Zab vs. Paxos

Is Zab just a special implementation of Paxos?

No, Zab is a different protocol than Paxos, although it shares with it some key aspects, as for example:

- A leader proposes values to the followers
- Leaders wait for acknowledgements from a quorum of followers before considering a proposal committed (learned)
- Proposals include epoch numbers, which are similar to ballot numbers in Paxos

The main conceptual difference between Zab and Paxos is that it is primarily designed for primary-backup systems, like Zookeeper, rather than for state machine replication.

What is the difference between primary-backup and state machine replication?

A state machine is a software component that processes a sequence of requests. For every processed request, it can modify its internal state and produce a reply. A state machine is deterministic in the sense that, given two runs where it receives the same sequence of requests, it always makes the same internal state transitions and produces the same replies.

A state machine replication system is a client-server system ensuring that each state machine replica executes the same sequence of client requests, even if these requests are submitted concurrently by clients and received in different orders by the replicas. Replicas agree on the execution order of client requests using a consensus algorithm like Paxos. Client requests that are sent concurrently and overlap in time can be executed in any order. If a leader fails, a new leader that executes recovery is free to arbitrarily reorder any uncommitted request since it is not yet completed.

In the case of primary-backup systems, such as Zookeeper, replicas agree on the application order of incremental (delta) state updates, which are generated by a primary replica and sent to its followers. Unlike client requests, state updates must be applied in the exact original generation order of the primary, starting from the original initial state of the primary. If a primary fails, a new primary that executes recovery cannot arbitrarily reorder uncommitted state updates, or apply them starting from a different initial state.

In conclusion, agreement on state updates (for primary-backup systems) requires stricter ordering guarantees than agreement on client requests (for state machine replication systems).

What are the implications for agreement algorithms?

Paxos can be used for primary-backup replication by letting the primary be the leader. The problem with Paxos is that, if a primary concurrently proposes multiple state updates and fails, the new primary may apply uncommitted updates in an incorrect order. An example is presented in our DSN 2011 paper (Figure 1). In the example, a replica should only apply the state update B after applying A. The example shows that, using Paxos, a new primary and its follows may apply B after C, reaching an incorrect state that has not been reached by any of the previous primaries.

A workaround to this problem using Paxos is to sequentially agree on state updates: a primary proposes a state update only after it commits all previous state updates. Since there is at most one uncommitted update at a time, a new primary cannot incorrectly reorder updates. This approach, however, results in poor performance.

Zab does not need this workaround. Zab replicas can concurrently agree on the order of multiple state updates without harming correctness. This is achieved by adding one more synchronization phase during recovery compared to Paxos, and by using a different numbering of instances based on zxids.

Want to know more?

Have a look at our DSN 2011 paper, or contact us!