Apache Cassandra
Distributed Database Management System

Jayesh Kawli
Master of Science (Computer Science)
School of Informatics and Computing
Indiana University, Bloomington
(+1) (419) 285-6105
jkawli@indiana.edu

ABSTRACT
This survey paper gives detailed overview of Apache Cassandra – A distributed database management system. We are going step by step from its introduction to the evolution and then to the paradigm where it is used by major internet companies. While doing so, we will also discuss its features, relationship with other similar technologies, tools available and ongoing research to improve its performance.

1. Introduction

In its basic terminology, Cassandra can be regarded as distributed database system with combination of technologies from Amazon Dynamo and Google BigTable [5]. The roots of Apache Cassandra lie in the NoSQL database requirement for Facebook Corporation. Cassandra was developed by Avinash Lakshman and Prashant Malik at Facebook Corporation to boost its inbox search feature. It was designed to be a mean of database storage for distributed architecture. Being deployable on distributed platform it was highly expected to be able to handle data spread across geographically diverse servers, capable of providing seamless service with built in fault tolerance and no single point of failure [2]. One of the most surprising features of Cassandra is that, though it is said to be sharing a lot of design and internal architecture details with traditional relational database management system, it is by no mean a relational database system. Cassandra is responsible for providing a data model to its users which can then be customized according to data storage and access requirements. In short, Cassandra can be described as an enormously scalable, decentralized and fault tolerant database management system which stores attribute values in structured and indexed fashion for efficient querying using Cassandra query language (CQL). [4][5][7]

Implementing inbox search feature was one of the difficult tasks given the existing relational database management system. It required very high write throughput capability. Though possible, it was infeasible and inefficient to implements this feature with very high number of geographically diverse users. Since its introduction in 2008 number of Facebook there has been more than double increase in number of users and still Cassandra is giving satisfying performance. Not only by Facebook, but due to its rich set of features Cassandra has been deployed by various major E-commerce businesses such as Netflix, digg and Twitter to name a few. [2][5][7]

2. General Terms

While reading this survey paper reader might come across following terms related to Cassandra and Distributed system.

Scalability, high availability, fault tolerance, replication, BigTable, clustering, partitioning, NoSQL, bootstrapping, node, persistence, namespace, latency etc.

3. Evolution

Cassandra database system was born at Facebook in 2007 as a resource to handle inbox search feature which consisted of high scalability and real time usage. It was published as an open source project on Google code in 2008 and became top-level project of apache. [1][7]

Since its release it has undergone numerous changes and additions. In the version released in 2010, it added support for integrated caching. For year 2011 version it included support for Cassandra query language (CQL) and support for upgrades with no server downtime is required by many real time websites such as Facebook, Google and Amazon. As a part of recent revision it exhibits more advanced features such as support for SSD, self-tuning caches and row-level isolation. [1][2]

4. Related Work

Since the evolution of distributed file system, there has always been necessity to develop a database management system which is capable of providing core distributed features such as reliability, consistency, availability and scalability to span across mass base of real time users.

Before introduction of Cassandra, there has been significant research in distributed file and database system management. There are Ficus and Code DFS (distributed file systems) which replicate data across multiple geographical regions. However this system faces concerns while maintaining synchronization among geographical diverse data. Farsite is another DFS which does not have master server but provides similar benefits as that of distributed file system using replication. However due to lack of master, all servers continuously have to share data among all the other members of underlying infrastructure. [2][7]

Google developed their own file system known as Google file system (GFS) which manages multiple slave servers using singe master server. Master stores metadata and information about how data is divided and distributed among multiple children servers. This whole system is made fault tolerant using chubby abstraction for GFS master server. Among all the discussed DFS, they are capable of providing geographical and usage scalability along with consistency which is required by many current e-commerce service providers. However, these file systems differ in the way in the conflict resolution mechanism while providing consistency to geographically distributed data and file system. [7][8]

5. Data Model

5.1. Structure and Organization

Cassandra uses multidimensional table to store the values where each value is indexed by the key. Usage of key allows to access entire column as an atomic operation. However Cassandra does not put any restriction on size of key. However in practice depending upon application and usage key size typically varies between 16 to 36 bytes. Speaking about data columns, they are grouped into column families. Column families are further grouped into super column families. Cassandra allows application to sort the column values according to timestamp or their attribute values. This is beneficial in many cases where results need to be sorted according to their timestamps or associated values. [5][7]

Following figure shows diagrammatic representation of data model structure used by Cassandra. First column is occupied by key for indexing where remaining columns are used to store attribute values which are accessed using key column as a single atomic operation no matter how many reads or writes take place.

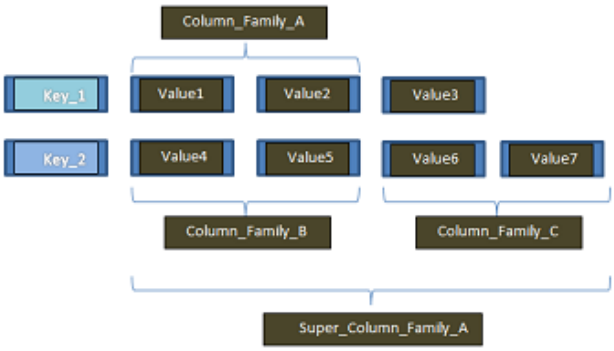

Figure 5.1 Cassandra data model

Referring to above figure, column names in column family and super column families can be accessed as follows.

