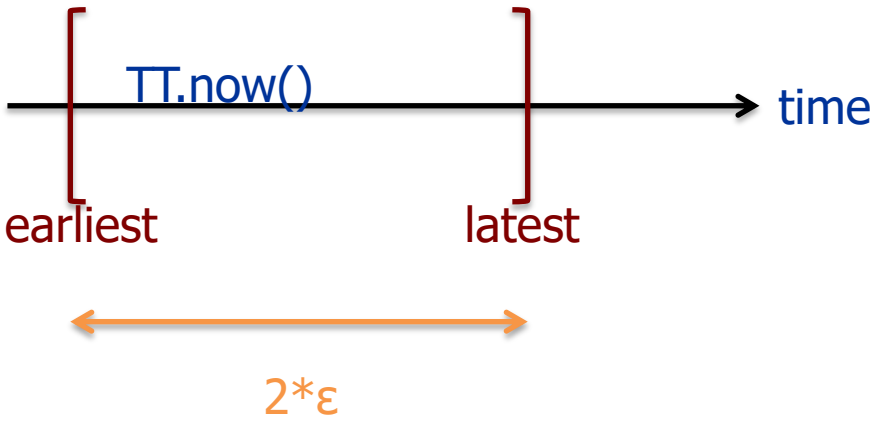
Two other boolean methods in API are

- After(t) – returns TRUE if t is definitely passed
- Before(t) – returns TRUE if t is definitely not arrived

TrueTime uses these methods in concurrency control and to serialise transactions

TrueTime

"Global wall-clock time" with bounded uncertainty



TrueTime-supported Transactions

Read-Write – requires locks

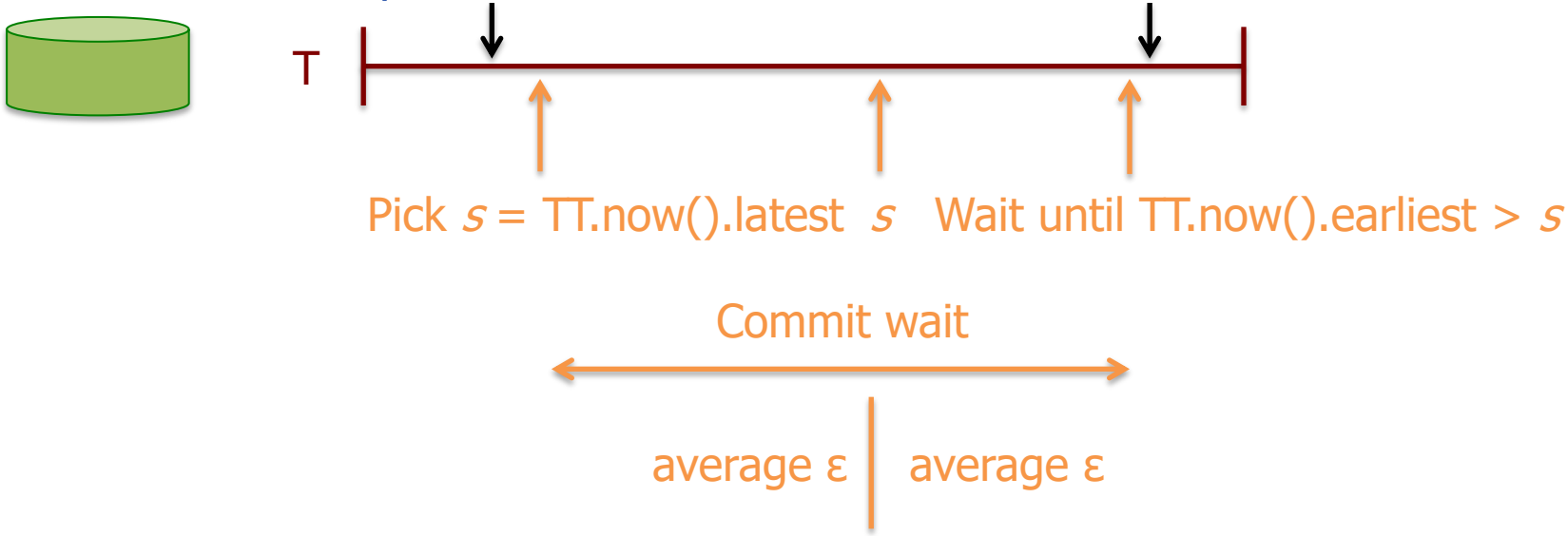
Read-Only – lock free

- Requires declaration before start of transaction
- Reads information that is up-to-date

