**Cassandra – A decentralized Structured Storage System**

**Summary:** In this work, authors have proposed a new decentralized structured storage system named Cassandra which was designed to solve Facebook inbox search problem.

This review focuses on Cassandra way of dealing with partitioning, replication, membership, failure handling and scaling.

**Introduction:** Cassandra is a NoSQL database management system efficiently designed to manage persistent state ensuring reliability and scalability. The system closely relates to a database but does not support a full relational data model. The paper discusses two main problems - first, whether Cassandra can support high update throughput while delivering low latency along with good read efficiency to support Inbox search. Second, how Cassandra helped Facebook deal with failures in a huge infrastructure environment.

**Literature Review:** Traditional file systems ensure strong data consistency but are not very capable of scalability and availability specially in distributed systems. Modern systems like Ficus and Coda can replicate data at expense of consistency. Farsite and GFS are two distributed file system, however the latter is a centralized one. Bayou is a distributed relational database system. Similarly, Dynamo used by Amazon allows read write operations on distributed network systems during network failures and efficiently execute conflict management. Bigtable provides both structure and data distribution but relies on a distributed file system for its durability.

**Implementation:** Cassandra’s data model comprises of table which is a distributed multi-dimensional map indexed by a key. Cassandra maintains columns grouped together as column family, a design inspired by BigTable. Cassandra API consists of three methods – insert, get and delete that work on column families.

In Cassandra, *partitioning* is done with consistent hashing using order preserving hash function where in Cassandra assigns a key to each of the data element and this key is used to route requests which make departure or arrival of a node only affect its immediate neighbors and other nodes remain unaffected.

*Replication* of data item is done to N hosts, where N is the replication factor configured per instance. Replication is carried out depending upon the policy (Rack aware, Rack unaware, data-center Aware) chosen by the client. The preference list concept borrowed from Aamzon’s Dynamo is incorporated in Cassandra for maintaining the nodes range. Cassandra provides durability guarantees in the presence of node failures and network partitions by relaxing the quorum requirements.

*Membership* in Cassandra is based on Scuttlebutt, a very efficient anti-entropy Gossip based mechanism used also to disseminate other system related control state.

*Failure handling* is done by detecting node failure and making sure that the failed nodes are not contacted. Cassandra uses a modified version of the Φ Accrual Failure Detector in which a suspicion level(Φ) is emitted for each node.

*Scaling* in Cassandra is efficient. When a new node is added into the system, it gets assigned a token such that it can alleviate a heavily loaded node. This results in the new node splitting a range that some other node was previously responsible for. The Cassandra bootstrap algorithm is initiated from any other node in the system by an operator.

**Discussion:** Comparing the data models,Amazon DynamoDB is a *key-value* and *document-oriented* store, while Cassandra is a *column-oriented* data store. On architecture front it’s difficult to compare both, however, according to the CAP theorem, both databases are targeted at availability and partition tolerance. And this will lead to problems with consistency for both databases. Cassandra’s main advantages are lightning speed of writes and reads, constant availability, SQL-like Cassandra Query Language instead of a complex DynamoDB’s API, linear scalability and high performance.

**Results:** Cassandra can support a very high update throughput while delivering low latency. Cassandra storage system provides high scalability, high performance, and wide applicability over humungous distributed data over large networks.

**Conclusion:** Cassandra works efficiently while taking care of data replication, membership, failure detection and scaling. However, there can be some tuning that can be done to make the system more efficient.Future work would be adding parallel data transfer support in node transfer while bootstrapping, adding compression, ability to support atomicity across keys and secondary index support