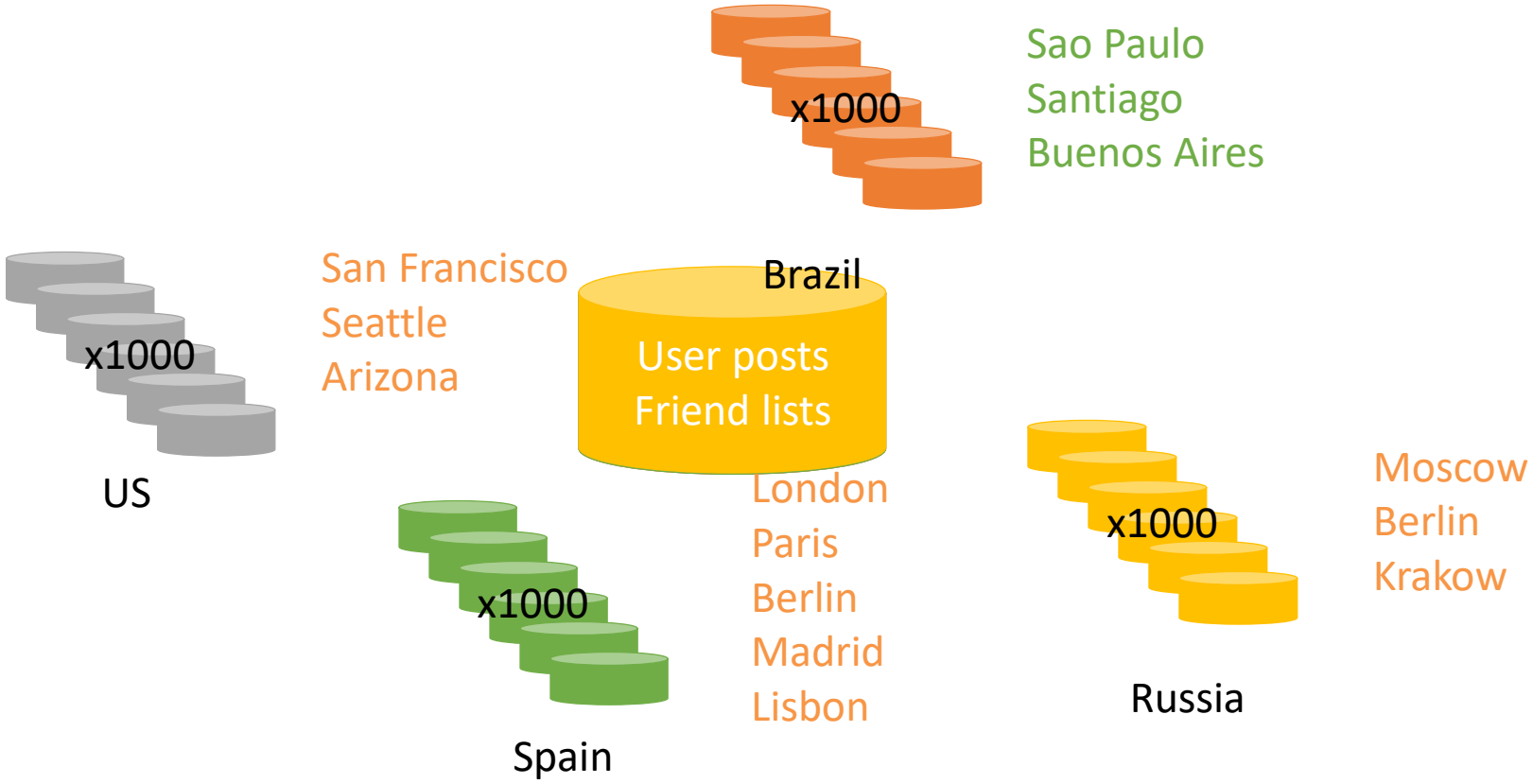
Spanner: Google's Globally Distributed Database

Sadman Sakib

What is Spanner?