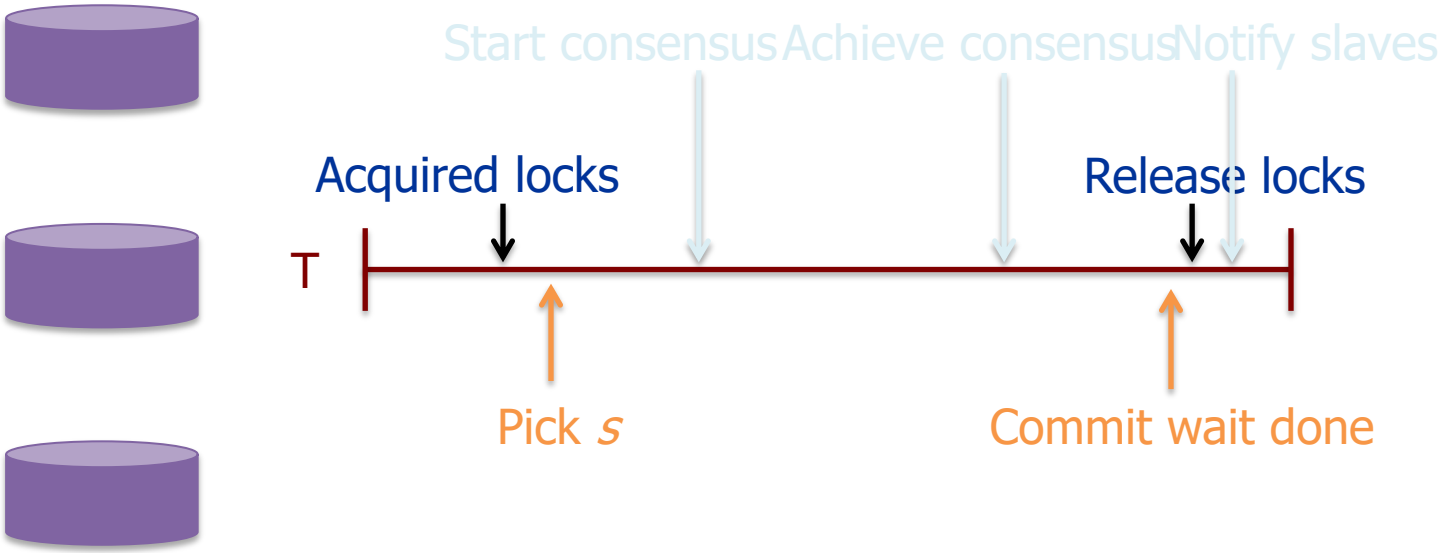
Snapshot-Read - read information from past by specifying timestamp or bound

- Use specific timestamp from past or timestamp bound so that data until that point will be read

