**Review 1- Dynamo: Amazon’s Highly Available Key-value Store**

**Summary:** As the title suggests,this paper discusses about Dynamo, highly available key-value store that is deployed in Amazon’s environment to ensure high availability, consistency and efficiency of its services like shopping carts and session management.

This review focuses on Dynamo’s power to combine various distributed system methods to solve Amazon’s issues. This review discusses about the Amazon’s way to deal with partitioning, replication and data versioning.

**Introduction:** Amazon provides one of the largest e-commerce business. For dealing with the large customer base and to ensure a good user experience, Amazon’s system need to be highly available. How a distributed system should be implemented in a way that it achieves high scalability while ensuring high availability and low latency?

**Literature Review:** Traditional production systems used relational database management systems that require complex querying, however, most of the Amazon’s services only uses primary keys to store and retrieve data. Use of Dynamo, that has simple key/value interface, uses a query model to solve the purpose with more consistency efficient resource usage and high availability.

**Implementation:** Dynamo implements a simple system interface that uses get(key) and put (key, context, object) methods to get and put the objects corresponding to a key and a 128-bit identifier is used to determine storage nodes.

*Partitioning*: Dynamo makes use of virtual nodes in the simple hashing based partitioning algorithm that keeps uniformity in data. Dynamo allows the system to scale with addition of nodes. For this, Dynamo relies on consistent hashing that assigns each data item to a node. This allows system to divide data dynamically over a set of nodes. These nodes are set in a ring which makes the departure and arrival of nodes from a cluster only affect the immediate neighbor nodes.

*Replication*: Dynamo replicates the data on multiple(N) hosts and maintains a *preference list* of nodes that store key, this ensures high availability and durability. Each data key is given a coordinator node that is responsible for replicating data to N-1 neighboring hosts in the ring.

*Data Versioning*: Dynamo uses data versioning to propagate the updates to all the replicas asynchronously. To maintain data uniformity, Dynamo allows for multiple versions of an object to be present in the system at the same time, that may be reconciled later. Moreover, Dynamo uses vector clocks (list of node and counter pairs) in order to capture causality between different versions of the same object. Additionally, to ensure that vector clock size doesn’t grow much, Dynamo stores a timestamp that indicates the last time the node updated the data item and when the vector clock list exceed a threshold value, it removes the older pair. However, this may cause inefficiencies in reconciliation.

Also, Dynamo focuses on Service Level Agreements, not just to improve the average response time, but in terms of the response time of the 99.9th percentile. Dynamo uses consistency protocol similar to quorum systems while performing read and write operations among various replicas by the help of two key values R (min no. of nodes while reading) and W (min no. of nodes while writing). However, Dynamo uses sloppy quorum to make sure that the read and write operations are not failed due to temporary node failures.

**Discussion:** From the knowledge gained, it could be said that Dynamo is only a good fit for certain applications. Dynamo only stores simple key-value pairs and it doesn't strictly implement the ACID properties. Certainly, there are applications that would not work with Dynamo. For example, the applications that cannot compromise consistency or maybe those applications that require the values of multiple keys to be updated simultaneously. However, perhaps it would be fine because Dynamo is always available to write. Therefore, all of the keys would most likely get written to the storage, as requests are rarely rejected.

**Results:** Dynamo was designed to address the Amazon’s high availability issue. Dynamo has provided the desired levels of availability, that is always up for writes. Dynamo successfully deployed a blend of database techniques to solve Amazon’s issues.

**Conclusion:** The primary purpose of this paper was to list the techniques from the literature that could be combined to create a highly available distributed system. Dynamo successfully combined the existing database techniques of consistent hashing, object versioning, sloppy quorum, decentralized replica synchronization, gossip-based failure detection, anti-entropy using Merkle trees, and membership management, and vector clocks.