

CPSC 416 – Distributed Systems

Lecture 14: Flexible Paxos

UBC Vancouver

2026-02-26

From Last Time

Turn to your neighbor. 2 minutes.

Multi-Paxos skips Phase 1 for subsequent slots.

Why is that safe? What property still holds?

(Pairs: 2 minutes)

Checkpoint 1

Phase 1 establishes *who gets to propose*.

Multi-Paxos reuses that promise across slots.

But what about the quorums themselves?

L10 told us: a quorum is any collection where every pair intersects.

L11–L13 used *majority* quorums.

Were we forced to?

State Machine Replication

Goal: All replicas apply the same commands in the same order.

The *log* is the fundamental abstraction.

Each slot in the log is one Paxos instance.

Agreeing on a sequence = agreeing on each entry independently.

Who Is the Learner?

What does “chosen” mean?

A value is chosen when accepted by a *quorum of acceptors*.

Who are the acceptors? The replicas. The database nodes.

They already know what they accepted. *They are the learners.*

The client doesn't hear from the quorum. It asks one replica.

That replica must ensure consistency before answering:

- 1 Check for in-flight proposals that could change the value
- 2 Serialize the read against pending writes
- 3 Only then return the result

The learner is the database, not the client.

The Log

The log as the fundamental abstraction:

- **Writes:** Propose a value for the next slot (via Phase 2)
- **Reads:** Find the current value, ensuring no in-flight conflicts
- **Snapshots:** Capture state at a point, truncate log before it

The log grows. Snapshots let you forget.

(The idea of compaction will come back later in the course.)

Checkpoint 5

You're building a replicated key-value store using Multi-Paxos.

A client reads key K . There's a Phase 2 in progress for a write to K .

What does the learner do?

(Pairs: 2 minutes)

What We Learned

- Phase 1 is leader election
- **Multi-Paxos** = leader election once + repeated Phase 2
- The **learner is the database**, not the client
- Consistent reads require serialization against in-flight proposals
- The **log** is the fundamental abstraction

Paxos Made Live: The Gap

The paper you read (Paxos Made Live, Google 2007) describes the gap between the algorithm and production.

Key gaps:

- Disk corruption
- Leader leases
- Group membership changes
- Snapshots

Understanding single-decree Paxos is necessary but not sufficient.

L14 returns to first principles: *the majority was never the point.*

Evidence Question

You're operating a Multi-Paxos system. A client reports **stale reads**.

Where in the system would you look? What would you check?

(Hint: think about the learner's serialization.)

Failure-First Question

We've been using majority quorums.

You have 5 nodes: 2 in Oregon, 3 in Virginia.

Is majority still the right choice?

What if Oregon goes down? What if Virginia does?

Majority treats all nodes equally. Your data centers aren't equal.

Recall: What Paxos Actually Requires

From L10, the quorum intersection property:

$$\forall Q_1, Q_2 \in \mathcal{Q} : Q_1 \cap Q_2 \neq \emptyset$$

From L11–L12, Paxos uses quorums in two places:

- **Phase 1 (Prepare)**: Contact a quorum to claim a ballot
- **Phase 2 (Accept)**: Contact a quorum to commit a value

Question: Do these have to be the *same* quorum system?

The Key Insight (Howard, Malkhi, Spiegelman 2016)

Phase 1 quorums don't need to intersect with each other.

Phase 2 quorums don't need to intersect with each other.

The only requirement:

Every Phase 1 quorum must intersect every Phase 2 quorum.

$$\forall Q_1 \in \mathcal{Q}_{\text{Phase1}}, Q_2 \in \mathcal{Q}_{\text{Phase2}} : Q_1 \cap Q_2 \neq \emptyset$$

Why This Works

What does Phase 1 do?

- Discovers any value that might have been chosen
- A value is chosen when accepted by a **Phase 2 quorum**

So Phase 1 must contact at least one node from every possible Phase 2 quorum.

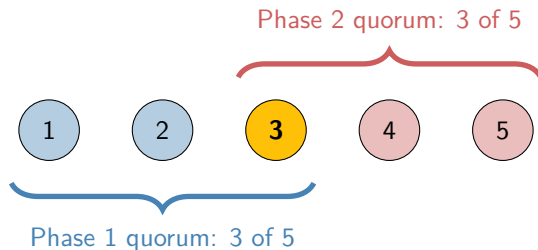
That's the intersection requirement: $Q_1 \cap Q_2 \neq \emptyset$.

