Distributed Systems

CS425/ECE428

Instructor: Radhika Mittal

Acknowledgements for the materials: Spanner authors

Logistics

- MP2 due soon (Monday).

Distributed Transactions and Replication

- Transaction processing can be *distributed* across multiple servers.
  - Different objects can be stored on different servers.
  - An object may be replicated across multiple servers.
- **Case study: Google's Spanner System**
  - Note for exams:
    - no detailed questions from Spanner paper.
    - only some high-level objective questions from materials in slides.

Spanner: Google's Globally-Distributed Database

- First three lines from the paper:
  - Spanner is a scalable, globally-distributed database designed, built, and deployed at Google.
  - At the highest level of abstraction, it is a database that shards data across many sets of Paxos state machines in datacenters spread all over the world.
  - Replication is used for global availability and geographic locality; clients automatically failover between replicas.

Spanner: Google's Globally-Distributed Database

Wilson Hsieh
representing a host of authors

OSDI 2012

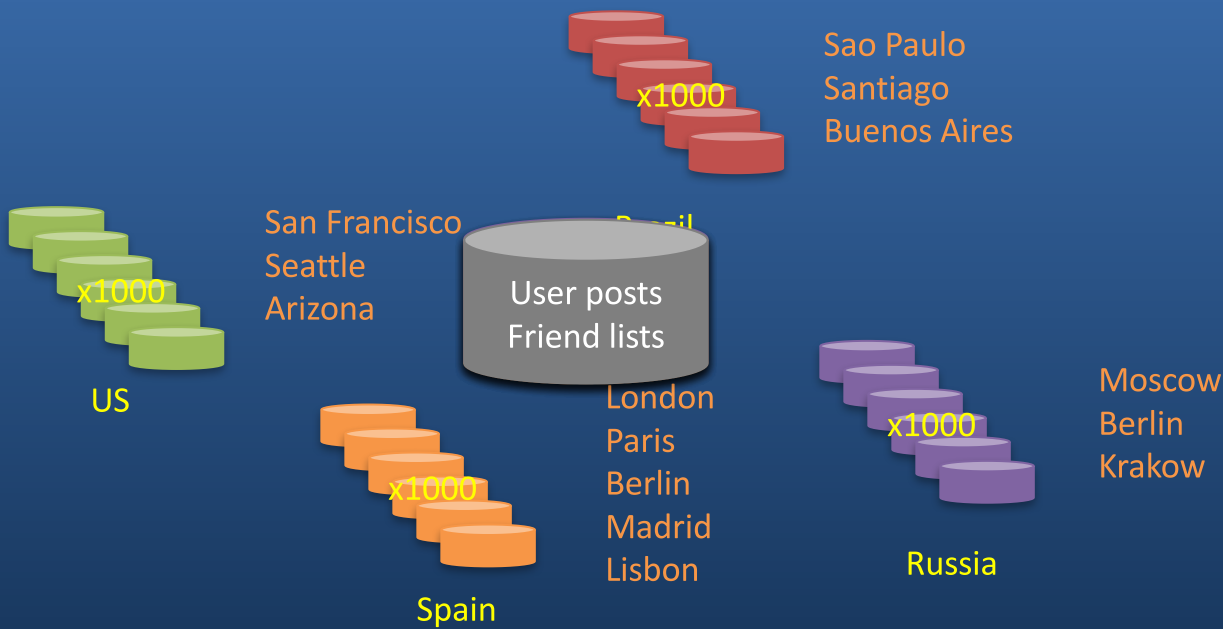
Google™

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
  - First system at global scale
- Implementation: Integration of concurrency control, replication, and 2PC
  - Correctness and performance
- Enabling technology: TrueTime
  - Interval-based global time

