Dynamo Characteristics and Criticism

Let's explore the characteristics and the criticism of Dynamo's architecture.

We'll cover the following

- Responsibilities of a Dynamo’s node
- Characteristics of Dynamo
- Criticism on Dynamo
- Datastores developed on the principles of Dynamo

Responsibilities of a Dynamo’s node#

Because Dynamo is completely decentralized and does not rely on a central/leader server (*unlike GFS, for example)*, each node serves three functions:

1. **Managing `get()` and `put()` requests:** A node may act as a coordinator and manage all operations for a particular key or may forward the request to the appropriate node.
2. **Keeping track of membership and detecting failures:** Every node uses gossip protocol to keep track of other nodes in the system and their associated hash ranges.
3. **Local persistent storage:** Each node is responsible for being either the primary or replica store for keys that hash to a specific range of

values. These (key, value) pairs are stored within that node using various storage systems depending on application needs. A few examples of such storage systems are:

- BerkeleyDB Transactional Data Store
- MySQL (for large objects)
- An in-memory buffer (for best performance) backed by persistent storage

Characteristics of Dynamo#

Here are a few reasons behind Dynamo's popularity:

- **Distributed:** Dynamo can run on a large number of machines.
- **Decentralized:** Dynamo is decentralized; there is no need for any central coordinator to oversee operations. All nodes are identical and can perform all functions of Dynamo.
- **Scalable:** By adding more nodes to the cluster, Dynamo can easily be scaled horizontally. No manual intervention or rebalancing is required. Additionally, Dynamo achieves linear scalability and proven fault-tolerance on commodity hardware.
- **Highly Available:** Dynamo is fault-tolerant, and the data remains available even if one or several nodes or data centers go down.
- **Fault-tolerant and reliable:** Since data is replicated to multiple nodes, fault-tolerance is pretty high.
- **Tunable consistency:** With Dynamo, applications can adjust the trade-off between availability and consistency of data, typically by configuring replication factor and consistency level settings.
- **Durable:** Dynamo stores data permanently.
- **Eventually Consistent:** Dynamo accepts the trade-off of strong consistency in favor of high availability.

Criticism on Dynamo#

Criticism on Dynamo#

The following list contains criticism on Dynamo's design:

- Each Dynamo node contains the entire Dynamo routing table. This is likely to affect the scalability of the system as this routing table will grow larger and larger as nodes are added to the system.
- Dynamo seems to imply that it strives for symmetry, where every node in the system has the same set of roles and responsibilities, but later, it specifies some nodes as seeds. Seeds are special nodes that are externally discoverable. These are used to help prevent logical partitions in the Dynamo ring. This seems like it may violate Dynamo's symmetry principle.
- Although security was not a concern as Dynamo was built for internal use only, **DHTs can be susceptible to several different types of attacks**. While Amazon can assume a trusted environment, sometimes a buggy software can act in a manner quite similar to a malicious actor.
- Dynamo's design can be described as a "**leaky abstraction**," where client applications are often asked to manage inconsistency, and the user experience is not 100% seamless. For example, inconsistencies in the shopping cart items may lead users to think that the website is buggy or unreliable.

Datastores developed on the principles of Dynamo#

Dynamo is not open-source and was built for services running within Amazon. Two of the most famous datastores built on the principles of Dynamo are Riak (https://riak.com/) and Cassandra (https://cassandra.apache.org/). Riak is a distributed NoSQL key-value data store that is highly available, scalable, fault-tolerant, and easy to operate. Cassandra is a distributed, decentralized, scalable, and highly available

NoSQL wide-column database. Here is how they adopted different algorithms offered by Dynamo:

Technique	Appache Cassandra	Riak
Consistent hashing with virtual nodes	Yes	Yes
Hinted Handoff	Yes	Yes
Anti-entropy with Merkle tress	Yes (manual repair)	Yes
Vector Clocks	No (last-write-wins)	Yes
Gossip-based protocol	Yes	Yes

← Back

Next →

Gossip Protocol

Summary: Dynamo

Mark as Completed

Report an Issue