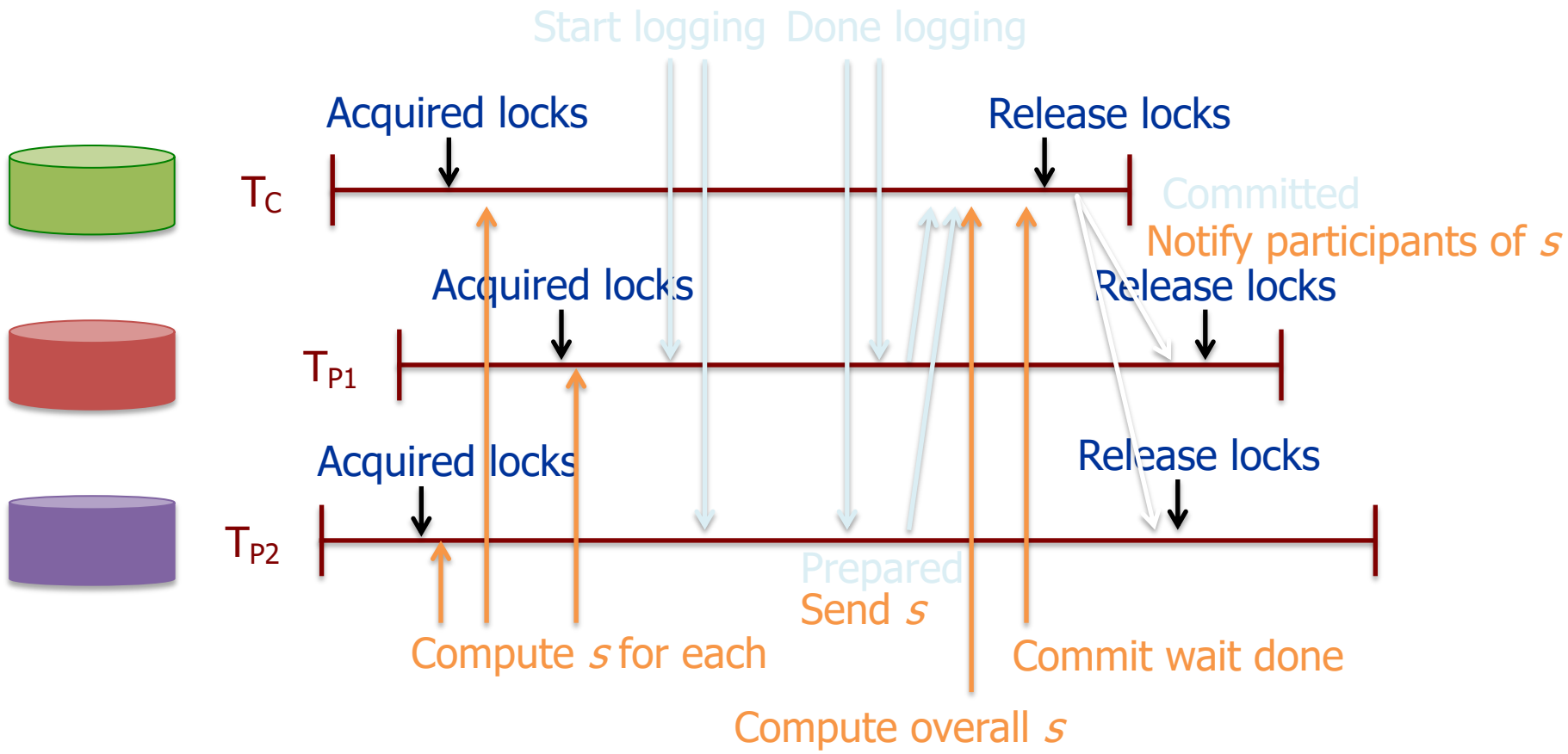
Timestamps + TrueTime



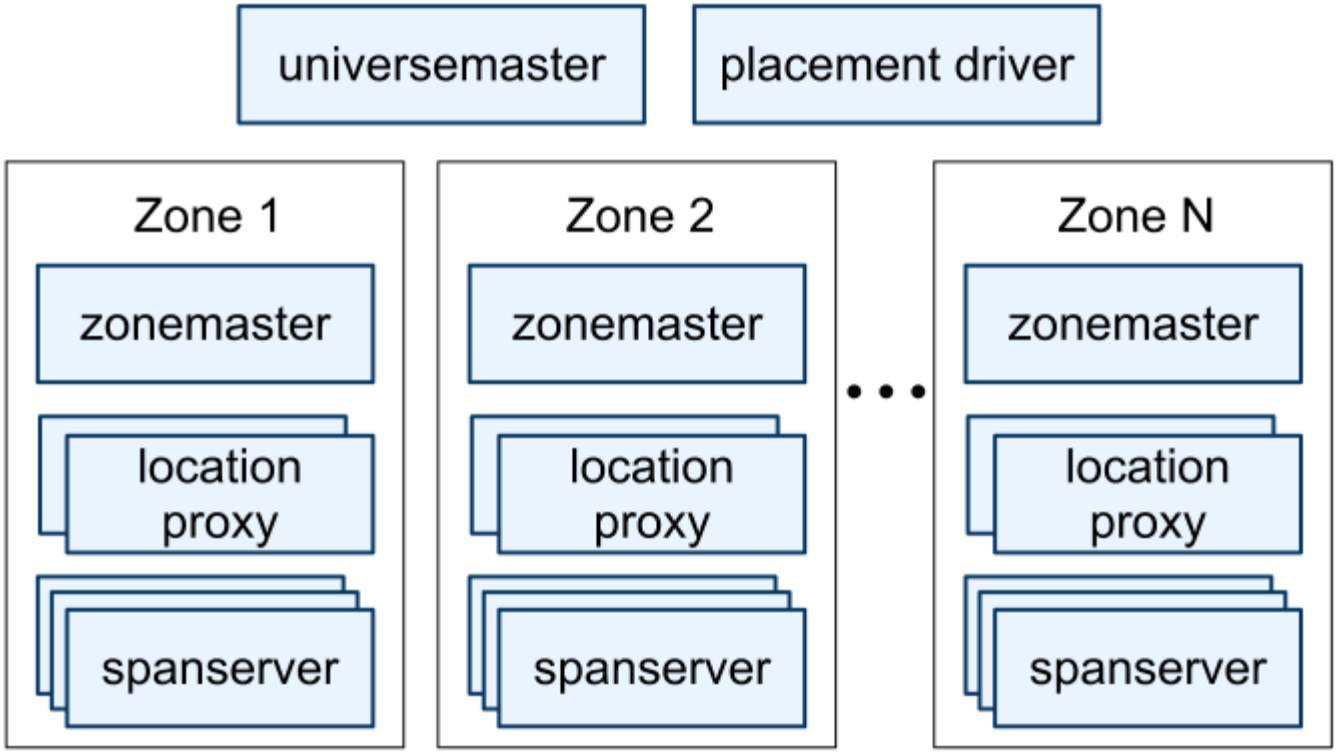
Commit Wait and Replication



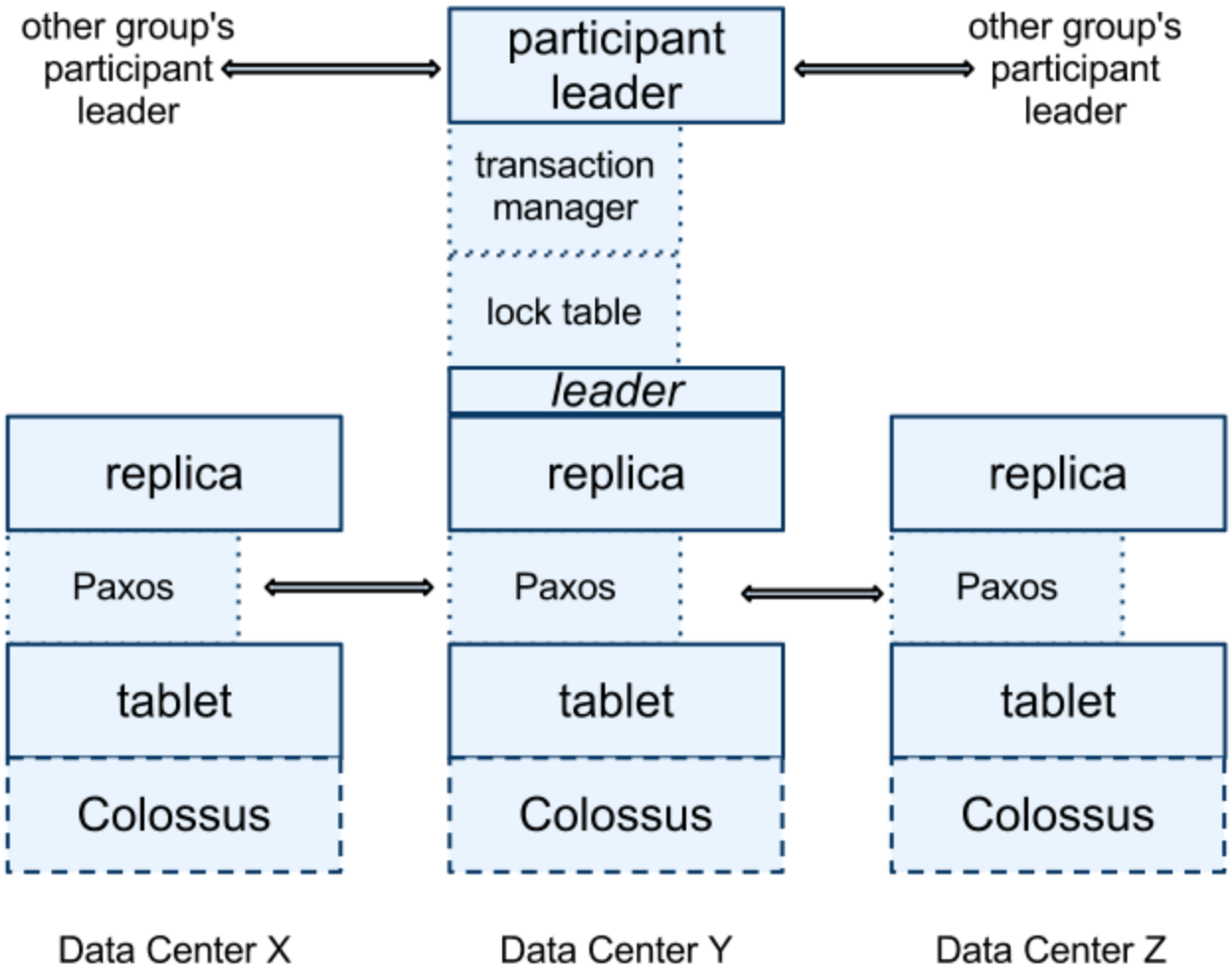
Commit Wait & 2-Phase Commit