Column_Family_A:value1

And Super_column_Family_A:Column_Family_B:Value4 [7]

5.2. Querying

Cassandra offers flexibility in way in which database can be queried. User can access single column or multiple column having distinct keys for bulk data access. Since for a given key column read/write operation is atomic, it does not matter how many columns are being written or read by the application. It is also possible to query on individual column family or super column family as a single request. Cassandra also supports accessing column set by specifying range of key indices. [5] [7]

5.3. Available database APIs [1][7]

Cassandra allows permissible operations on database using following set of basic APIs

insert (tablename, keyname, rowMutation)

get (table, key, columnName)

delete (table, key, columnName)

tablename: Name of the distributed table

keyname: Unique key value used to index data items. As long as single table is concerned, key identifies each row uniquely.

rowmutation: Parameter used to indicate writes are more frequent than reads. This happens in cases such as database Flush where sort and write operations are much slower than expected.

Cassandra APIs are available in different languages such as Ruby,

Python and Scala. For example Cassandra Thrift client API is offered in 12 different languages. [1][5]

6. Cassandra Architecture

Given the performance of Cassandra, it has very complex architecture to support all those features. It provides various fundamental as well as extended features required by an ideal distributed database management system such as data persistence, scalability, load balancing, fault tolerance, concurrency and robustness. Besides these it also offers various facilities related to administrative tasks such as task scheduling, system monitoring, failure notification, alarming and configuration management. These tasks are divided among dedicated system components and work in synchronization with each other for each read/write. [1][7]

Read/write operations are performed based upon the response received from quorum of replicas for particular key index. Response is needed to verify that all the distributed replicas are in sync with each other.

Some of the key features of Cassandra are as follows:

6.1. Partitioning

Cassandra provides incremental partitioning feature to handle the large amount of data being inserted into database. Cassandra maintains a list of nodes and as the data comes, it distributes it among nodes in either consistent or inconsistent manner. In the former approach system evaluates the given data using hash function and given key. Then data is assigned to node corresponding the value generated by hash function. Hash function is designed to treat cluster of nodes in circular fashion where largest node value wraps to the value of lowest node. Partitioning takes care of choosing coordinator for nodes where all the requests for that node are routed. [7]

6.2. Replication

Cassandra is responsible for providing high data availability by replicating data across remaining N replicas by quorum protocol to acknowledge read/writes spread to all the replicas. Whenever a node is updated/wrote by an application, coordinator spreads relevant updates across remaining N-1 nodes. User has an option to specify how this replication should take place across all N replicas. They include various policies such as ‘Rack Aware’, ‘Rack Unaware’ and ‘Datacenter Aware’. Policies decide the order in which replicas will be chosen to spread updates. [7]

6.3. Membership

Cassandra does not introduce new mechanism of its own to maintain membership among all system nodes. It uses Scuttlebutt gossip protocol to maintain membership among participating nodes in the system. It is responsible to carry out frequent checks for membership information and spread out relevant information to all the other nodes so that all the nodes within system boundary

know about existing, live and communicable nodes. Besides transmitting membership information, scuttlebutt can also be used to spread out system related control information. [7]

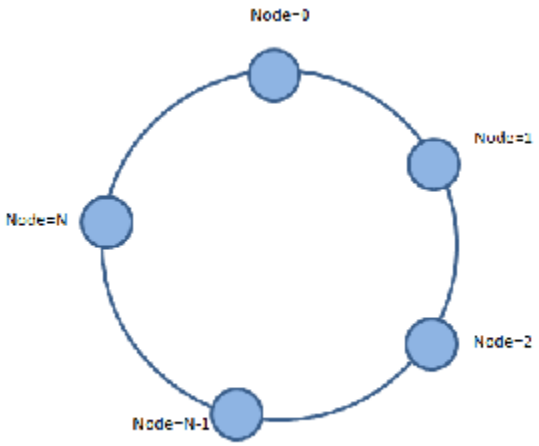

Figure 3.4 Membership distributions for system nodes [5] [6]

Membership diagram for N nodes arranged in circular fashion.