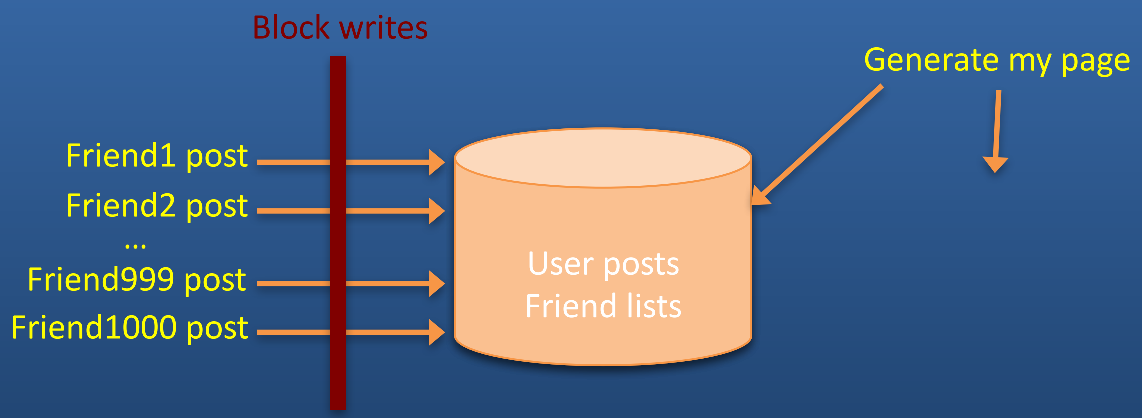
Read Transactions

- Generate a page of friends' recent posts
  - Consistent view of friend list and their posts

