Cassandra – A Decentralized Structured Storage System

Naga Rekha Malae (nrmalae@indiana.edu)

Abstract

Cassandra is a distributed **database** bringing together Dynamo's fully distributed design and Bigtable's ColumnFamily-based data model. It is highly scalable both in terms of storage volume and request throughput while not being subject to any single point of failure. This paper presents an architectural overview of **Cassandra** and also discusses how its design is founded in fundamental principles of distributed systems. Some practical applications that use Cassandra are also listed. This paper also presents why Cassandra may not be a good choice for some applications by highlighting its limitations.

Introduction

Cassandra is a distributed key-value store capable of scaling to arbitrarily large sets with no single point
of failure [1]. Every node in the cluster is identical. There are no network bottlenecks. Data is automatically replicated to multiple nodes , racks and even multiple data centers for fault-tolerance. Failed nodes can be replaced with no downtime. Read and write throughput both increase linearly as new machines are added[2].
Cassandra's core design brings together the data model described in Google's Bigtable paper [4] and the eventual consistency behavior of Amazon's Dynamo [4]. **Cassandra** was initially developed by Facebook as a data platform to build many of their social services such as Inbox Search that scale to several hundreds of millions of users [3]. It was submitted to the Apache Software Foundation Incubator in 2009 and in 2010
March **Cassandra** was accepted as a top-level Apache project[5].

Data Model

A table in Cassandra is a map which is distributed in dimensions. It is indexed by a key. The value is an object which is very highly structured. The row key in a table is just a string with no restrictions on size, although typically 16 to 36 bytes long. Every operation such as insert, get, delete etc under a single row key is atomic per replica no matter how many columns are being read or written into.

- Column: A column is the atomic unit of information and is expressed in the form name : value.
- Super Column: Super columns group together like columns with a common name and are useful for modeling complex data types such as addresses other simple data structures.
- Row: A Row is the data which is uniquely identifiable by a key. It also groups together columns and super columns.

- Column Families : Columns are grouped together into groups known as column families very much similar to what happens in the Google Bigtable[4]. There are two such column families in Cassandra. Simple column families and Super column families. A column family within another column family can be described as Super column families. The columns can be sorted either by time or by name. Time sorting is generally preferred for searching inbox where the results are displayed in the time sorted order. Although Cassandra system supports the notion of multiple tables all

deployments have only one table in their schema[3].

- Keyspace: The Keyspace is the top level unit of information in **Cassandra system**. Its general representation would be get(keyspace, column family, row key)

API

The Cassandra API consists of the following three simple methods.

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

A columnName can refer to a specific column within a column family or a super column family, or a column within a super column.

Distributed hash Tables

Before going into Cassandra architecture details lets first discuss what exactly is the distributed hash table and why, where and when is it used.
A **distributed hash table** (**DHT**) is a class of a decentralized distributed system that provides a lookup service similar to a hash table; (*key*, *value*) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures[6]. Any read or write in the DHT locates the node containing information by means of key-based routing algorithms. This is called as lookup. The lookup complexity in Cassandra's DHT is O(1) and is referred to as one-hop DHT. The Cassandra architecture uses DHT gossip functionality. It means that eventually every node has state information such as nodes availability, keys it is responsible for etc of every other node. Nodes in Cassandra cluster are arranged in a ring fashion. Servers are numbered sequentially around the ring with the highest numbered connecting back to the lowest numbered. Each server has a uniquely assigned token which represents the range of keys for which it is responsible for. The value of a token t may be any integer such that $0 \leq t \leq 2127$. Keys in **Cassandra** may be a sequence of bytes or a 64-bit integer. However, they are converted into the token domain using a consistent hashing algorithm. By default MD5 hashing is used. If required an application may supply its own hashing function.

Architecture

In the following sections we discuss how **Cassandra has achieved** several goals – scalability, high performance, high availability and applicability.