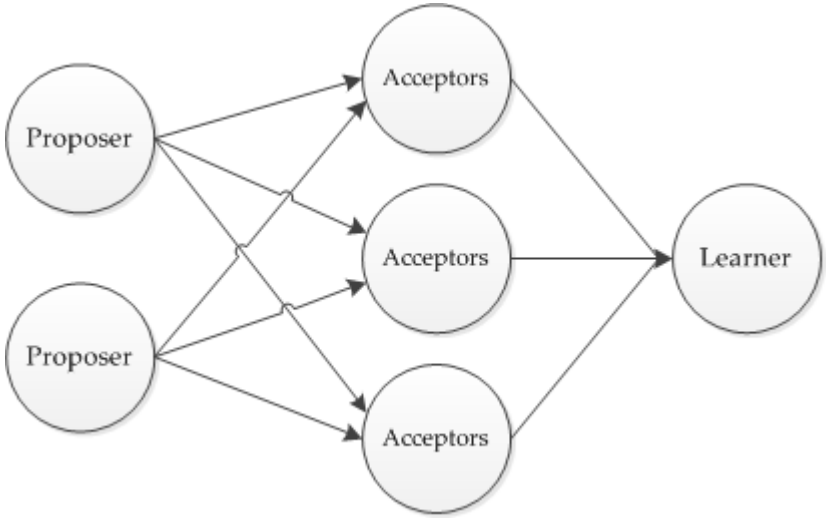
System Architecture



Software Stack



Paxos Algorithm



Paxos consensus algorithm (proposed by Leslie Lamport):

- Leader receives client's command