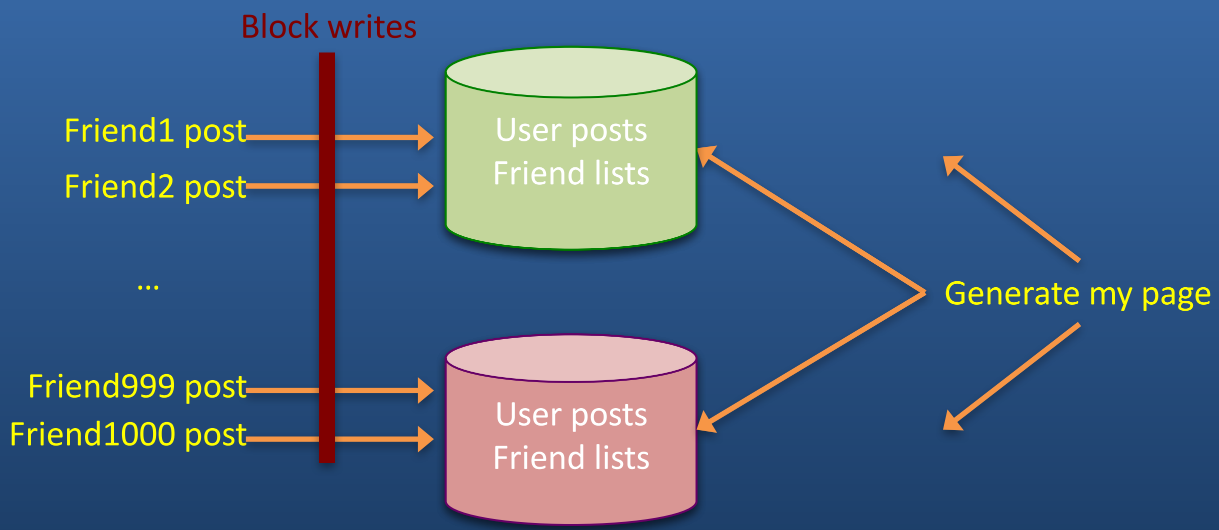
Why consistency matters

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

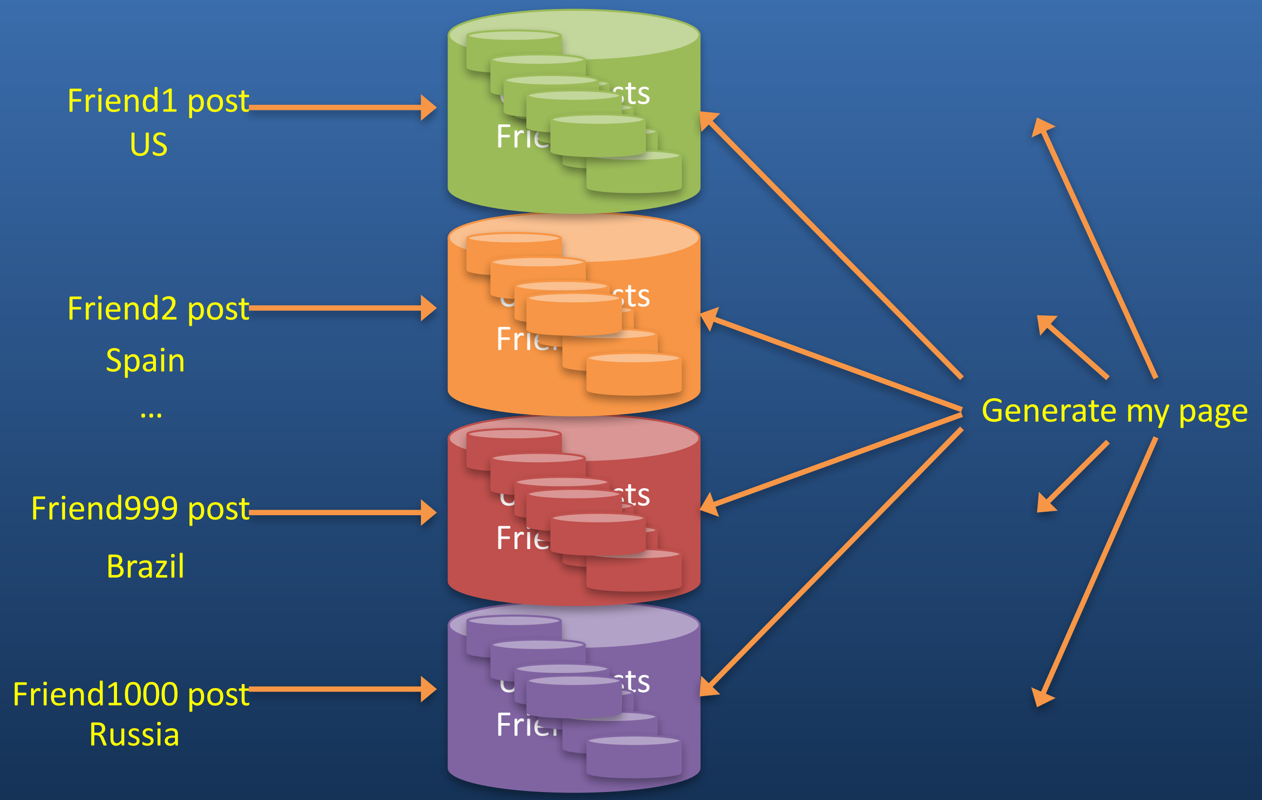
Single Machine



Multiple Machines



Multiple Datacenters



Version Management

- Transactions that write use strict 2PL
  - Each transaction *T* is assigned a timestamp *s*
  - Data written by *T* is timestamped with *s*

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

Synchronizing Snapshots

Global wall-clock time

==

External Consistency:

Commit order respects global wall-time order

==

Timestamp order respects global wall-time order

given

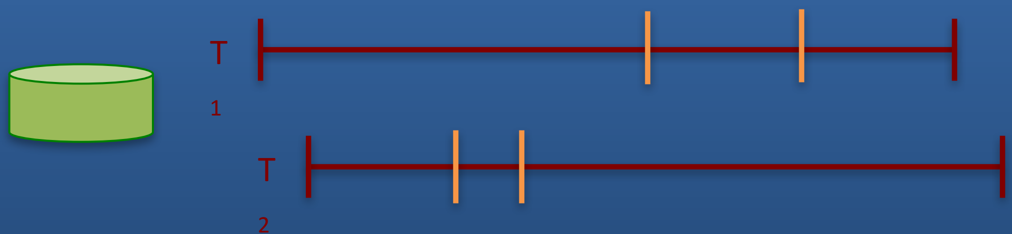
timestamp order == commit order

Timestamps, Global Clock

- Strict two-phase locking for write transactions
- Assign timestamp while locks are held