- Distributed multiversion database
  - General-purpose transactions (ACID)
  - SQL query language
  - Schematized tables
  - Semi-relational data model
- Running in production
  - Storage for Google’s ad data
  - Replaced a sharded MySQL database

Example: Social Network

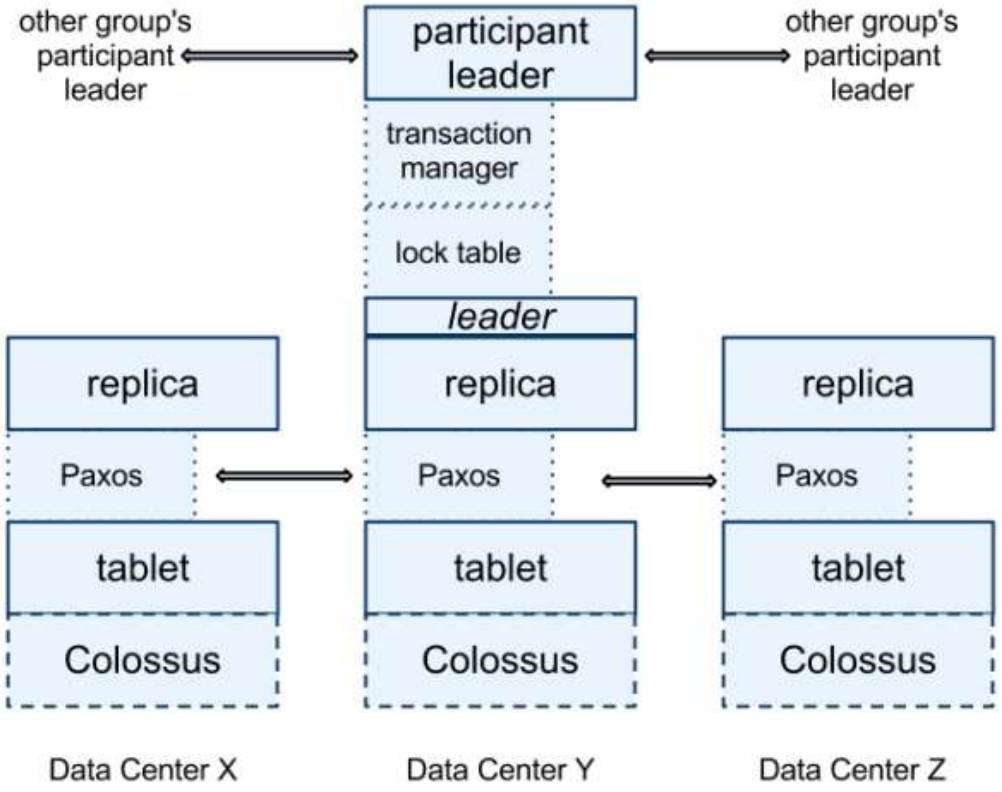


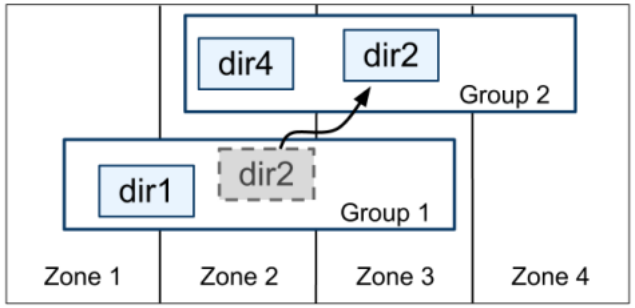
Overview

- Feature: Lock-free distributed read transactions
- Property: External consistency of distributed transactions
- Implementation: Integration of concurrency control, replication, and 2PC
  - correctness and performance
- Enabling technology: TrueTime
  - interval-based global time

Node Software

- Tablet: a bag of mappings
  - (key, timestamp) -> string
- Colossus: Distributed file system
  - B-tree-like files & write-ahead log
- Paxos: consistency protocol
