XFT: Practical Fault Tolerance beyond Crashes

Presented by Clay Connors
Introduction

Byzantine fault tolerance (BFT) is too strong.

Crash fault tolerance (CFT) is too weak.

This paper introduces:

- Cross fault tolerance (XFT), an alternate form of BFT
- XPaxos, an XFT protocol for state-machine replication (SMR)
Problems with CFT

CFT does not give resilience to:

- Malicious behavior
- Hardware errors
- Corrupted data from storage
- etc.

Large companies (Google, Amazon) occasionally face failures which could have been prevented by BFT.
Problems with BFT

Extra cost compared to CFT ($3t + 1$ instead of $2t + 1$).

BFT assumes that an arbitrary number of correct machines can be partitioned.
More practical than BFT:

- No extra resource cost (2t+1 replicas)
- Provides CFT
- Correct when minority (up to t) replicas are Byzantine-faulty or partitioned

Good for geo-replicated systems where network partitions are uncommon.

Also good for blockchains where actors are malicious but uncoordinated.
XFT

vs CFT:

- Always stronger reliability
- Always stronger consistency

vs BFT:

- Always stronger reliability
- Sometimes stronger consistency
Safety of XFT

A replica is partitioned when it is not in the largest set of synchronous replicas.

The system is in anarchy when:

- There is at least one non-crash fault, and
- The sum of crash faults, non-crash faults and partitioned replicas is > t

XFT protocols are safe if the system has never entered anarchy.
Liveness depends on the implementation.
XPaxos

Three main components:

- Common-case — replicates and orders client requests
- View-change — transfers information between views
- Fault-detection — handle non-crash faults to avoid anarchy
Common-case

Each view has a set of active replicas (primary and t followers).

If all active replicas are correct:

- Client sends signed request to primary
- Primary sends signed PREPARE to followers
- Followers send signed COMMIT to active replicas
- Active replicas send authenticated REPLY to client
Common-case

Client

Primary

Follower

Follower

Non-Follower

Non-Follower
View-change

A view-change is triggered by:

- Active replica receives nonconformant message
- Client indicates a time out, and a retransmission attempt also times out
- Current view-change times out
- Active replica receives signed SUSPECT message from another active replica

Upon starting a view change, a replica sends a SUSPECT message for the previous view.
View-change

All replicas from the old view send VIEW-CHANGE to all active replicas in the new view.

New active replicas then:

- Wait for $t + 1$ VIEW-CHANGE messages, then send them all as VC-FINAL.
- Wait for all VC-FINAL messages, then combine the commit logs.

New primary sends NEW-VIEW to active replicas with all pending messages.
Fault detection

Optional mechanism to reduce likelihood of anarchy.

The described mechanism deals with data loss:

- Replicas exchange prepare logs during a view-change
- PREPARE events from the faulty replica must causally precede COMMIT events from the correct replicas
XPaxos optimizations

**Checkpointing:** Commit proofs are periodically distilled to a single checkpoint proof

**Lazy replication:** Active replicas send commit log to passive replicas

**Batching:** Primary collects multiple requests to send as a single batch

**Pipelining:** Primary sends new ordering requests before the previous ones have finished
XPaxos performance

Compared on geo-distributed system with PBFT, Zyzzyva, and an efficient implementation of Paxos.

Main results:

- XPaxos has nearly identical latency to Paxos at low loads, but lower peak throughput (~80%).
- PBFT and Zyzzyva have double the latency at low loads and much lower peak throughput (~50% of Paxos, ~60% of XPaxos)
XPaxos has an advantage in geo-distributed systems:

- Simpler message pattern than PBFT
- Less replication than Zyzzyva

With $t=1$, XPaxos and Paxos are both just a single round trip.
XPaxos performance

Testing under faults showed that view changes occur in under 10 seconds.

Additionally, both Paxos and XPaxos outperform Zab, the CFT protocol used in ZooKeeper, when used as a replacement.
Consistency of XPaxos

**Lemma 1, paraphrased:** If a benign client commits a request to a slot, that slot will either contain that request or be empty for all future views in all benign replicas, outside of anarchy.

Total order follows from this.
Consistency of XPaxos

Paper contains some detailed results on reliability.

Main points:

- For $t=1$, XPaxos adds nines of reliability over CFT proportional to either $9^{\text{correct}}$ or $9^{\text{synchrony}}$.
- For other cases, XPaxos still has higher reliability than CFT.
- For $t=1$, XPaxos has higher reliability than asynchronous BFT when $p^{\text{correct}} \times p^{\text{synchrony}} > p^{\text{benign}}^{1.5}$. 
Conclusion

Summarizing:

- XFT provides all of CFT and a different non-crash guarantee than BFT
- XFT is strong enough to use in practice but weak enough to avoid overhead
- XPaxos keeps pace with Paxos and significantly outperforms BFT in geo-distributed systems
- XPaxos has higher reliability than CFT and overlapping reliability with BFT