Timestamp Invariants

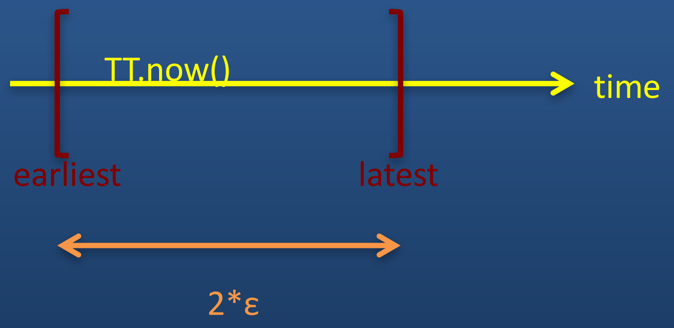
- Timestamp order == commit order



- Timestamp order respects global wall-time order

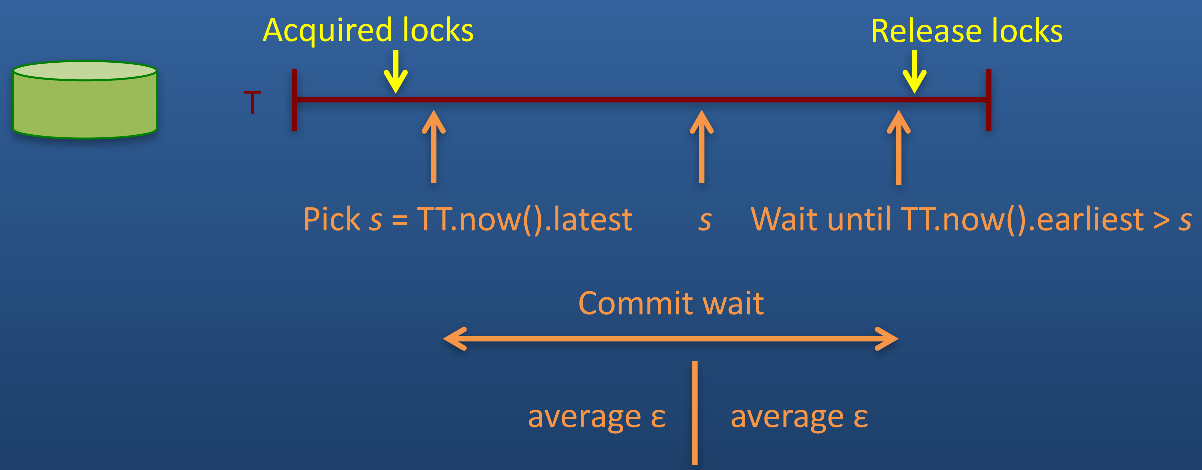
TrueTime

- "Global wall-clock time" with bounded uncertainty

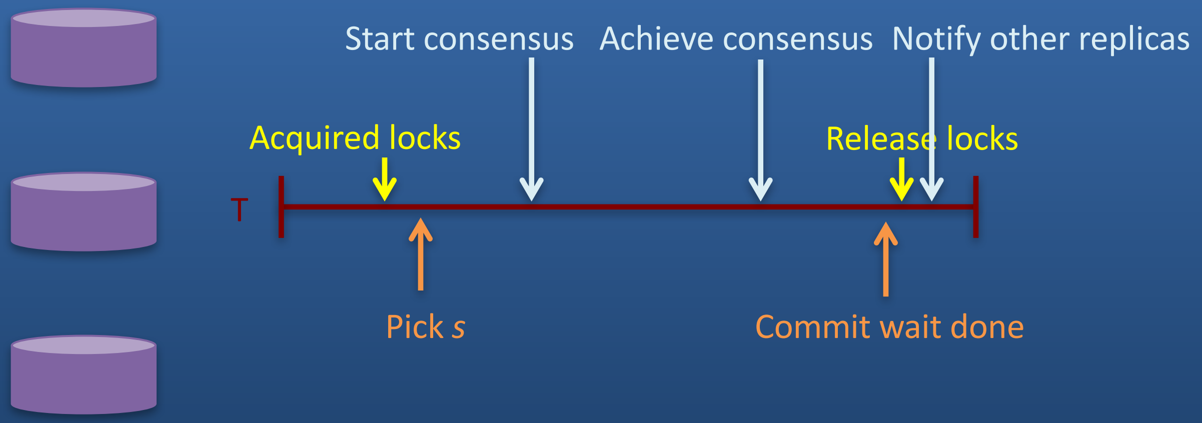


Google™