What Phase 1 does NOT need:

- To overlap with other Phase 1 quorums (Phase 1 doesn't commit anything)
- Phase 2 quorums don't need to overlap with each other (ballots provide ordering)

The majority was never the point. Intersection was.

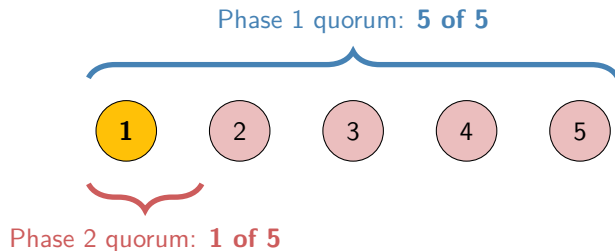
Traditional Paxos: Symmetric Majority



Both phases use majority ($\lceil (n+1)/2 \rceil = 3$).
Intersection guaranteed by pigeonhole.

This is what we've been doing. It works. But is it optimal?

Flexible Paxos: Asymmetric Quorums



Phase 1 = all nodes. Phase 2 = any single node.

Intersection? $5 \cap 1 \geq 1$. ✓

Trade-off: Writes are fast (1 node). Recovery is slow (all nodes).

The Design Space Opens

With $n = 5$ nodes, let $|Q_1|$ = Phase 1 size, $|Q_2|$ = Phase 2 size.

Intersection requires: $|Q_1| + |Q_2| > n$

$ Q_1 $	$ Q_2 $	Tolerate	Profile
3	3	2 crashes	Traditional (balanced)
5	1	0 crashes (P1), 4 crashes (P2)	Write-optimized
1	5	4 crashes (P1), 0 crashes (P2)	Read-optimized*
4	2	1 crash (P1), 3 crashes (P2)	Write-leaning
2	4	3 crashes (P1), 1 crash (P2)	Recovery-leaning

*“Read-optimized” here means Phase 1 (recovery/leader election) is cheap.

Same safety guarantee. Different engineering trade-offs.

Checkpoint 2

In Multi-Paxos, the stable leader skips Phase 1 for normal operations.

Which phase matters most for steady-state performance?

Which matters most for recovery after a leader crash?

What does this tell you about how to size your quorums?

(Pairs: 2 minutes)

The Multi-Paxos Sweet Spot

In Multi-Paxos with a stable leader:

- Phase 1 runs **rarely** (only on leader change)
- Phase 2 runs **constantly** (every write)

Flexible Paxos says: Make Phase 2 quorums small. Pay the cost in Phase 1.

Small Phase 2 = fast writes, low latency

Large Phase 1 = slow recovery, but recovery is rare

Optimize for the common case. Pay for the uncommon case.

Quorums Meet Geography

Recall L10's grid:

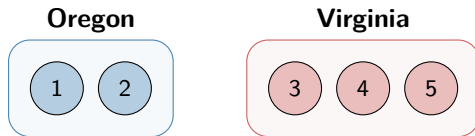
	Replica A	Replica B	Replica C
Oregon	1	2	3
Virginia	4	5	6

6 nodes across 2 data centers. Majority = 4.

Problem: If Oregon (2 nodes) goes down, majority still works. If Virginia (3 nodes) goes down, majority fails.

Majority doesn't know about your failure domains.

Asymmetric Quorums for Asymmetric Infrastructure



Write-optimized

Phase 2 quorum: 1 Oregon + 1 Virginia = 2 nodes

Phase 1 quorum: All Oregon + 2 Virginia = 4 nodes

Intersection: $4 \cap 2 \geq 1 \checkmark$

Read-optimized

Phase 2 quorum: All Oregon + 2 Virginia = 4 nodes

Phase 1 quorum: 1 Oregon + 1 Virginia = 2 nodes

Intersection: $2 \cap 4 \geq 1 \checkmark$

Grid Quorums Revisited

From L10: a quorum = any row + any column.

With Flexible Paxos, we can go further:

- **Phase 1 quorum:** A full row + a full column (“L-shaped”)
- **Phase 2 quorum:** One node per row (“column-like”)

Phase 1 (L-shaped) always hits every row.

Phase 2 (column-like) has one node in every row.