  - Replicated bag of mappings
- Lock table: Two-phase lock state
- Directory: unit of data movement



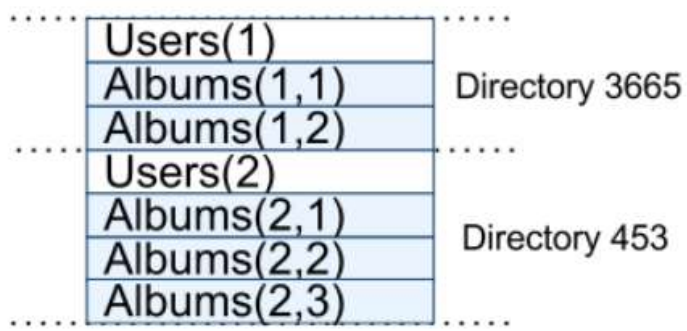


Data Model

- Data features
  - Schematized semi-relational
  - SQL-like Query language
  - General purpose transactions
- Table: relational + key-value store
  - Required primary-keys
- Database: hierarchies of tables

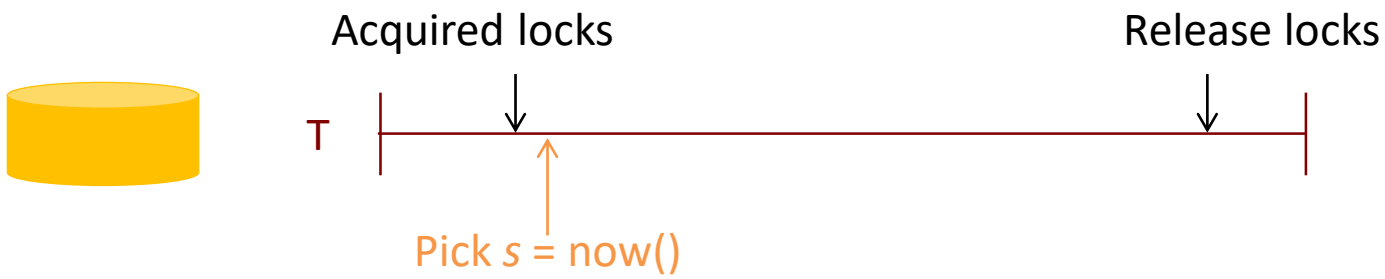
```
CREATE TABLE Users {
  uid INT64 NOT NULL, email STRING
} PRIMARY KEY (uid), DIRECTORY;

CREATE TABLE Albums {
  uid INT64 NOT NULL, aid INT64 NOT NULL,
  name STRING
} PRIMARY KEY (uid, aid),
  INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```



Timestamps

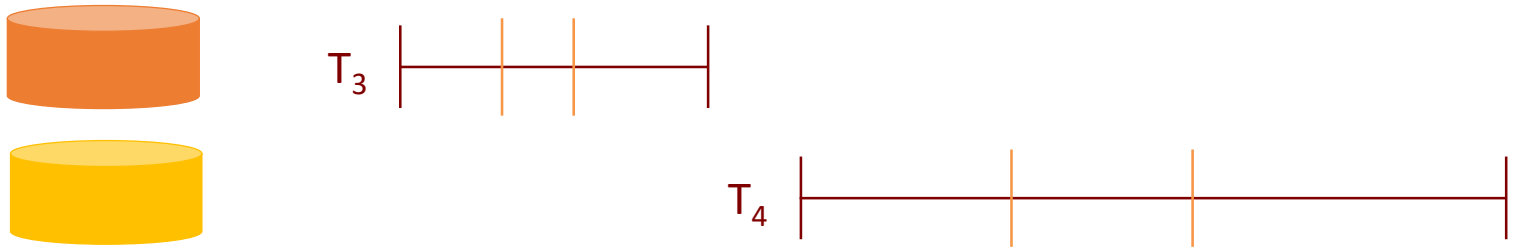
- Strict two-phase locking for write transactions
- Assign timestamps while locks are held
- Data written by T is timestamped with s.