Anatomy of reads and writes

To the users, all nodes in the Cassandra look identical. The fact that each node is responsible for managing different part of the whole data is transparent to the users. Each and every node is responsible for servicing any request made by the user whether it is a write or read operation. Requests are directed to the appropriate node by checking the key against the local replica of the token table. Once a write request reaches the appropriate node it is written to the commit log. A write request will not return a response until that write is durable in the commit log. Simultaneously memtable is updated with this write. Once the memtable reaches to maximum size it s contents are periodically flushed into a durable storage known as SSTable. Reads are much more intensive than writes. Reads first search the memcache for any requested column and then search SSTables. As the number of keys increase the number of SSTables also increases. To control the read latency SSTables are periodically combined [1].

Replication

Cassandra uses replication strategy to achieve availability and durability. Each data item is replicated to N hosts. Cassandra provides users various replication policies such as Rack ware, Rack unaware and Datacenter aware. In Rack unaware policy non-cordinator replicas are chosen by picking N-1 successors of the coordinator on the ring[4]. For Rack Aware and datacenter Aware policies the algorithm is more involved. Cassandra system elects a leader amongst several nodes by using a system called Zookeeper[23]. When nodes want to join the network they first contact the leader who tells them for what ranges they are replicas. The metadata about the ranges a node is responsible is cached locally at each node. This way even when the node is crashed it could be easily come back to its original position as it knows the range of nodes it was previously responsible for. As explained before every node is aware of every other node and the range they are responsible for. Datacenter failures are generally caused due to power failures, cooling failures and natural disasters. Cassandra provides durability guarantee even in these kinds of situations because it is configured in such a way that each row is replicated across multiple datacenters and these are connected through high speed network links.

Consistency

Cassandra provides consistent data store. Cassandra allows the user to make tradeoff s between consistency and latency. The various consistency levels a client can specify are ZERO, ONE, QUORUM, ALL, or ANY with each read or write operation. These consistency levels should be used in such a way that there should be a appropriate balance between consistency and latency for the application. According to [3]

- A consistency level of ZERO indicates that a write should be processed completely asynchronously to
  the client. This gives no consistency guarantee but offers the lowest possible latency. This mode must only be used when the write operation can happen at most once and consistency is unimportant since
  there is no guarantee that write will be durable and ever seen by another read operation.

- A consistency level of ONE means that the write request won't return until at least one server where the key is stored has written the new data to its commit log. If no member of the replica group for applicable token is available, the write fails. Even if the server crashes immediately following this operation, the new data is guaranteed to eventually turn up for all reads after being brought back online.

- A consistency level of ALL means that a write will fail unless all replicas are updated durably.

- QUORUM requires that N+ 1 server should have durable copies where N is the number of replicas.

- In Any Consistency mode nodes can perform hintedoff mechanism. It means that when a request is sent to o node in the cluster, if that node is not responsible for the key then the requested is transparently proxide to the replica for that token. In this way the request eventually gets to the correct replica node.

Scaling the cluster

The process of introducing a new node into Cassandra cluster is known as boot strapping. In Cassandra system when a new node joins it immediately gets assigned a token such that it reduces the load of a heavily loaded node. This results that the new node splits the range that previous node is responsible for. The Cassandra bootstrap algorithm is initiated by an administrator in the system either by using the command line utility or the Cassandra web dashboard. Data can

be transferred at the rate of 40 MB/sec from a single node.

Elastic Storage

The major beauty of Cassandra is that it allows scaling to large datasets without rethinking the fundamental approach to storage. As storage requirements grow hardware requirements and costs also grow linearly. Clusters in Cassandra scale through the addition of new servers rather than purchasing more powerful servers. From this it can be inferred that Cassandra scales up horizontally and as well as vertically.

Client Access

Client interaction with a decentralized distributed system like **Cassandra** can be challenging. One of the obvious problems for the client would be to decide to which host it has to send the request. But in Cassendra, every node is capable of responding to any request initiated by the client. The data store in Cassandra is implemented in java. But there is no native java API to use Cassandra from a separate address space. Instead implements services using thrift. Cassandra's client API is built entirely on top of Thrift[2].

