Flexible Paxos: Quorum Intersection Revisited

Heidi Howard Dahlia Malkhi Alexander Spiegelman

Presenter: Yatian Liu

November 1, 2021

Outline

1. Background and Motivation
   - Review of the Paxos Algorithm
   - Quorum Requirement of Paxos
2. Detailed Analysis
   - Weakened Quorum Requirement
   - Alternative Quorum Systems
3. Evaluation

Outline

1. Background and Motivation
   - Review of the Paxos Algorithm
   - Quorum Requirement of Paxos
2. Detailed Analysis
   - Weakened Quorum Requirement
   - Alternative Quorum Systems
3. Evaluation

Review on Paxos

- Two phases to decide a value that cannot be changed
- Phase 1: sends *Prepare(p)* to all and waits for $f + 1$ *Promise(p', v')*
- Phase 2: sends *Propose(p, v)* to all and waits for $f + 1$ *Accept(p)*
- More generally, Paxos requires a majority quorum for both phase 1 and phase 2
  - $\lfloor n/2 \rfloor + 1$, also applies to even number of replicas

Drawbacks of Paxos

- Paxos requires a majority quorum for both phase 1 and phase 2 for intersection
  - High network traffic pressure for large systems
  - Limits throughput and increases latency

Drawbacks of Paxos

- Paxos requires a majority quorum for both phase 1 and phase 2 for intersection
  - High network traffic pressure for large systems
  - Limits throughput and increases latency
- In fact, only intersection between quorums in phase 1 and quorums in phase 2 is needed!
  - Intersection between quorums *in the same phase* is not needed
  - The quorum requirement can be weakened to get lower latency and higher throughput

Drawbacks of Paxos

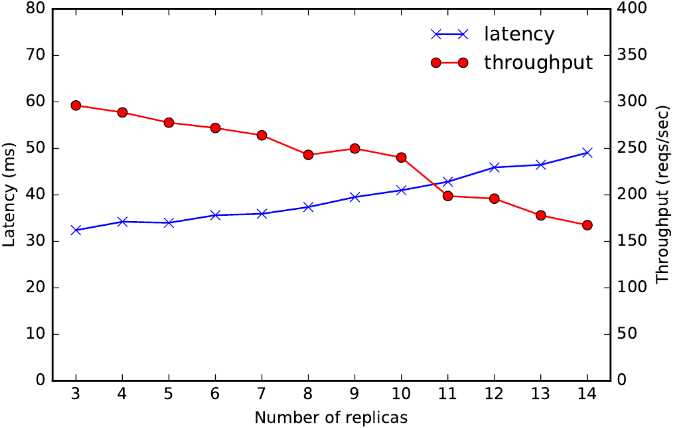


Figure 1: Performance of `LibPaxos3`

Basic Idea

- For Multi-Paxos, phase 1 only need to be executed once if primary does not fail
- Can reduce work for phase 2 at the cost of increasing work for phase 1, weakened liveness guarantee for phase 1, ...
- Justification: tolerating $\lfloor n/2 \rfloor$ failures is not always needed for large systems

Outline

Weakened Quorum Requirement

Proposition

Paxos is still safe as long as a quorum system that guarantees intersection between any phase 1 quorum and phase 2 quorum is used.

Formally speaking, a quorum system satisfies this property if:

- $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are the sets of all valid phase 1 and phase 2 quorums respectively
- $\mathcal{A}$ is the set of all acceptors
- $\forall Q_1 \in \mathcal{Q}_1 : Q_1 \subset \mathcal{A}$
- $\forall Q_2 \in \mathcal{Q}_2 : Q_2 \subset \mathcal{A}$
- $\forall Q_1 \in \mathcal{Q}_1, \forall Q_2 \in \mathcal{Q}_2 : Q_1 \cap Q_2 \neq \emptyset$

This kind of modified Paxos algorithms is called Flexible Paxos (FPaxos for short).

Weakened Quorum Requirement: Proof

"FPaxos is safe" $\Leftrightarrow$ "All decisions are final"

Theorem

Given a valid quorum system, if a value v is decided with proposal number p, then for any message Propose(p', v') where $p' > p$, $v' = v$.

Proof

Use proof by contradiction. Suppose there exists messages *Propose(p', v')* where $p' > p$ and $v' \neq v$, and choose the message that has the smallest p'.
$Q_{p,2}$: quorum for p, phase 2 (*Propose(p, v)*).
$Q_{p',1}$: quorum for p', phase 1 (*Prepare(p')*).
From specification of quorum system: $\bar{A} = Q_{p,2} \cap Q_{p',1} \neq \emptyset$.

