

CPSC 416 – Distributed Systems

Lecture 19: Geo-Replication & Spanner

UBC Vancouver

2026-03-17

From Last Time

Turn to your neighbor. 2 minutes.

CAP says: pick two of three.

But real systems don't just "pick." What does a real system actually give up, and when?

(Pairs: 2 minutes)

Checkpoint 1

CAP is about *partitions*.

When the network is healthy, you can have consistency *and* availability.

The question is: what happens when a partition occurs?

Today: A system that chooses strong consistency, accepts the latency cost, and engineers around partitions.

Failure-First Question

Your data centers are 100ms apart.

A user in Tokyo writes. A user in London reads 50ms later.

What do they see?

- If eventual consistency: maybe the old value
- If strong consistency: the write, but at what cost?
- If the Tokyo DC is partitioned: ...?

L10's Grid Returns

Recall from L10: arrange nodes in a grid. Rows = failure domains.

	Replica A	Replica B	Replica C
US-East	1	2	3
US-West	4	5	6
EU	7	8	9

L10 said: “The quorum structure should match the failure structure.”

L14 said: “Asymmetric quorums are exactly what geo-replication needs.”

Today: we see what Google actually built.

WAN Latency Is the Dominant Cost

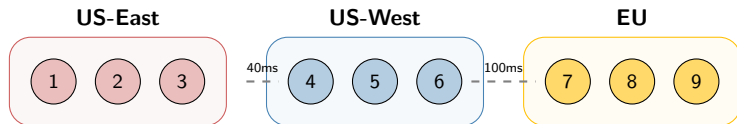
Hop	Latency	Analogy
Same rack	< 1 ms	Free
Same DC	~ 1 ms	Free
Cross-coast	~ 40 ms	Expensive
Cross-ocean	~ 100 ms	Very expensive

Local operations cost microseconds. WAN round-trips cost milliseconds.

A majority quorum across 3 DCs requires at least **1 cross-DC round-trip per operation**.

Quorum design = latency engineering.

Quorum Design for Geography



Local read (EU): Read quorum = {7, 8, 9}. Zero WAN cost.

Durable write: Write quorum = one node per DC: {1, 4, 7}. Two WAN hops.

Intersection: $\{7, 8, 9\} \cap \{1, 4, 7\} = \{7\} \neq \emptyset \checkmark$

Two Strategies

Strategy 1: Local reads, durable writes.

- Read quorum = all nodes in one DC (fast, no WAN)
- Write quorum = one node per DC (slow, all WAN links)
- Intersection: every DC appears in both → guaranteed

Strategy 2: Local writes, global reads.

- Write quorum = all nodes in one DC (fast, no WAN)
- Read quorum = one node per DC (slow, all WAN links)
- Same intersection guarantee, opposite trade-off

This is L14's Flexible Paxos in production.

The quorum sizes aren't symmetric. The *latency profile* is the design variable.

Checkpoint 2

Articulate:

Your system serves users in EU and US. The workload is 80% reads from EU, 20% writes from US.

How would you configure your quorums?

What happens when the US-East DC goes down?

(Pairs: 3 minutes)

Enter Spanner

Spanner (Google, 2012):

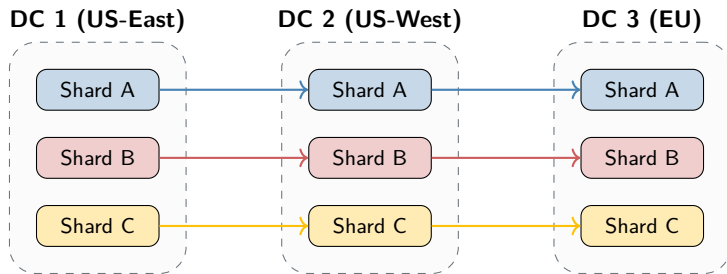
Globally distributed, strongly consistent, SQL.

All three of those properties together were supposed to be impossible (or at least impractical).

- Data sharded across **Paxos groups**
- Each shard replicated across data centers via Multi-Paxos
- Cross-shard transactions via 2PC
- External consistency: real-time ordering of transactions

Spanner doesn't violate CAP. It makes a specific trade-off.

Spanner's Architecture



Each shard is a **Paxos group** replicated across DCs.

Within a shard: Multi-Paxos for ordering.

Across shards: 2PC for transactions (L20).

The Ordering Problem, Revisited

L10 established: quorum gives visibility. Ballots give ordering.

But ballots are *logical* — they order proposals, not real-world events.

External consistency requires:

If transaction T_1 commits before T_2 starts (in real time),
then T_1 's commit timestamp $<$ T_2 's commit timestamp.