Prominent Users of Cassandra

- Cisco's WebEx uses Cassandra to store user feed and activity in near real time.[9]
- Cloudkick uses Cassandra to store the server metrics of their users.[10]
- Digg, a large social news website, announced on Sep 9th, 2009 that it is rolling out its use of Cassandra[11] and confirmed this on March 8, 2010.[12] TechCrunch has since linked Cassandra to Digg v4 reliability criticisms and recent company struggles.[13] Lead engineers at Digg later rebuked these criticisms as red herring and blamed a lack of load testing.[14]
- Facebook uses Cassandra to power Inbox Search, with over 200 nodes deployed.[2] This was abandoned in late 2010.[15]
- IBM has done research in building a scalable email system based on Cassandra.[16]
- Rackspace is known to use Cassandra internally.[17]
- Reddit switched to Cassandra from memcacheDB on March 12, 2010[18] and experienced some problems with overload handling in Cassandra in May.[19]
- Twitter announced it is planning to use Cassandra because it can be run on large server clusters and is capable of taking in very large amounts of data at a time.[20][21] Twitter continues to use it but not for Tweets themselves.[22]
- Yakaz uses Cassandra on a five-node cluster to store millions of images as well as its social data.[23]

Cassandra limitations

Usage of Cassandra does not guarantee acid properties. Cassandra does not support a fully relational data model. So application designers choosing a data store should consider these criteria carefully against their requirements while selecting a data store. In some cases, if it is possible applications designers may still use Cassandra by choosing to put some subset of data in a Cassandra data store and some other data which needed to be guaranteed to ACID properties in the transactional database.

Conclusion

This paper has provided an introduction to Cassandra and Fundamental distributed principles on which it is based. This paper also discusses how the data can be distributed across the networks to maximize the availability at the time of node failure and network portioning. The architecture of the Cassandra system is very well explained. I conclude that Cassandra has succeeded in achieving several goals of the distributed database such as scalability, high performance, high availability and applicability. Cassandra almost resembles a database. It shares many design and implementation strategies with databases. Cassandra does not support a full relational data model.

References:

[1] Cassandra: Principles and Application, Dietrich Featherston.

[2] http://cassandra.apache.org/

[3] Cassandra - A Decentralized Structured Storage System, Avinash lakshman and Prashant Malik.

[4] http://en.wikipedia.org/wiki/Distributed hash table

[5] Apache Software Foundation, *The Apache Software Foundation Announces New Top-Level Projects*

[6] http://en.wikipedia.org/wiki/Distributed_hash_table

[7] Cassandra users survey, Mail-archive.com. 2009-11-21. Retrieved 2010-03

[8] https://www.cloudkick.com/blog/2010/mar/02/4_months_with_cassandra/

[9]Ian Eure. "Looking to the future with Cassandra".

[10] John Quinn. "Saying Yes to NoSQL; Going Steady with Cassandra".

[11] Erick Schonfeld. "As Digg Struggles, VP Of Engineering Is Shown The Door".

[12] Is Cassandra to Blame for Digg v4's Failures?

[13] Kannan Muthukkaruppan. "The Underlying Technology of Messages".

[14] Powered by Google Docs. Docs.google.com. Retrieved 2010-03-29.

[15] N2.nabble.com. Retrieved 2010-03-29.

[16]Posted by david [ketralnis] (2010-03-12). "what's new on reddit: She who entangles men". blog.reddit. Retrieved 2010-03-29.

[17] Posted by the reddit admins at (2010-05-11). "blog.reddit -- what's new on reddit: reddit's May 2010 "State of the Servers" report". blog.reddit. Retrieved 2010-05-16.

[18] Posted by the reddit admins at (2010-05-11). "blog.reddit -- what's new on reddit: reddit's May 2010 "State of the Servers" report". blog.reddit. Retrieved 2010-05-16.

[19] Popescu, Alex. "Cassandra @ Twitter: An Interview with Ryan King". myNoSQL. Retrieved 2010-03-29.

[20]Babcock, Charles. "Twitter Drops MySQL For Cassandra - Cloud databases". InformationWeek. Retrieved 2010-03-29.

[21] Cassandra at Twitter Today.

[22] Yakaz Technologies.

[23] Benjamin Reed and Flavio Junquieira. Zookeeper.