Paper Review –7

Cassandra

Title: Cassandra: A Decentralized Structure Storage System

Summary

Cassandra is a distributed storage system that helps store large amounts of structured data across commodity servers. Cassandra uses a synthesis of well known techniques to achieve scalability and availability. Cassandra was created in Facebook to help power their inbox search feature.

Good Aspects

In order to meet Facebook demands of performance, reliability and efficiency and continued growth, Cassandra was designed to fulfill this in their inbox search problem. Cassandra deals with failures such a way that its considered a norm rather than exception hence any node failures that occur will only be known to users after a long time since it has its built in reliability and scalability.

Review

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

Related Work

- Before going into the concepts, its important to see what others have done and how far they have succeeded in building a scalable distributed system similar to Cassandra.
  - Ficus & Coda – Replicate files with high availability but don't guarantee consistency. Update conflicts are managed using specialized conflict resolution procedures
  - Farsite – Achieves high scalability & availability using replication.
  - GFS – Google File system uses a master and slaves approach. It has also built-in fault tolerance using Chubby.
  - Dynamo DB- A product from amazon that allows read and write operations to continue even when during network partitions. Has both inbuilt and client driven conflict resolution.

Data Model

- A table in Cassandra is a distributed multi dimensional map indexed by a key.
- The value is an object that is highly structured.
- The row key in a table is a string which is highly structured.
- Columns are grouped together into sets called column families.
- There are 2 types of column families
  - Simple Column Families
  - Super Column Families
- System allows columns to be sorted either by time or by name.
- Any column within the column family is accessed using the convention
  - *column_family : column*
- Any column within the column family that is of type super is accessed using
  - column_family : super_column : column

API

- insert(table, key, rowMutation)
- get(table, key, columnName)
- delete(table, key, columnName)

System Architecture

- These are the following characteristics required in a storage system
  - Scalable & Robust Load Balancing Solution
  - Membership
  - Failure detection
  - Failure recovery
  - Replica synchronization
  - Overload Handling
  - State transfer, Concurrency, Job Scheduling, Request Marshalling
  - Request routing, System Monitoring & Alarming.

- **Partitioning**
  - Cassandra partitions data across the cluster using **Consistent Hashing**
  - In consistent hashing, the output range of the hash function is treated as a fixed circular space mapped to nodes. Each node in such a system is assigned a position in this circular space randomly.
  - Each node becomes responsible for the region in the ring between it and its predecessor node.
  - One of the main advantages of Consistent hashing is that any change in the nodes only affects the neighboring nodes and the hash need not be redistributed like traditional hash tables.

- **Replication**
  - Cassandra uses replication to achieve high availability and durability.
  - Each data item is replicated at N hosts, where N is called replication factor.
  - Cassandra has unique algorithms for replication policies like Rack Unaware, Rack Aware & Datacenter Aware.
  - For Rack Aware and Datacenter aware strategies, Zookeeper is used to elect leader amongst nodes. All nodes joining a cluster ask the leader for what ranges they are replicas for.

- **Membership**
  - Cluster membership in Cassandra is based on Scuttlebutt, a efficient anti-entropy Gossip based mechanism.
    - **Failure Detection**
      - In order to detect a node failure, instead of direct No or Yes based mechanism, it uses Accrual Failure Detector.
      - Instead of generating a Boolean value, it generates a suspicion value phi that uses a Exponential Distribution.
- **Bootstrapping**
  - When a node starts for first time, a random token for its position is taken in the ring.
  - This token information is gossiped around the cluster.
  - If a node needs to join a cluster, it can read the configuration file which contains a list of contact points within the cluster.
  - These seeds can also come from Zookeeper.

- **Scaling the cluster**
  - When ever a node is added into the system, it gets assigned a token such that it can alleviate a heavily loaded node.

- **Local Persistence**
  - Cassandra relies on local file system for data persistence.
  - Write involves write to a commit log for durability and recoverability.
  - Update in-memory is done after a successful write to commit log.

- A dedicated disk on each machine is present for commit log and all data is stored sequentially.
- When the in memory data structure crosses a certain threshold, its dumped to disk.
- All writes are sequential and generate index for efficient lookup based on row key.
- These indices are also persisted along with the data file.
- Over time just like big table, all these files are merged or collated to one big file.
- Typical read is first done in memory then goes to disk.
- Since for a given key, it can be stored in multiple files, a bloom filter is used which provides membership info for a key in a certain file.

- **Implementation Details**
  - These are the following abstractions in Cassandra: -
    - Partitioning Module
    - Cluster Membership
    - Failure detection
    - Storage engine
  - Each of these modules rely on an event substrate where the message processing pipeline and task pipeline are split into multiple stages like SEDA (Staged Event Driven Architecture).
  - Cluster membership and failure detection is built on Network layer which uses non blocking I/O that rely on UDP messages.
  - When a read/write request arrives at any node, the state machine morphs through these states –
    - Identify the nodes that own data for the key
    - Route the requests to the nodes and wait for responses
    - If the replicas don't arrive after timeout, then return failure.
    - Figure out the latest response based on timestamp
    - Schedule a repair of the data at any replica if they don't have the latest piece of data.
    - The waits can be synchronous or asynchronous.
    - For synchronous, node waits for a quorum of responses.
  - Purging commit logs are done at every 128MB size. Every commit log has a header which is basically a bit vector. Every time the in memory data structure for a particular column family is dumped into the disk, the corresponding bit is set.
  - The server instance of Cassandra is practically lockless for read/write operations.
  - Cassandra indexes all data based on the primary key. Each data file is broken down into blocks Each block consists of max 128 keys and is demarcated by a block index.

Practical Applications

- Facebook Inbox Search
    - There are two kinds of search features – Term and Interactions
    - For Term, the key is user id and the words that make up the message become the super column.
    - Individual message identifiers of the message that contain the word become the columns within the super column.
    - For Query based on user, the user id is key and recipient id are super columns.
    - In order to make inbox search faster, intelligent caching is employed.

Conclusion and Summary

The following paper presents a new distributed database that employs lot of intelligent algorithms to provide high availability, Scalability, Data replication.

Cassandra system has benefitted greatly at Facebook by help powering the Inbox search feature and is still in use in other areas in Facebook. Cassandra has employed lots of techniques from different Distributed systems, but at the same time has used advanced algorithms and tweaked some to obtain best efficiency and throughput.

By minimizing data transfer to ensure consistency, it helps provide consistency without much expense and still be able to scale quite well by employing something like a peer to peer system.