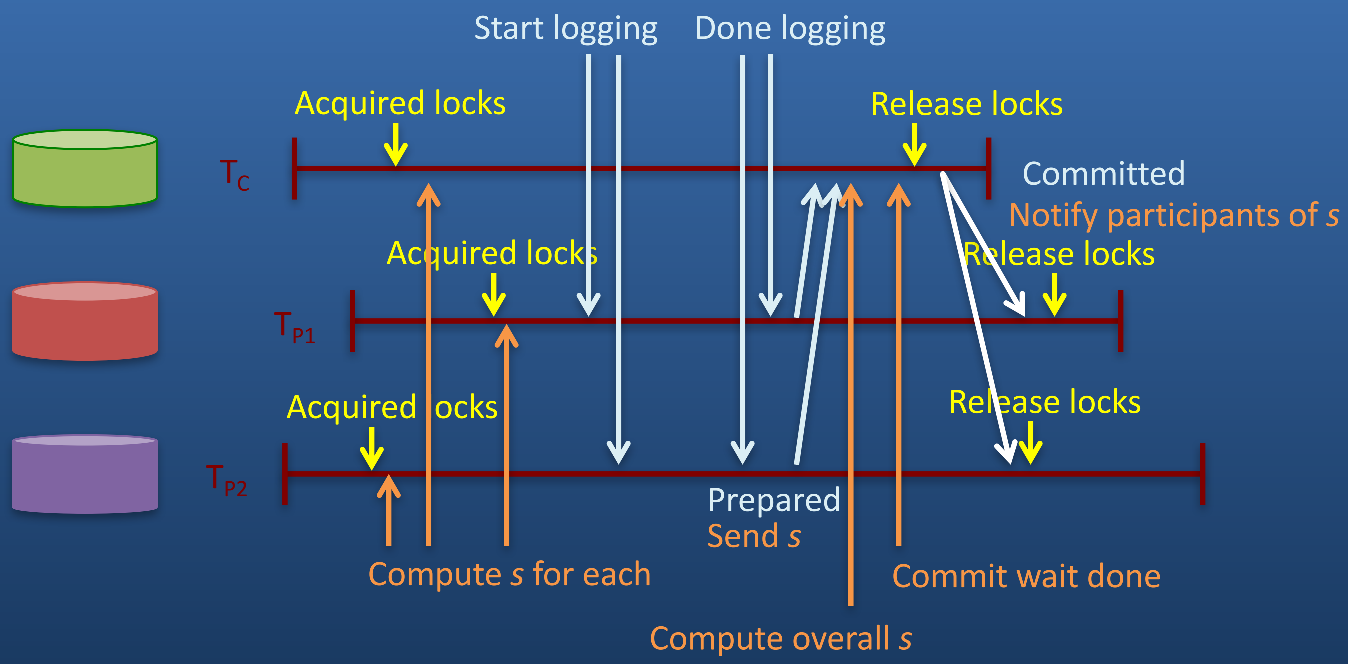
Timestamps and TrueTime



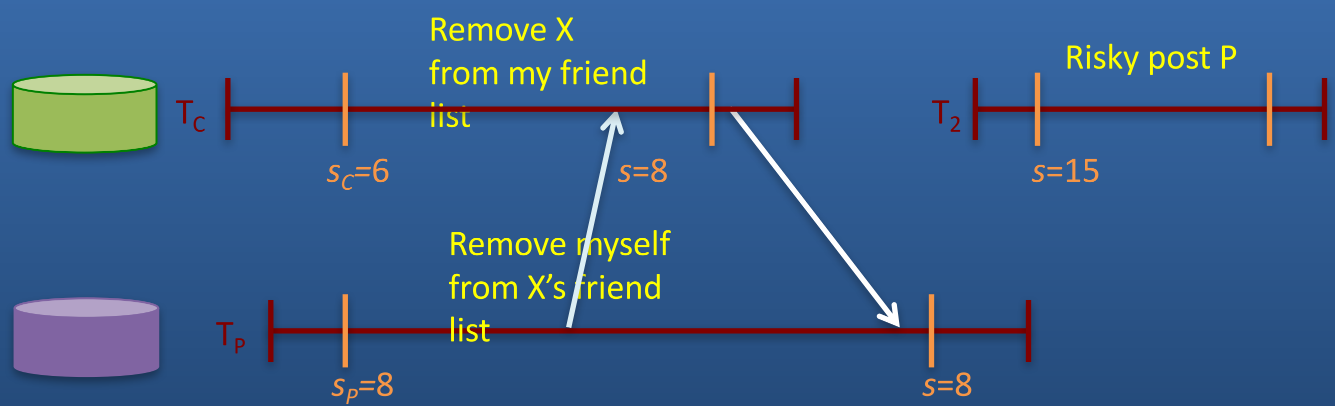
Commit Wait and Replication



Commit Wait and 2-Phase Commit



Example



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

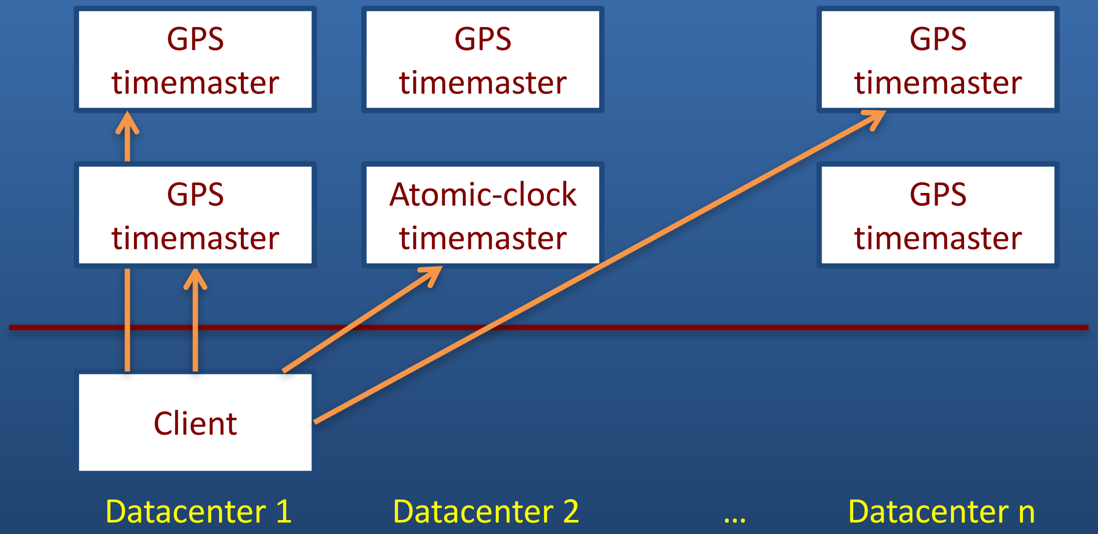
What Have We Covered?

- Lock-free read transactions across datacenters
- External consistency
- Timestamp assignment
- TrueTime
  - Uncertainty in time can be waited out

What Haven't We Covered?

- How to read at the present time
- Atomic schema changes
  - Mostly non-blocking
  - Commit in the future
- Non-blocking reads in the past
  - At any sufficiently up-to-date replica