Lamport clocks can't do this. They capture causality, not real time.

Spanner's answer: Use actual clocks. But make them trustworthy.

TrueTime

TrueTime API:

`TT.now()` $\rightarrow$ $[earliest, latest]$

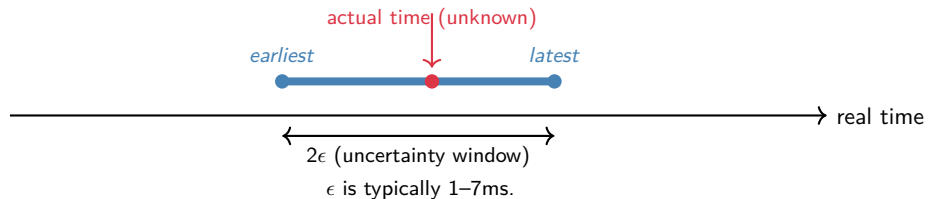
Not a point in time. An **interval**.

Guarantee: The actual time is somewhere in the interval.

- GPS receivers + atomic clocks in every data center
- Two independent time references (GPS and atomic) for fault tolerance
- Uncertainty interval ϵ typically 1–7ms
- The system *knows how uncertain it is*

This is the anti-Lamport: ordering events by physical time, not logical causality.

TrueTime: The Uncertainty Interval



GPS receivers synchronize across DCs.
Atomic clocks bound drift between GPS fixes.

Why Intervals, Not Points?

Ordinary clocks give you a point: “It’s 10:00:00.003.”

But that point might be *wrong* by an unknown amount.

TrueTime gives you an interval: “It’s between 10:00:00.001 and 10:00:00.005.”

The actual time is *guaranteed* to be in that interval.

The key insight: You can’t eliminate uncertainty in physical clocks.

But you can *bound* it. And if you can bound it, you can *wait it out*.

Commit-Wait: Trading Latency for Certainty

The rule:

A transaction must not report “committed” until its timestamp is *certainly in the past*.

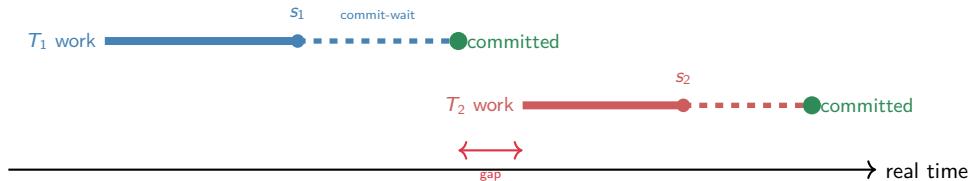
Concretely:

- ① Transaction T acquires locks, does its work
- ② T picks commit timestamp $s = \text{TT.now().latest}$
- ③ T **waits** until $\text{TT.now().earliest} > s$
- ④ Only then: release locks, report committed

The wait is at most 2ϵ (typically 2–14ms).

Spanner trades latency for a real-time ordering guarantee.

Commit-Wait: Why It Works



T_1 commits before T_2 starts (in real time).

Commit-wait ensures $s_1 < s_2$.

External consistency: real-time order = timestamp order.

Why Commit-Wait Is Necessary

Without commit-wait:

- 1 T_1 picks timestamp $s_1 = 100$ at DC-East
- 2 T_1 commits immediately (no wait)
- 3 T_2 starts at DC-West, picks timestamp $s_2 = 99$ (clock skew!)
- 4 T_2 commits

Result: T_1 committed before T_2 in real time, but $s_2 < s_1$.

A reader at timestamp 99 sees T_2 but not T_1 . External consistency violated.

With commit-wait:

T_1 waits until s_1 is certainly in the past. By then, any clock anywhere reads $> s_1$. So T_2 's timestamp must be $> s_1$. Order preserved. ✓

Checkpoint 3

Articulate:

TrueTime costs money (GPS receivers, atomic clocks in every DC).

Commit-wait costs latency (2–14ms per transaction).

What do you get for that cost?

What would you give up if you used Lamport clocks instead?

(Pairs: 3 minutes)

Spanner's CAP Trade-Off

CAP theorem (from L18):

During a network partition, choose **consistency** or **availability**.

Spanner chooses:

- **Consistency:** Always. External consistency via TrueTime.
- **Partition tolerance:** Yes. Paxos groups survive minority DC failures.
- **Availability:** Sacrificed during partitions. A partitioned minority DC cannot serve writes.

And one more thing:

- **Latency:** Sacrificed always. Commit-wait adds delay to every write.

Spanner's answer to CAP: strong consistency is worth the latency cost.