Timestamp Invariants

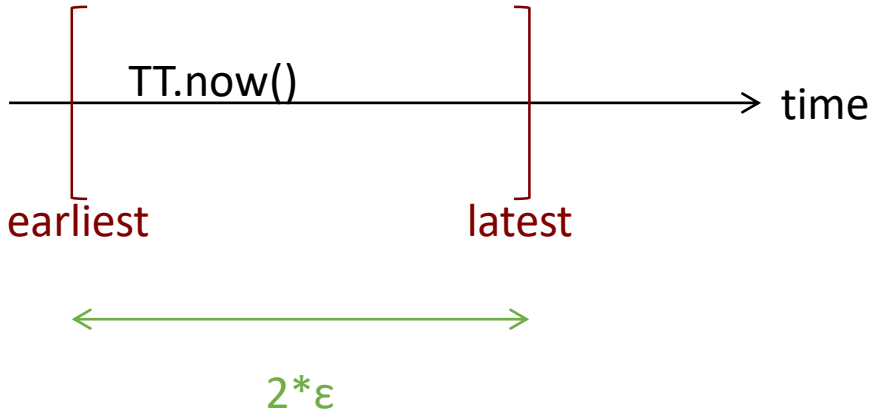
- Timestamp order == commit order

- Timestamps order respects global wall-time order

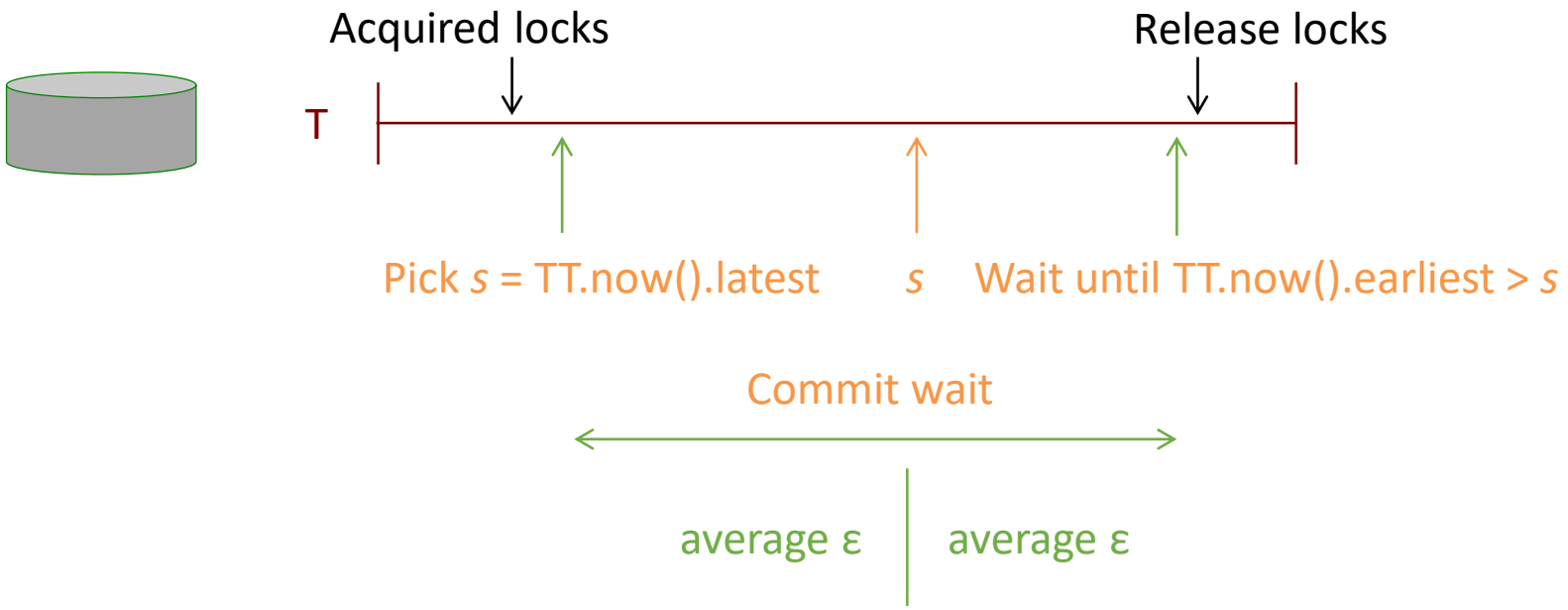


TrueTime