- Assigns it new command number i
- Runs i^{th} instance of consensus algorithm **in parallel**

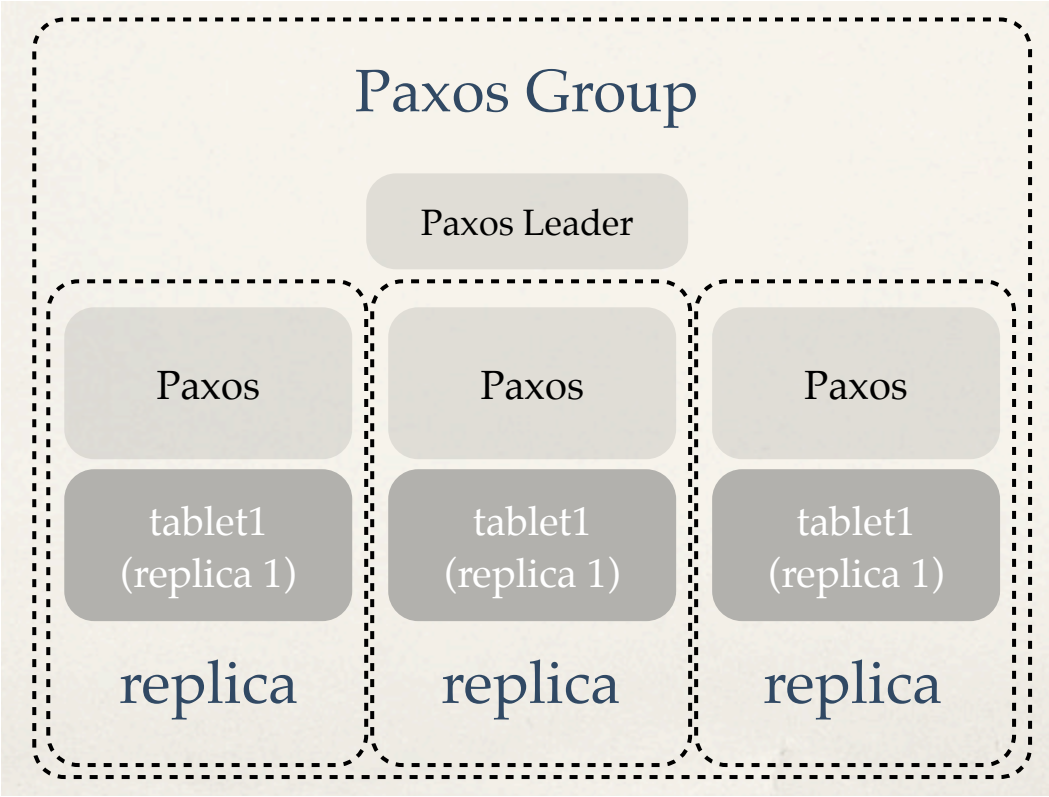
Paxos group: All machines involved in instance of Paxos