The Anti-Lamport

Lamport clocks (L04):

- $a \rightarrow b \implies C(a) < C(b)$ (causality $\implies$ order)
- $C(a) < C(b) \not\Rightarrow a \rightarrow b$ (order $\not\Rightarrow$ causality)
- No physical time. No real-world ordering guarantee.

TrueTime (Spanner):

- T_1 finishes before T_2 starts $\implies s_1 < s_2$ (real time $\implies$ order)
- Requires physical clocks with bounded uncertainty
- External consistency: timestamp order = real-world order

Lamport said: physical clocks are unreliable, use logical ones.

Google said: make physical clocks reliable enough, then use them.

Both are right. They solve different problems.

In-Class Activity

Setup: 3 data centers, 3 replicas per shard.

US-East, US-West, EU. Latencies: $US-E \leftrightarrow US-W = 40ms$, $US \leftrightarrow EU = 100ms$.

Design a quorum configuration for each goal:

- (a) **Optimize EU read latency.** What is the read quorum? What is the write quorum? What is the read latency for an EU client?
- (b) **Optimize US write throughput.** Minimize the number of cross-DC round-trips per write. What's the trade-off?
- (c) **Survive one DC failure.** Which configurations from (a) and (b) still work? What must you change?

(Groups of 3: 10 minutes. Then present.)

Activity Debrief

(a) EU read latency:

- Read quorum = all EU replicas (0ms WAN). Write quorum = 1 per DC (100ms WAN).
- EU reads are free. Writes pay the cross-ocean cost.

(b) US write throughput:

- Write quorum = US-East + US-West (40ms WAN). Read quorum = all 3 DCs.
- Writes are fast (40ms). Reads pay 100ms to reach EU.

(c) DC failure:

- (a) EU down: reads fail (quorum lost). Writes survive (2 of 3 DCs). **EU reads break.**
- (b) US-West down: writes need US-East + EU (100ms). Still works, but slower.
- **Key:** Any single-DC read quorum breaks when that DC fails.

Fast local reads and DC failure tolerance are in tension. Spanner navigates this per-shard.

What Spanner Actually Does

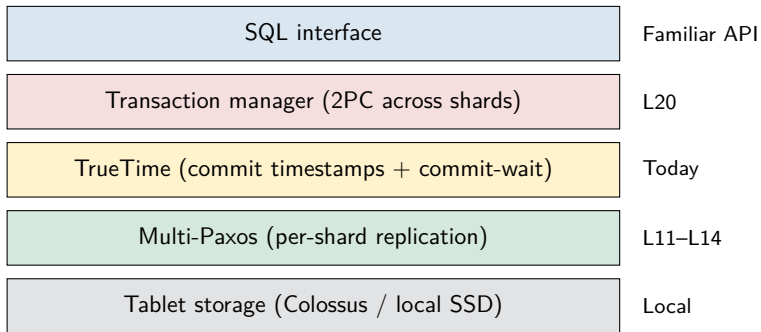
- Each shard has a **Paxos leader** (usually in the DC closest to most clients)
- Leader handles writes via Phase 2 (Multi-Paxos, stable leader)
- **Leader leases:** Leader can serve *reads at a timestamp* without a quorum round-trip
- **Snapshot reads:** Read at a past timestamp, served from any replica with sufficiently recent data

Cross-shard transactions:

- 2PC across Paxos groups (each participant is itself replicated)
- TrueTime provides the commit timestamp
- Commit-wait ensures external consistency

L20 will look at the 2PC layer in detail.

The Stack



Evidence Question

A Spanner deployment reports that commit-wait times have increased from 7ms to 50ms.

What went wrong? Where would you look?

(Hint: What determines the size of the uncertainty interval?)

(Consider: GPS receiver failure, atomic clock drift, network partition to time servers.)

What We Learned

- **Quorum structure should match failure structure** — L10's grid is the template for geo-replication
- **WAN latency is the dominant cost** — quorum design is latency engineering
- **L14's Flexible Paxos** enables asymmetric quorums tuned for geography
- **TrueTime** bounds clock uncertainty with GPS + atomic clocks
- **Commit-wait** trades latency for external consistency
- **Spanner's CAP answer:** strong consistency, sacrifice latency (not availability during normal operation)

The thread:

L10 (quorums) → L14 (asymmetric quorums) → L19 (quorums meet geography + time).

Next: Spanner uses 2PC across shards. What does that cost?

- **Reading for L20:** Spanner (Corbett et al., 2012) — focus on Section 4 (transactions)
- **Also read:** Spanner sections 3 (TrueTime) if you haven't yet
- **Next lecture (L20):** Distributed Transactions — 2PC revisited, now with Paxos participants
- Self-assessment opens 13:00, due next class 11:30 (Pacific)
- Capstone checkpoint details on Canvas