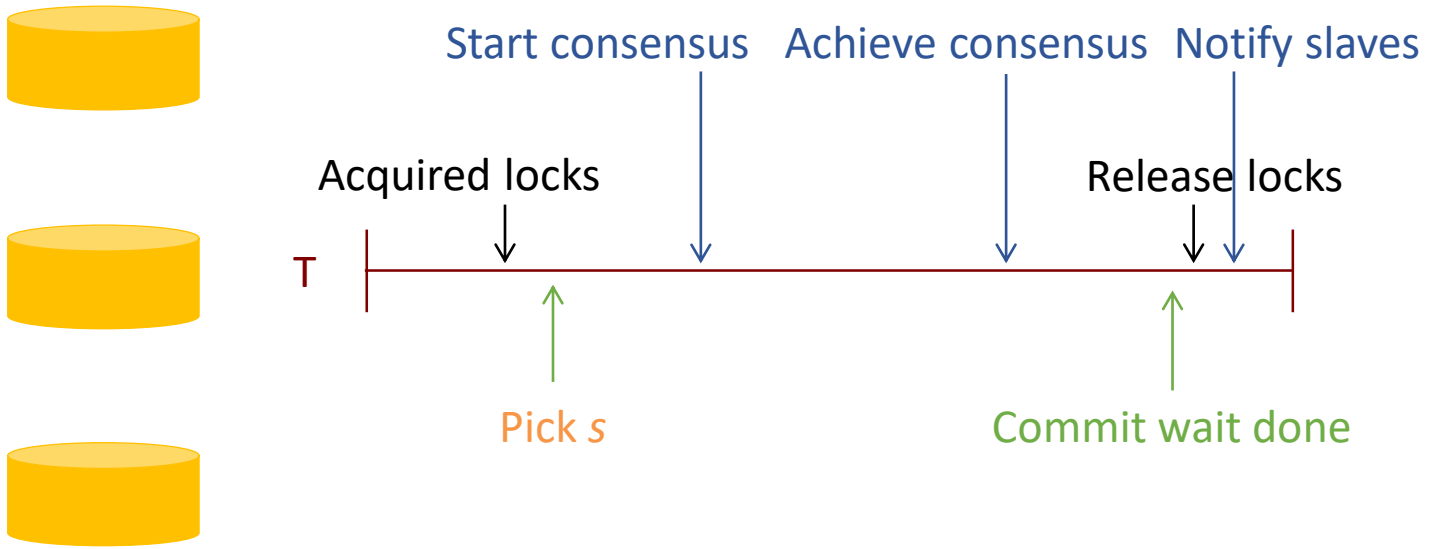
- Global wall-clock time with bounded uncertainty



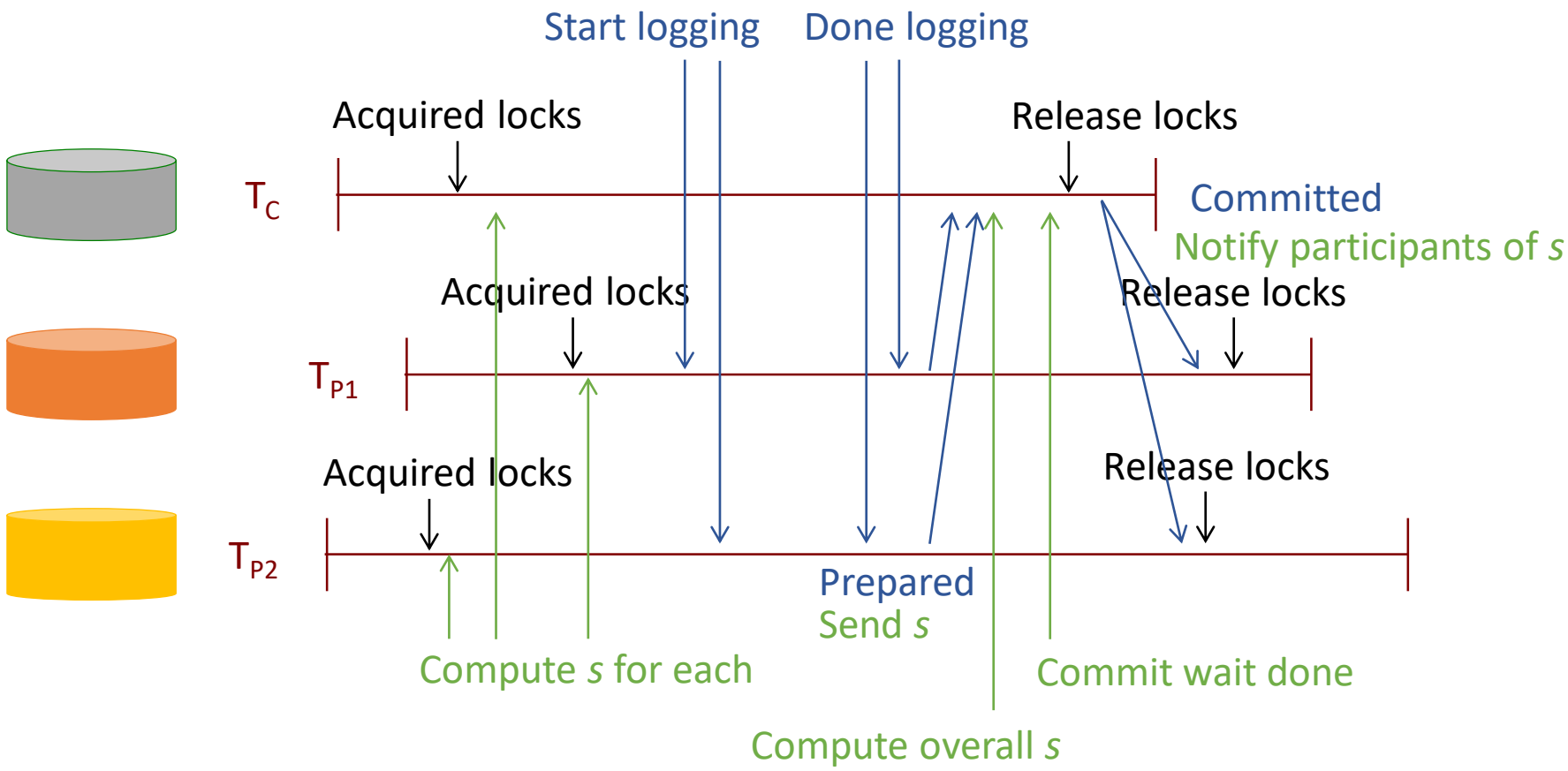
Commit Wait


Acquired locks
Release locks
T
Pick s = TT.now().latest
s
Wait until TT.now().earliest > s