Within Paxos group, leader may fail and may need re-election, but safety properties always guaranteed

Paxos Groups in Spanner

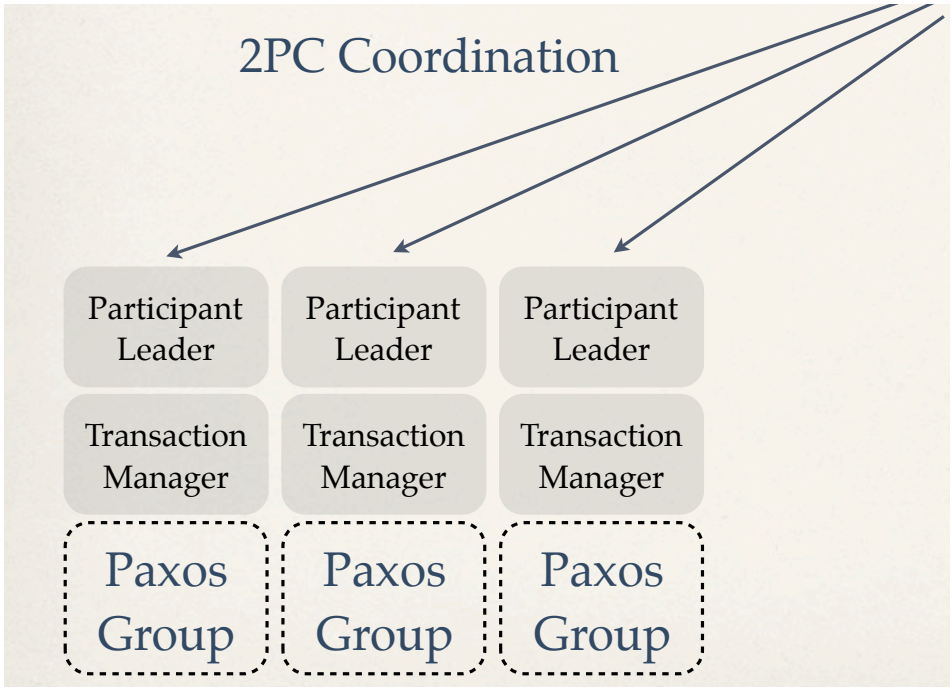
Tablets are replicated (between datacenters, possibly inter-continental), concurrency coordination by Paxos

A transaction needs consistency across its replicas; coordinated by Paxos



Paxos Group: A tablet and its replicas as well as the concurrency machinery across the replicas

Transactions with Multiple Paxos Groups



If transaction involves multiple Paxos Groups, use transaction management machinery atop of Paxos groups to coordinate 2PC

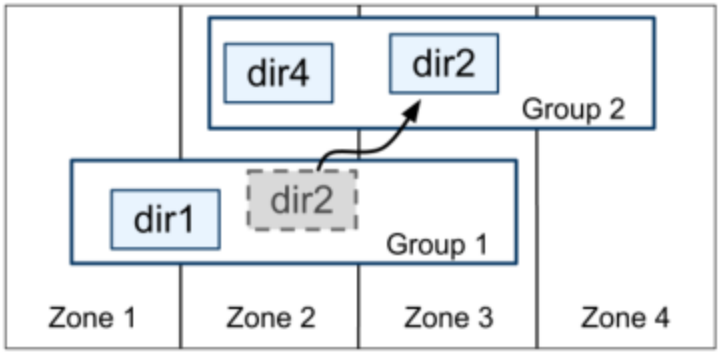
If transaction involves single Paxos Group, can bypass Transaction Manager and Participant Leader machinery

