Paper Review –4

Dynamo

Title: Dynamo: Amazon's Highly Available Key-value store

Summary

The paper presents the design and implementation of Dynamo which is amazon's highly available key-value storage system that is used internally by many of Amazon's core services. The main goal of Dynamo is to create a model that is available all the time thereby sacrificing on consistency of data in case of failure's. It uses advanced algorithms to provide consistency and versioning of objects. It provides a highly decentralized loosely coupled, service oriented architecture. One use case where it is used is in the Amazon's shopping cart where the user's shopping cart must be available all the time.

Good Aspects

Amazon Dynamo has addressed an issue by building a feature specific product that targets availability and high speed fetching/updating. It also provides a very simple interface that uses primary keys to clients. One of the big plus of Dynamo is that uses an modified version of Consistent hashing that is used in Cassandra. The consistency model instead of conventional systems uses a technique called quorum that helps Dynamo achieve its SLA (Service level agreement). Dynamo is an example of research done on various techniques to bring together a highly available system. Dynamo is currently used in production by many of Amazons real time services.

Review

Following details will cover all aspect of the paper providing crucial and also interesting aspects of the paper that appealed the reader.

Amazon Dynamo is a highly available key-value storage system that some of Amazon's services use to provide an "always-on" experience.

Requirements

Query Model – A simple model that uses a primary key for read and write operations. Targets applications that have object storage needs of less than 1MB.

Acid Properties – (Atomicity, Consistency, Isolation and Durability) is a set of properties that are guaranteed by Traditional Database systems. Dynamo targets only application with weaker C in ACID. Also dynamo doesn't guarantee any isolation and permits only single key updates.

Efficiency – In Amazon's platform, service have latency requirements measured in 99.9^{th} percentile of distribution. These services must be able to configure dynamo such that they constantly achieve their latency and throughput requirements.

SLA (Service Level Agreements) - Clients and services engage in a service level agreement (SLA), a formally negotiated contract where a client and a service agree on several systems related characteristic. A common approach in industry for forming a performance oriented SLA is to describe it using average, median and expected variance. However, this is not good enough to please all customers in Amazon which addresses a majority. Amazon uses a SLA expressed and measured at the 99.9th percentile of the distribution. One of the main design considerations for Dynamo is to give services control over their system properties, such as durability and consistency, and to let services make their own tradeoffs between functionality, performance and cost-effectiveness.

Design Considerations –

- For systems prone to server and network failures, availability can be increased by using optimistic replication techniques, where changes are allowed to propagate to replicas in the background, and concurrent, disconnected work is tolerated. Such a system must handle conflict in changes. There are 2 aspects that it has to look into when to resolve and who resolves.
- Incremental Scalability – Should be able to scale out one node at a time with minimal impact.
- Symmetry – Every node should have same set of responsibilities
- Decentralization – Centralized control has dependency on a master and can result in outages if master dies. Hence the design is to favor decentralized peer to peer.
- Heterogeneity – It must be proportional to the capabilities of the individual servers.

Key value based storage

A key value store is more suitable in this case

- It is intended to store relatively small objects.
- Key value stores are easier to configure on a per app basis.

Difference from other decentralized storage systems

- Dynamo targeted mainly at applications that need an always write able data store where no updates are rejected due to failures.
- Built for an infrastructure within a single admin domain where all nodes are assumed to be trusted.
- Applications that use dynamo don't require support for hierarchical name spaces or complex relational database.
- Dynamo has a SLA to serve all applications with 99.9% of guaranteed Read write operations to be performed in a few hundred milliseconds.
- dynamo is a Zero-hop DHT where each node maintains enough routing information locally to route a request to the appropriate node directly.

Various Techniques in Dynamo

Problem	Technique
Partitioning	Consistent Hashing
High availability for writes	Vector Clocks
Handling temporary failures	Sloppy Quorum and hinted handoff
Permanent Failure Recovery	Anti-entropy using Merkle Trees
Membership and failure detection	Gossip Protocol with failure detect

Technical Details

- **Partitioning Algorithm**
  - Dynamic Partitioning the data over the set of nodes using a hash function whose output range is considered as a ring. The largest hash value wraps around the smallest hash value.
  - Consistent hashing algorithm is used but the basic algorithm presents some challenges-
    - The random assignment of each node on the ring leads to non uniform data and load distribution.
    - The basic algorithm is oblivious to the heterogeneity in the performance of nodes.
  - Amazon uses a fine tuned consistent hashing algorithm to map a node to different nodes to multiple points in the ring. Uses concept of Virtual Node where each node is responsible for many Virtual Nodes.
  - Advantages –
    - If a node becomes unavailable, the load handled by this node is evenly dispersed across the remaining available nodes.
    - When a node becomes available again, the newly available node accepts the roughly equivalent amount of load.
    - The number o virtual nodes required can be dynamic since the system is heterogeneous.