Intersection: the Phase 1 row contains a node from Phase 2's column. ✓

The grid structure from L10 naturally supports asymmetric quorums.

Checkpoint 3

You're designing a system deployed across 3 data centers.

Your workload is 90% reads, 10% writes.

Using Flexible Paxos, how would you configure your quorums?
What are you sacrificing?

(Pairs: 3 minutes)

What Flexible Paxos Changes

Before Howard (2016):

- Everyone used majority quorums
- “Paxos requires a majority” was repeated in every textbook
- Quorum sizes were fixed at $\lceil (n + 1)/2 \rceil$

After Howard (2016):

- Phase 1 and Phase 2 quorums are independent design choices
- The constraint is cross-phase intersection, not within-phase majority
- Quorum design becomes a tuning knob for workload and geography

The result was always implicit in the proof. Nobody noticed for 25 years.

Implications for Real Systems

Geo-replication:

- Writes touch fewer nodes → lower cross-DC latency
- Recovery contacts more nodes → but recovery is rare

Read-heavy workloads:

- Small Phase 2 quorum means fewer nodes to contact per write
- If reads go through the leader, write latency dominates

Write-heavy workloads:

- Large Phase 2 quorum tolerates more failures during writes
- Small Phase 1 quorum means fast leader election

We'll see this applied to Spanner's geo-replication in L19.

In-Class Activity

Setup: 5 nodes across 2 data centers.

Oregon: nodes $\{1, 2\}$ Virginia: nodes $\{3, 4, 5\}$

Design quorum configurations for:

- (a) **Write-optimized:** Minimize Phase 2 quorum size
- (b) **Read-optimized:** Minimize Phase 1 quorum size
- (c) **Balanced:** Survive one full DC failure in both phases

For each: what breaks if Oregon goes down? What breaks if Virginia goes down?

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

Activity Debrief

(a) Write-optimized: Phase 2 = 2 nodes (1 per DC). Phase 1 = 4 nodes.

- Oregon down: Phase 2 still works (1 Virginia node). Phase 1 fails.
- Virginia down: Phase 2 fails (need 1 Virginia node). Phase 1 fails.

(b) Read-optimized: Phase 1 = 2 nodes. Phase 2 = 4 nodes.

- Oregon down: Phase 1 might work (if both from Virginia). Phase 2 needs 4 of 3 — fails.
- Virginia down: Phase 1 from Oregon. Phase 2 needs 4 of 2 — fails.

(c) Balanced: Phase 1 = 3 (majority). Phase 2 = 3 (majority).

- Oregon down: 3 Virginia nodes remain. Both phases work. ✓
- Virginia down: 2 Oregon nodes remain. Neither phase reaches 3. **Fails.**

Asymmetric infrastructure exposes the limits of symmetric quorums.

The Safety Argument

Why is Flexible Paxos safe?

The Paxos safety proof relies on one property:

“If a value v was chosen (accepted by a Phase 2 quorum), then any subsequent Phase 1 will discover v .”

This requires: every Phase 1 quorum intersects every Phase 2 quorum.

It does **not** require:

- Phase 1 quorums to intersect each other
- Phase 2 quorums to intersect each other
- Any particular quorum size

The proof was always more general than the instantiation.

Evidence Question

You deploy Flexible Paxos with Phase 2 quorum = 1 node.

A leader crash triggers Phase 1 recovery.

The new leader's Phase 1 quorum contacts all 5 nodes — but 2 are slow to respond.

Is the system safe? Is it live?

What would you monitor to detect this before it causes an outage?

What We Learned

- **Majority was never the point** — quorum *intersection* was
- Phase 1 and Phase 2 quorums can differ in size
- The only constraint: $Q_1 \cap Q_2 \neq \emptyset$ across phases
- This opens a design space: optimize for workload and geography
- The grid quorum model from L10 naturally supports asymmetric quorums

The lattice connection:

L10 gave you quorum intersection as the foundation.

L11–L13 walked the Paxos path with majority quorums.

L14 shows: the foundation was always more general than the path we walked.

Next: what if we don't need voting at all?

- **Reading for L15:** Chain Replication (van Renesse & Schneider, 2004)
- **Optional:** CRAQ (Terrace & Freedman, 2009)
- **Next lecture (L15):** Chain Replication — structure replaces quorums
- Self-assessment opens 13:00, due next class 11:30 (Pacific)
- Capstone checkpoint details on Canvas