Weakened Quorum Requirement: Proof

Proof (Continued)

Consider one acceptor $a \in \bar{A}$. From definition of phase 1 and phase 2 quorum a has received and replied to both *Propose(p, v)* and *Prepare(p')*.

- If a received *Prepare(p')* earlier $\Rightarrow$ cannot accept *Propose(p, v)* ↯
- If a received *Propose(p, v)* earlier:
  - a replied to *Prepare(p')* with $Promise(q, v'')$ where $p \leq q < p'$
  - By smallest p' assumption, $v'' = v$
  - For all other $Promise(q', v''')$ received, three cases: $q' < q$, $q \leq q' \leq p'$, $p' < q'$
  - v will be chosen by p' in all cases ↯

□

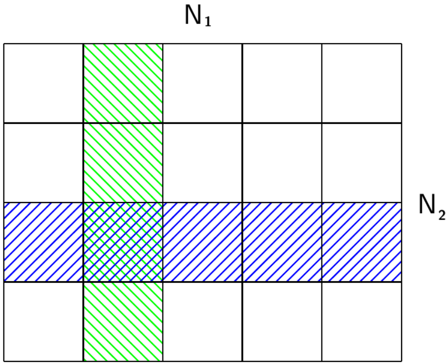
Modified Majority Quorums

- For even number n, original Paxos require size $n/2 + 1$ quorum for both phase 1 and phase 2
- FPaxos: only size $n/2$ quorum required for phase 2
- Slightly reduce latency and improve throughput
- Slightly increase liveness guarantee

Simple Quorums

- To guarantee phase 1 and phase 2 quorums intersect $\Rightarrow |Q_1| + |Q_2| > N$, choose $N + 1$
- Phase 2 more common than phase 1 $\Rightarrow$ choose $|Q_2| < N/2$ and $|Q_1| = N + 1 - |Q_2| > N/2$
- Also, can send fewer messages in phase 2
  - At cost of fault tolerance
- Reduce latency and improve throughput
- Sacrifice liveness guarantee
  - Only guarantee liveness under $N - |Q_1| = |Q_2| - 1$ failures
  - Handle up to $N - |Q_2|$ failures if primary does not fail

Grid Quorums



$$N = N_1 \times N_2$$

$$\mathcal{Q}_1 = \{\text{all the rows of length } N_1\}, \mathcal{Q}_2 = \{\text{all the columns of length } N_2\}$$

Grid Quorums

- Can choose non-majority quorums for both phase 1 and phase 2
- Better latency and throughput
- Worse liveness guarantee
  - Worst case: only tolerate $\min\{N_1, N_2\}$ failures
  - "Which" is more important than "how many"
  - Can possibly recover by reconfiguration

Outline

Implementation & Setup

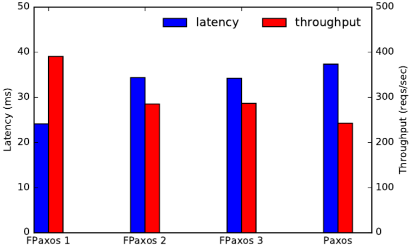
Implementation

- Modifies LibPaxos3
- Use simple quorums with varying $|Q_1|$ and $|Q_2|$
- Choose quorums at random, only send messages to selected nodes

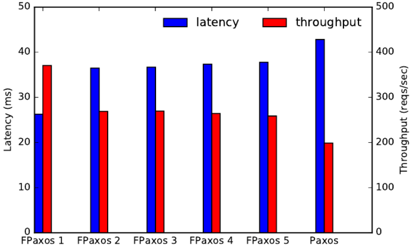
Experimental Setup

- Run on a single Linux VM with single core and 1 GB RAM
- Use Mininet with 10 Mbps bandwidth, 20 ms round trip time
- Run for 120 seconds and discard first and last 10 second data

Experiment Results



(a) Performance of FPaxos and LibPaxos3 with 5 replicas.



(b) Performance of FPaxos and LibPaxos3 with 8 replicas.

Figure 2: Performance comparison of Paxos and FPaxos. Numbers refer to $|Q_2|$ in simple quorums.

Conclusion

- Quorum requirement of Paxos can be weakened
- Alternative quorum systems can improve latency and throughput at the cost of liveness guarantee
- Allow more choices for performance tradeoff

Thanks!

Discussion