- **Data Replication**
  - Replication at N Nodes can be configured per instance.
  - Each key 'k' is assigned a coordinator node that is responsible for replicating to N-1 nodes in a clockwise manner. The coordinator also stores the key locally.
  - This way each node is responsible for the data between it and its Nth predecessor.
  - System itself can determine the authoritative version via Syntactic reconciliation however certain conflict versions must be handled by Client.
  - Different versions of Same object handling uses Vector clocks which helps to determine which is the latest copy and if two changes are required and require reconciliation.
  - Context stores information about the vector clocks.

- **Execution of get() and put() operations**
  - Both get and put operations are invoked using Amazon's infrastructure-specific request processing framework over HTTP.
  - The two strategies to select a node are –
    - Route its request through a generic load balancer
    - Use a partition aware client library that routes requests directly to the appropriate coordinator.
  - A node handling read or write operation is called a Coordinator. This is the first of the best N nodes.
  - To maintain consistency amongst its replicas, Dynamo uses a consistency protocol similar to those used in quorum systems.
  - This protocol has 2 configurable values R and W.
  - R is the minimum number of nodes that must participate in a successful read operation.
  - W is the minimum number of nodes that must participate in a successful write operation.
  - Put request –
    - The coordinator generates a vector clock for the new version and writes the new version locally.
    - The coordinator then sends the new version to the N highest ranked reachable nodes.
    - If at least W-1 nodes respond, then the write is considered successful.
  - Get Request –
    - The coordinator requests all existing versions of data for that key from the N highest-ranked reachable nodes in the preference list and then waits for R responses.

- **Handling Failures**
  - **Hinted Handoff**
    - A modified quorum approach called sloppy quorum where all read and write operations are performed on the first N healthy nodes from the preference list.
    - Nodes that receive hinted replicas will keep them in a separate local data base and scan it periodically. Upon detecting that the server has recovered it will attempt to deliver to the node and delete its contents from local store to keep a control on replicas.
    - Hence hinted handoff ensures that read and write operations are not failed due to temporary node or network failures.

- **Handling Permanent Failures**
  - Hinted handoff works best only if the system membership is low and node failures are transient.
  - To handle permanent failures, amazon uses Anti-entropy or replica synchronization to keep the replicas synchronized.
  - If hash of root is same, then the child nodes will be the same since the hash of parent is calculated using the hash of the children.
  - This helps to save amount of data that is transferred to check for inconsistencies.
  - Dynamo uses Anti- Entropy to detect inconsistencies in replicas faster and thereby minimizing the amount of data that has to be transferred.
  - A Merkle tree is used that implements this feature and a merkle tree is quite efficient with logarithmic time access where leaves are hashes of the values and parent nodes are hashes of children.
  - A merkle tree is essentially a Complete Binary search tree with imbalance in the root.
  - Merkle trees use anti-entropy as follows –
    - Each node has a separate merkle tree for each key range.
    - This allows nodes to compare whether keys within the range are up to date or not.
    - A repair coordinators job is to match the two perfect binary trees
- **Components**
  - Three main components
    - Request coordination and membership
    - Failure detection
    - Local persistence engine
- **Performance and Tuning**
  - The primary advantage of Dynamo is that it provides tuning parameters (N,R,W) to tune their instances based on their needs.
  - Using Tokens per node and equal size partitions has
    - Faster Bootstrapping/recovery time.
    - Ease of archival.

Conclusion and Summary

The following paper has carefully analyzed the need of a system that can be targeted at availability and for applications that can function even with a few data loss.

The dynamo DB has combinations of various algorithms like Anti-entropy, Merkle trees , Consistent Hashing and Handoff which makes it very performance oriented and highly available.

Dynamo is also incrementally scalable and meets its SLA's . It also allows clients to fine tune parameters as per their needs.

Dynamo is in production at Amazon and has provided a highly available system that is used by many Amazon services and applications.