# Summary: Dynamo

Here is a quick summary of Dynamo for you!

**We'll cover the following**



# Summary#

1. Dynamo is a highly available **key-value store** developed by **Amazon** for their internal use.
2. Dynamo shows how business requirements can drive system designs. Amazon has chosen to sacrifice strong consistency for **higher availability** based on their business requirements.
3. Dynamo was designed with the understanding that system/hardware failures can and do occur.
4. Dynamo is a **peer-to-peer** distributed system, i.e., it does not have any leader or follower nodes. All nodes are equal and have the same set of roles and responsibilities. This also means that there is **no single point of failure**.
5. Dynamo uses the **Consistent Hashing** algorithm to distribute the data among nodes in the cluster automatically.
6. Data is replicated across nodes for fault tolerance and redundancy. Dynamo replicates writes to a **sloppy quorum** of other nodes in the system instead of a strict majority quorum.

7. For anti-entropy and to resolve conflicts, Dynamo uses **Merkle trees**.
8. Different storage engines can be plugged into Dynamo's local storage.
9. Dynamo uses the **gossip protocol** for inter-node communication.
10. Dynamo makes the system "always writeable" by using **hinted handoff**.
11. Dynamo's design philosophy is to ALWAYS allow writes. To support this, Dynamo allows concurrent writes. Writes can be performed by different servers concurrently, resulting in multiple versions of an object. Dynamo attempts to track and reconcile these changes using **vector clocks**. When Dynamo cannot reconcile an object's state from its vector clocks, it sends it to the client application for reconciliation (*the thought being that the clients have more semantic information on the object and may be able to reconcile it*).
12. Dynamo is able to successfully pull together several distributed techniques such as consistent hashing, p2p, gossip, vector clocks, and quorum, and combine them into a complex system.
13. Amazon built Dynamo for internal use only, so **no security** related issues were considered.

The following table presents a summary of the list of techniques Dynamo uses and their respective advantages.

| Problem | Technique | Advant |
|---|---|---|
| Partitioning | Consistent Hashing | Incremental S |
| High availability for writes | Vector clocks with reconciliation during reads | Version size is de update r |
| Handling temporary failures | Sloppy Quorum and Hinted Handoff | Provides high av durability guarantee the replicas are |
| Recovering from permanent failures | Anti-entropy using Merkle trees | Synchronizes diverg the backg |
| Membership and failure detection | Gossip-based membership protocol and failure detection | Preserves symme centralized m |

# System design patterns#

Here is a summary of system design patterns used in Dynamo:

- **Consistent Hashing:** Dynamo uses Consistent Hashing to distribute its data across nodes.
- **Quorum:** To ensure data consistency, each Dynamo write operation can be configured to be successful only if the data has been written to at least a quorum of replica nodes.
- **Gossip protocol:** Dynamo uses gossip protocol that allows each node to keep track of state information about the other nodes in the cluster.
- **Hinted Handoff:** Dynamo nodes use Hinted Handoff to remember the write operation for failing nodes.
- **Read Repair:** Dynamo uses ‘Read Repair’ to push the latest version of the data to nodes with the older versions.
- **Vector clocks:** To reconcile concurrent updates on an object Dynamo uses Vector clocks.
- **Merkle trees:** For anti-entropy and to resolve conflicts in the background, Dynamo uses Merkle trees.

# References and further reading#

- Amazon’s Dynamo (https://www.allthingsdistributed.com/2007/10/amazons_dynamo.html)

- Eventually Consistent (https://www.allthingsdistributed.com/2007/12/eventually_consistent.html)
- Bigtable (https://research.google/pubs/pub27898/)
- DynamoDB (https://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html)
- CRDT (https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type)
- A Decade of Dynamo (https://www.allthingsdistributed.com/2017/10/a-decade-of-dynamo.html)
- Riak (https://docs.riak.com/riak/kv/2.2.0/learn/dynamo/)
- Dynamo Architecture (https://www.youtube.com/watch?v=w96lLsbI1q8)
- Dynamo: A flawed architecture (https://news.ycombinator.com/item?id=915212)

← Back

Next →

Dynamo Characteristics and Criticism

Quiz: Dynamo

Mark as Completed

Report an Issue