paper: Amazon Aurora: On Avoiding Distributed Consensus for I/Os, Commits, and Membership Changes

by Diego Pacheco

Amazon Aurora On Avoiding Distributed Con... 1 MB

- Aurora High Throughput Cloud-native relational database
- Aurora avoid distributed consensus under most circumstances
- Leveraging local transient state
- Doing so improving performance and reduces variability and lower costs.
- 3/6 Read quorum
- 4/6 Write quorum
- Why aurora need 6 Copies

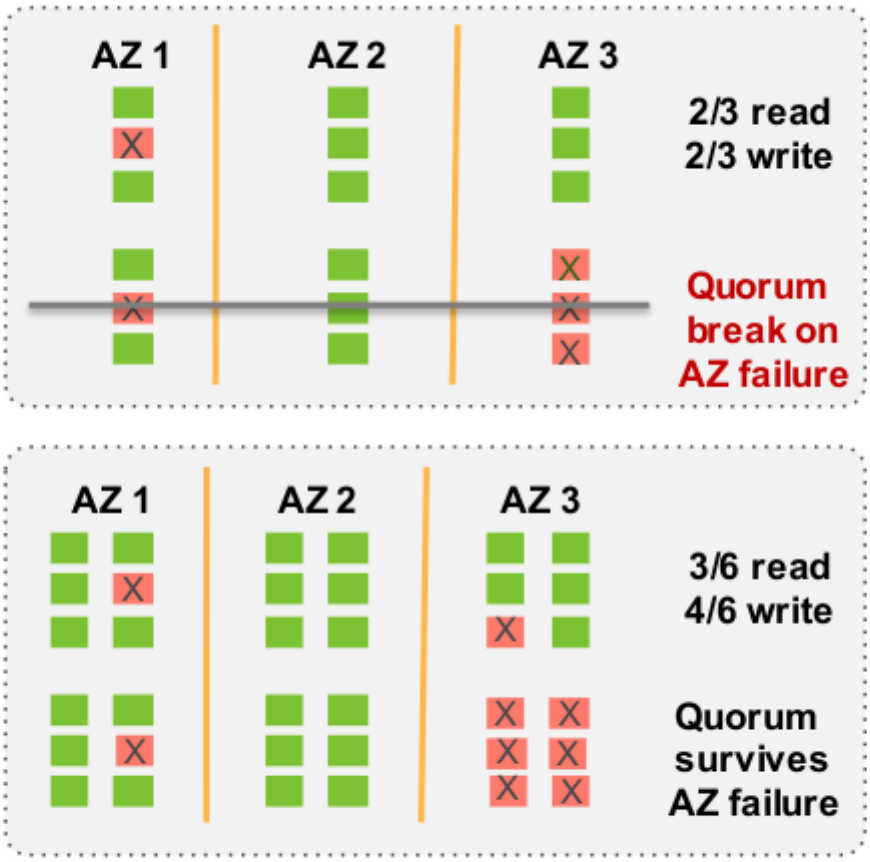

Figure 1: Why are 6 copies necessary ?

- Segments in Aurora are the minimum unit of failure (10GB Each)
- Aurora Storage Consistency Points

Figure 3: Storage Consistency Points

- Storage nodes dont have a vote on accepting writes - they must do so.
- Locking tx management, deadlocks, constraints and others conditions that influence if operation should proceed are all resolved on database tier.
- Failed Storage nodes can be repaired without involving the database instance
- Crash Recovery in Aurora

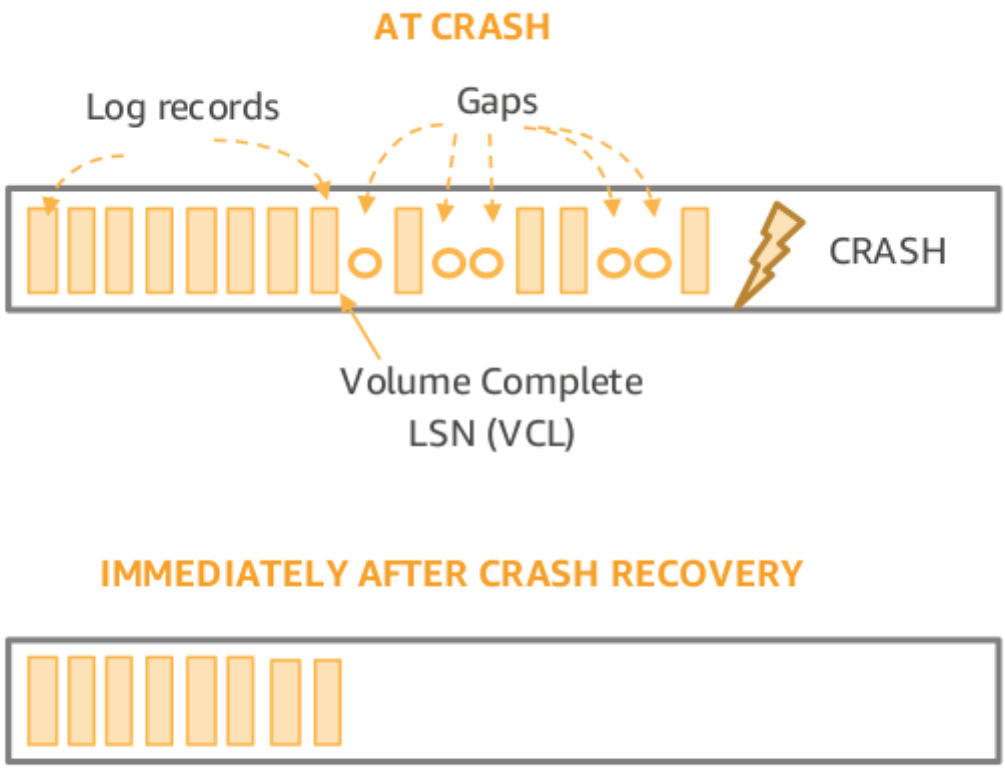

Figure 4: Log truncation during crash recovery

- If Customer shutdown, resize, restore to an old point auror will need to do distributed consensus.
- This trade of is worth since commits are ORDER in magnitude more commons than CRASHES.
- IF Aurora is not able to establish write quorum for one of its protection groups it initiates repair from available read-qurum to rebuild failed segments.
- Making Reads Efficient
- Reads are few operations in aurora where threads need to WAIT.
- Unlike writes that can stream async to storage nodes or commits where the worker can move to other worker while wait.
- Avoiding Quorum Reads
- Aurora use read views to support snapshot isolation using MVCC.
- Aurora does not do quorum reads
- The database instance knows which segments have the last durable version of data block.
- Aurora can request directly to any of those segments avoiding the amplification of read quoruns.
- Which makes aurora subject to latency when storage nodes are down or jitter when they are busy.
- Scaling Reads using read Replicas
- In many Database systems Sync or Async Read replica replication is a problem:
  - Sync replication: Introduce Performance Jitter and failures on write path
  - Async replication: Introduce data loss on failure of the writer.
- Aurora supports logical replication to communicate with non aurora systems.
- In cases where the application does not want physical consistency.
- Durable state is sharad so aurora customers can scale up and down replicas with no issues
- Quorum Sets to Change Membership
- Aurora uses abstraction of Quorum Sets to quickly transition membership changes.

- Using boolean logic we can proof each transition is correct, safe and reversible.
- Tradeoff between SSD for Performance and HDD for COST reduction.