Data Chunks

Directory – analogous to bucket in BigTable

- Smallest unit of data placement
- Smallest unit to define replication properties

Directory might in turn be sharded into fragments if it grows too large



Evaluation

Evaluated for replication, transactions and availability

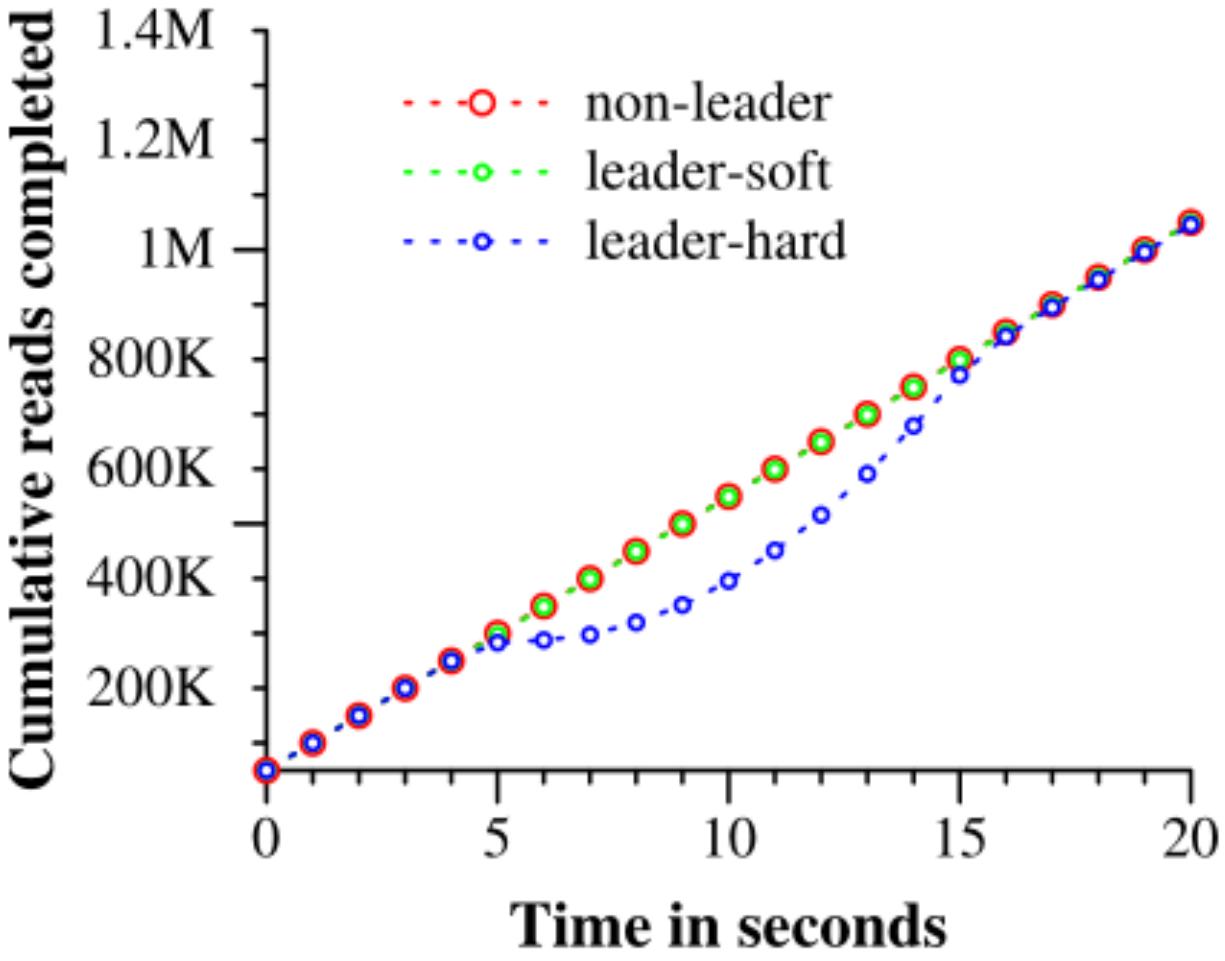
Results on epsilon of TrueTime

Benchmarked on Spanner deployment with

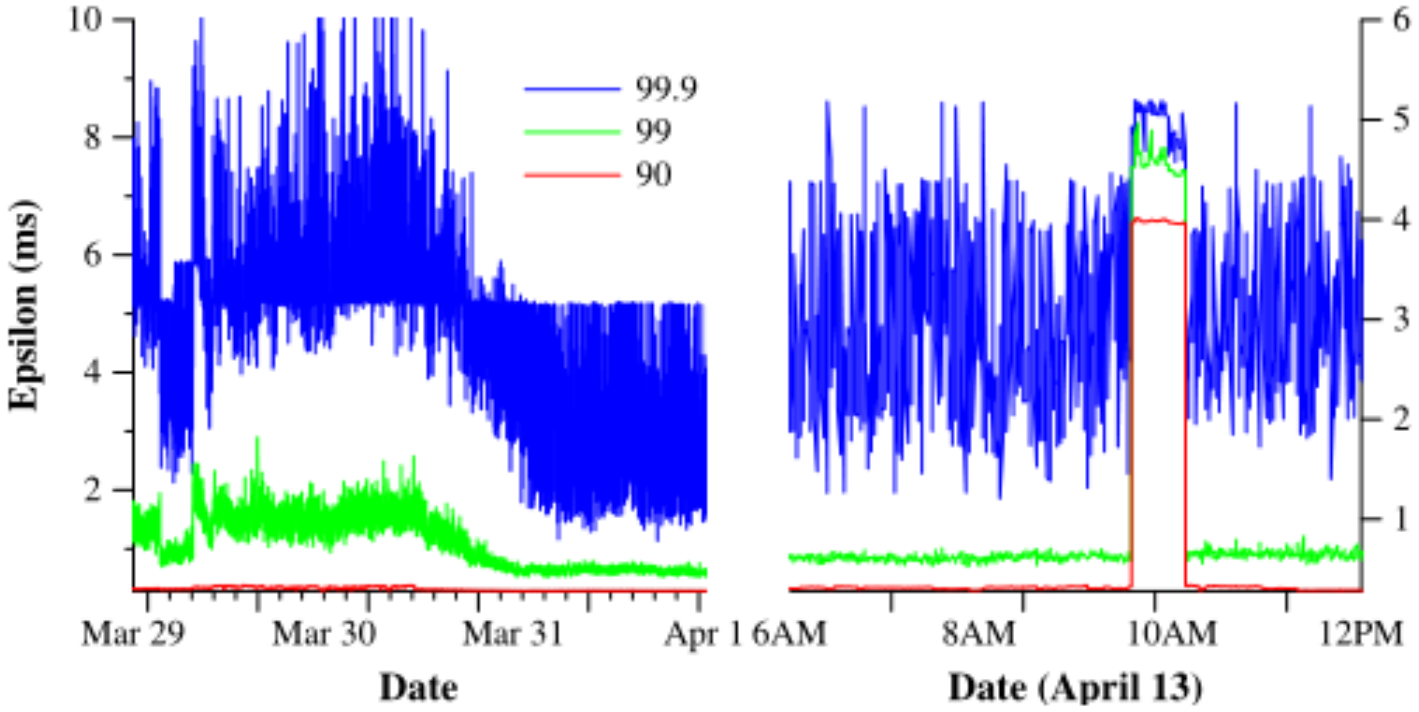
- 50 Paxos groups (doing replication)
- 250 Directories
- Clients (applicatons) & Zones are at network distance of 1 ms

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

Evaluation: Availability



Evaluation: Epsilon



Case Study: F1

Spanner in production: Google's advertising backend F1

F1 previously used MySQL

- Requires strong transactional semantics
- NoSQL database solution impractical

Spanner provides synchronous replication and automatic failover for F1

Enabled F1 to specify data placement via directories of Spanner based on their needs

F1 operation latencies measured over 24 hours

operation	latency (ms)		count
	mean	std dev	
all reads	8.7	376.4	21.5B
single-site commit	72.3	112.8	31.2M
multi-site commit	103.0	52.2	32.1M