Facebook: Cassandra

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Outline

Motivation

Motivation

- Social Networking $\Rightarrow$ Facebook
- Facebook is HUGE
  - Billion+ users
  - Millions of updates per minute
  - Many thousands of servers
  - Reliability is a big issue
- Cassandra: Made for the Inbox Search problem
  - Allows users to search through their Facebook Inbox
- It is also used for many other Facebook services

Related Work

- Distributed File Systems: Coda, AFS: Hiearchical name space, specialized conflict resolution.
- Distributed Databases: Bayou
- Dynamo: Uses vector clocks to manage versions. Conflict resolution based on the vector timestamp, and specialized business logic.
  - Gossip based membership algorithm
- BigTable: Distributes data. However, it relies on an underlying distributed file system for durability (similar to Voldemort).

Outline

Data Model

- Table $\rightarrow$ Distributed multi-dimensional Map
- Key $\rightarrow$ 16 to 36 byte long string to uniquely identify a row
- Value $\rightarrow$ Highly structured object
- Every row operation is atomic on a replica
- Columns can be grouped into families
  - Simple and Super (family within a family)
  - Columns can be sorted by either the time or by name

API

API

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

System Architecture

- Cassandra has a cluster of nodes
- A read/write request for a key gets routed to a node in the cluster
- The node determines the replicas that contain the key
- We write to the write quorum .
- Either read from the closest replica or from the read quorum

Outline

Partitioning and Replication

- Partitions data using consistent hashing.
- Each data item is replicated on *N* hosts
- Various replication policies:
  - Rack Unaware
  - Rack Aware
  - Datacenter Aware
- Cassandra first elects a leader using Apache Zookeeper
- The leader assigns replicas to nodes.
- The meta-data, routing tables, and topology information are maintained at each node and also centrally by Zookeeper.

Membership

- Uses the Scuttlebutt anti-entropy based gossip system to propagate membership information.
- Failure detection
  - Φ Accrual failure detector
  - The failure detector gives a value, Φ, for each node that is not a Boolean number. It represents the level of suspicion for the node.
  - If $\Phi = 1$, the chance of an error is 10%
  - If $\Phi = 2$, the chance of an error is 1%
  - Every node maintains a sliding window of inter-arrival times of gossip messages
  - Φ is calculated based on the distribution of these inter-arrival times.

Bootstrapping

- When a node starts, it chooses a random token. The token defines its position in the ring.
- It saves the token locally and also gives it to Zookeeper.
- It then gets a list of contact points from Zookeeper, and contacts them such that it can join the ring.
- The membership information for the new ring is then gossiped in the cluster.
- There might be multiple Cassandra instances running (for different types of services)
  - Tag each message with the instance id
  - Administrators can manually manage Cassandra instances

Scaling

- The token assignment can be made more intelligent . Assign a node a position in the ring such that it can take some load off a heavily loaded node.
- Use kernel-kernel copy techniques to stream data from the old to the new nodes.
  - Parallelize this process by streaming data from multiple replicas
  - Similar to BitTorrent

Outline

Local Persistence

- Relies on the local file system for data persistence.
- Typical write in a fault tolerant file system:
  - Write to a commit log
  - Update an in-memory data structure
- Dedicated disk for the commit log
- Once the in-memory data structure exceeds a certain size, dump its contents to the commit log
- Writes are sequential to the disk. Simultaneously an index for disk data is created.
- Different files in the disk are later merged (compaction).

Read Operation

- First query the in-memory data structure.
- Search the files from newest to oldest.
- To speed up searches we can use a Bloom filter.
  - Can quickly indicate that a set of files do not contain a key
- It is often difficult to locate all the columns in a row.
  - Solution: Use a 256 KB index

Implementation Details

Main Modules

Partitioning Module, Cluster membership module, Failure detection module, Storage engine module

- Written in Java using non-blocking I/O primitives
- Control messages use UDP
- All application related messages use TCP

Detailed: List of steps

1. Identify the nodes that own the data for a key
2. Route the requests to the nodes and wait for the responses
3. If the replies do not arrive within a certain time, fail the request
4. Figure out the latest version based on the timestamp
5. Trigger a repair (if required) and return to the client

Journalling File System

- Uses a rolling commit log
  - Create a new one after the old one has reached a certain size
  - Use a size of 128 MB
- In Cassandra : Use an in-memory data structure, and a backing data file for each column family
- Every time the in-memory data structure is dumped to the disk, we set a bit in its commit log
- Each commit log maintains a bit vector (corresponding to disk dumps)
- We can delete commit logs for entries that have been persisted

The write operation

- Write operation
  - Normal mode : Unbuffered writes
  - Fast sync mode : All writes to the commit log, and the data file are buffered. Can lose data on a disk crash.
- Sequentialize writes to the disk.
  - Fast
  - Does not require costly B-tree updates

Cassandra Index

- All the data is indexed based on the primary key
- The data file on the disk is broken down into a sequence of blocks. Each block contains at most 128 keys.
- A block is demarcated by the block index , which captures the relative offset of a key within a block, and contains the size of its data.
- This index is also saved in memory.
- Search process:
  - First search the index in memory.
  - If it is not in memory, search index in the disk.
  - Locate the key's value in the in-memory data structure.
  - If not search in the data file.

Arrangement of Files

- Files are ordered according to their time of creation.
- We start by accessing the latest file (since we want the latest update)
- Because of temporal locality of accesses, we keep recent data in small data files
- Gradually we merge them and create bigger merged files

Lessons Learnt

- Map-reduce jobs process raw data (or from a MySQL database) and send data to the Cassandra instance.
- Implemented: atomic operation per key per replica
- Average failure detection time: 15 seconds
- The usage of Zookeeper as a co-ordination service is very useful.

Facebook Inbox Search

- Maintains a per user index of all messages sent and received
- Search features
  - (a)Term search
  - (b) Search by the name of the recipient
- For (a)
  - User id is the key, and the message words are the super column. Message ids are columns within the super column.
- For (b)
  - User id is the key and the recipient's ids are the super columns. For each super column, the message id is the column.
- Uses index prefetching


Lakshman, Avinash, and Prashant Malik. "Cassandra: a decentralized structured storage system." ACM SIGOPS Operating Systems Review 44.2 (2010): 35-40.