TrueTime Architecture

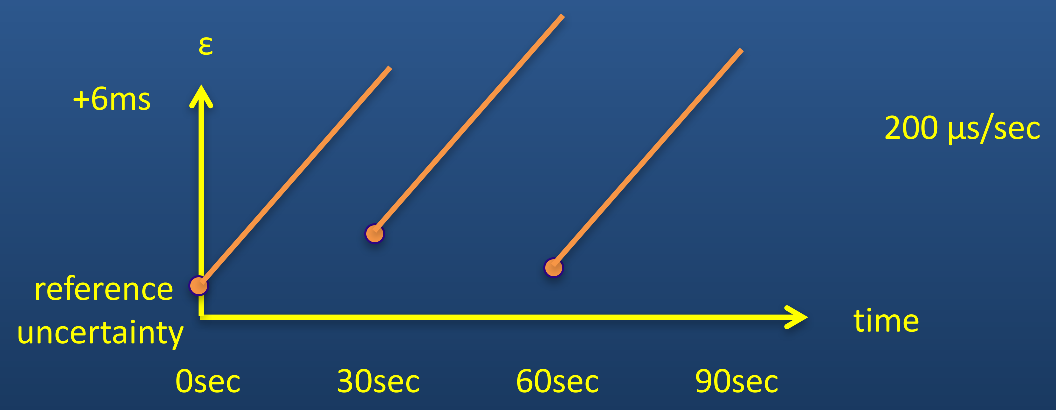


Compute reference [earliest, latest] = now ± ε

TrueTime implementation

now = reference now + local-clock offset

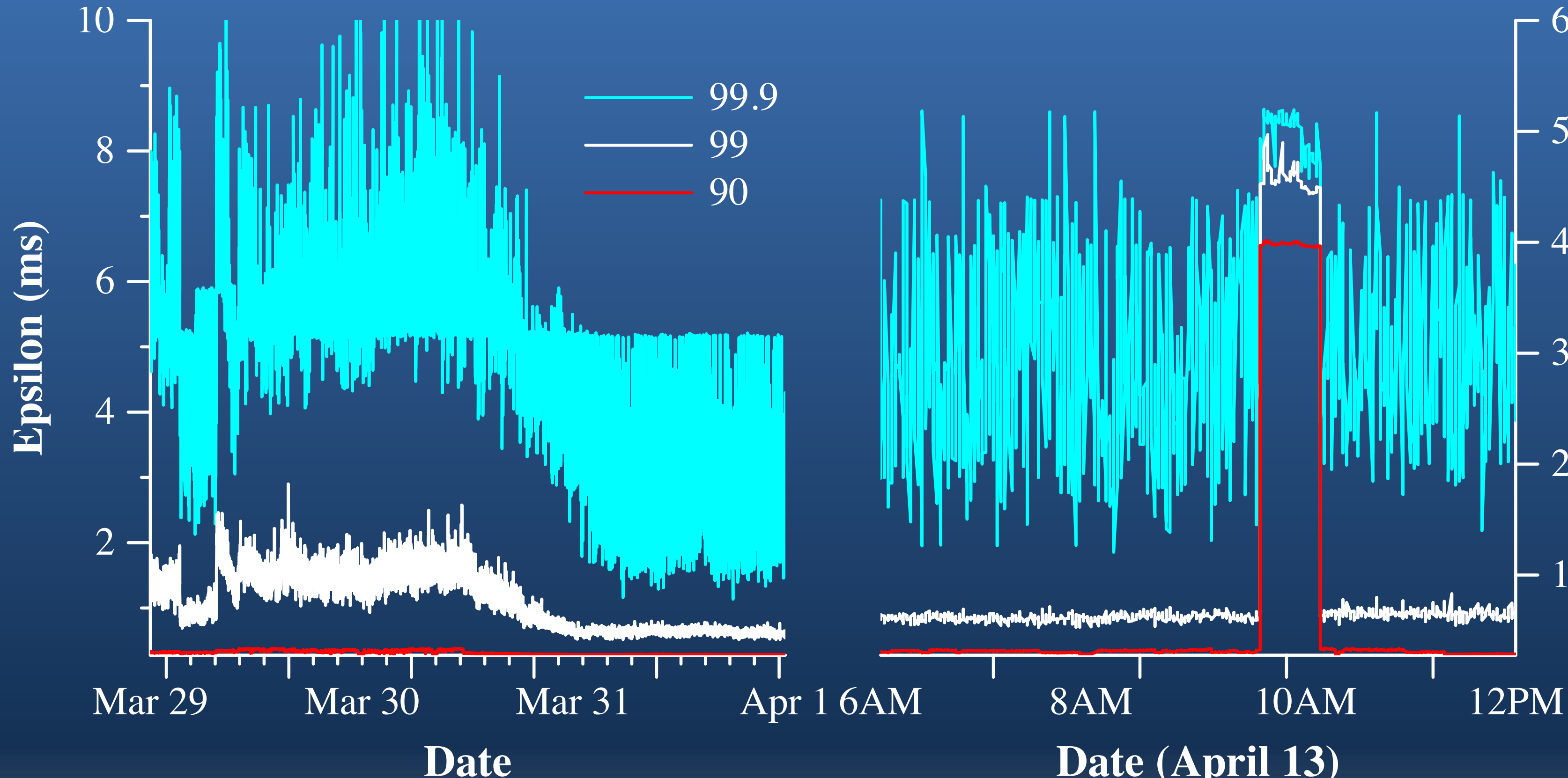
ε = reference ε + worst-case local-clock drift



What If a Clock Goes Rogue?

- Timestamp assignment would violate external consistency
- Empirically unlikely based on 1 year of data
  - Bad CPUs 6 times more likely than bad clocks

Network-Induced Uncertainty



What's in the Literature

- External consistency/linearizability
- Distributed databases
- Concurrency control
- Replication
- Time (NTP, Marzullo)

Future Work

- Improving TrueTime
  - Lower ε < 1 ms
- Building out database features
  - Finish implementing basic features
  - Efficiently support rich query patterns

Conclusions

- Reify clock uncertainty in time APIs
  - Known unknowns are better than unknown unknowns
  - Rethink algorithms to make use of uncertainty
- Stronger semantics are achievable
  - Greater scale != weaker semantics

Thanks

- To the Spanner team and customers
- To our shepherd and reviewers
- To lots of Googlers for feedback
- To you for listening!

- Questions?