Commit wait
average ε
average ε

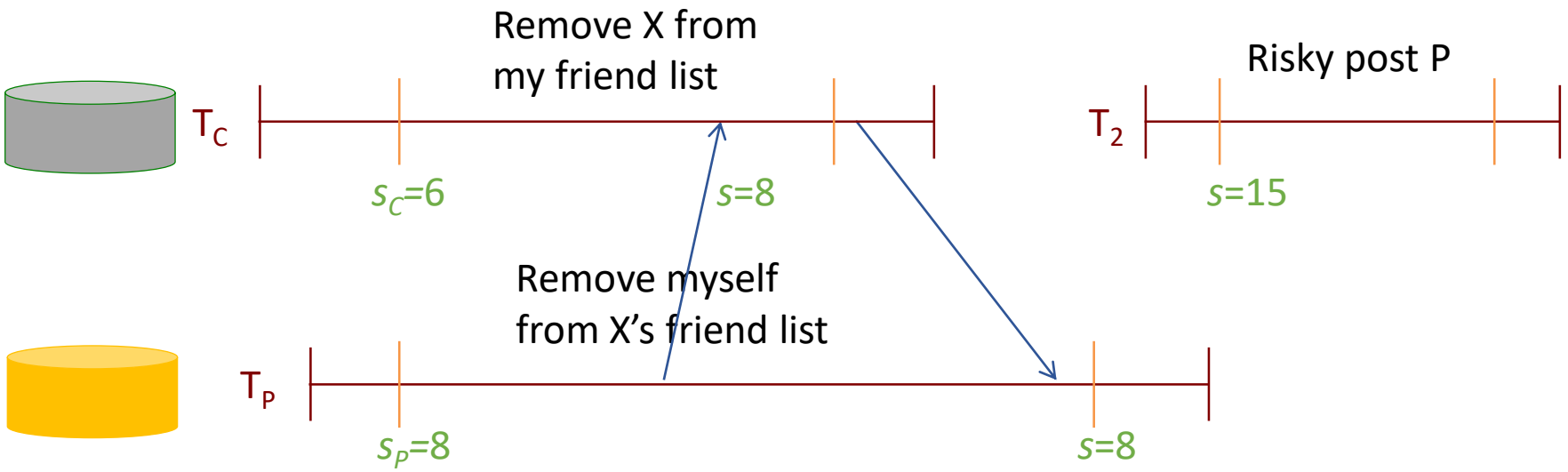
Commit Wait and Replication



Commit Wait and 2-Phase Commit

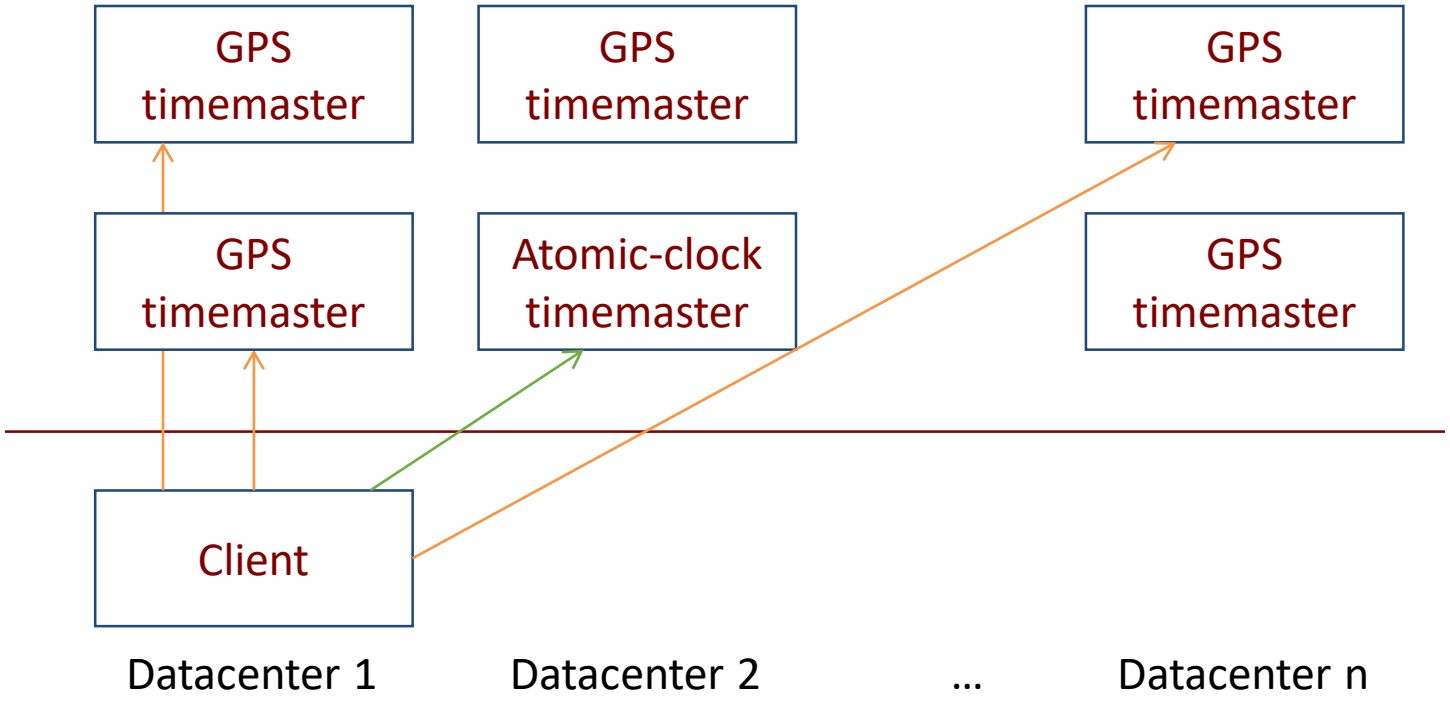


Example



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

TrueTime Architecture

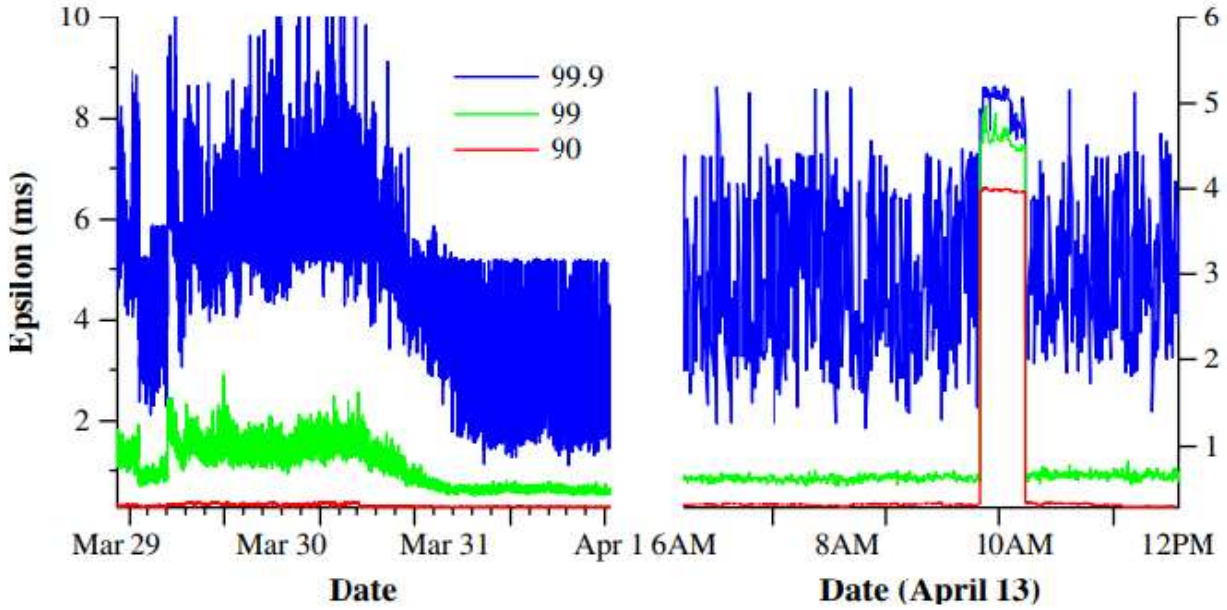


Compute reference [earliest, latest] = now ± ε

Evaluation

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



Distribution of TrueTime ϵ values

Conclusion

- Versioning to accurately synchronize a distributed database.
- Interval-based global time ensures strong consistency
- Timestamping provides non-blocking snapshot reads, lock-free read-only transactions.