Hash function is outvalue = (Key) % (N) where data is assigned to the node with sequence number=outvalue

6.4. Failure detection

It involves the task of checking whether any of the system nodes is up or down. Unlike traditional deterministic failure detection mechanism, Cassandra uses more efficient probabilistic model to check if node is down or up and running. This approach is based upon the accrual failure detection algorithm which uses the threshold parameter denoted asΦ. Its value is set according to local network conditions and load present at particular node for which we want to detect failure. As a part of failure detection, module emits the value known as suspicion level value which reflects the fact that current node is down. Now probability that this statement is false (due to late reception) depends upon the value of set thresholdΦ. More the value of Φ more is the probability that suspected node is actually down (And decision won't be affected by late message reception). Following table reflects this fact where probability is mapped against the value of threshold parameter Φ

Sequence No.	Φ	Probability of false detection
1	1	10
2	2	1
3	3	0.1
4	4	0.01
5	5	0.001

Accrual failure detection algorithm is flexible in a way that it has capability to adjust as per network conditions and server load. [7]

6.5. Load Balancing mechanism – At Nodes

When new data node is added to system, hash function evaluates and assigns a token in such a way that it can load balance heavily loaded node. Load balancing heavily loaded node is also called as Bootstrapping. The process of Bootstrapping can be initiated either through command line or Cassandra dashboard user interface. Whenever bootstrapping occurs, specific ranges of nodes get transferred from old node to newly introduced node. Copy operation is performed in data stream replication mechanism which takes place in kernel-kernel copy operation. However this example shows only single replica participating in Bootstrapping. It is also possible to deploy more replicas to perform parallel Bootstrapping on the behalf of participating nodes. [7]

6.6. Read and Write Operations

6.6.1. Write

Cassandra writes data in the persistent local file system. Data is stored in the format such that it would allow efficient and reliable read results. However before storing in the file system, it writes data in the commit log for durability in order to recover from any possible system failovers. After system's confirmation that desired data has been successfully written to commit log it actually performs write to the file system residing on system disk. Data is stored according to index defined on each key in a row for efficient search and query processing. Each write is maintained as separate file. Frequently a process called as merge is ran which unifies these files into single big file. [7]

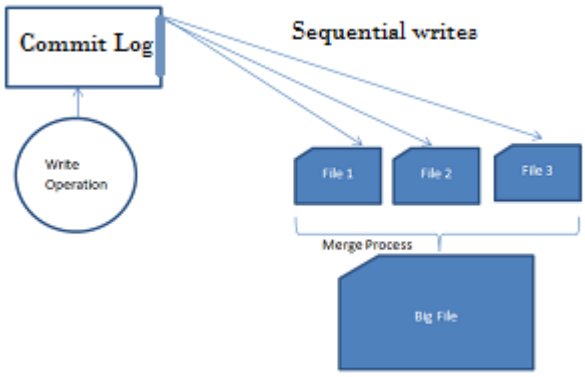

Figure 6.6.1 Cassandra writes operation

6.6.2. Read

Read operation takes help from in memory data structure which helps in locating key indices corresponding to relevant rows. Each row is uniquely identified and indexed by the key column associated with it. While looking for the key, it is possible to perform lookup in a file which does not has relevant key. This overhead can be eliminated by using bloom filter which makes use of another file which contains list of keys associated with that particular data file. This mechanism avoids look up in the data file which does not have specific key. As Cassandra offers this facility, column pair (Key, timestamp) are sorted in descending order of timestamps so that latest record always appears first. This

is in accordance with the fact that most users are in need of latest version of available data.

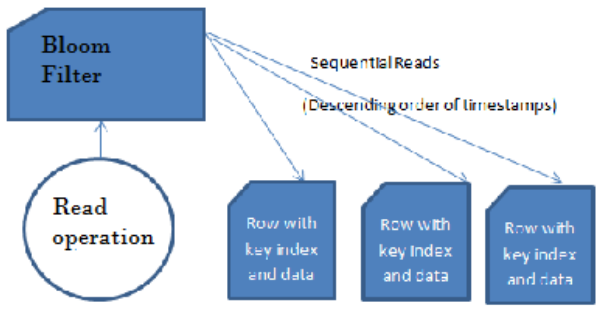

Figure 6.6.2 Cassandra read operation

7. Performance testing for Cassandra with production data

Cassandra was tested with production data of 100M users with size over 7 TB. Operation involved migrating data from legacy MySQL system to new Cassandra system using Map reduce technology. Since background processes were deployed to take care of reverse indexing and serialization, performance was affected only due to network latencies. [7]

It was observed that some of the applications suffered significant loss due to Cassandra's inability to provide secondary indices. [7]

Before actual production deployment, Cassandra was tested with various fault tolerant mechanisms. However, results depict that Accrual fault detector took much less time to detect faulty node.

Although virtually designed as decentralized distributed database management system, it was observed from various experiments that Cassandra needs somewhat support from external Coordination manager to keep track of various activities taking place in distributed environment. Some scientists suggest use of 'Zookeeper' distributed coordination control system. [7]

8. Performance improvement

Cassandra was tested on Facebook inbox data with more than 50 TB storage having total 150 Nodes. These nodes were spread evenly between east and west coast data center. Cassandra makes use of unique user id to associate them with messages exchanged between given pair of person. Inbox search is done either on the entities involved in or content of conversation. Whenever user clicks on search box, asynchronous message is sent to cluster which gathers a data into buffer with user's unique key index. In this way possible search results are already loaded into buffer memory before user starts typing in actual query. [7]

Result of performance testing done on production data is shown below.

Data size: 50+ TB Nodes: 150 Nodes in cluster

Latency Measurement	Search with Message word (ms)	Search with Recipient id (ms)
Minimum	7.7	7.6
Maximum	26.1	44.4
Median	15.7	18.3

Table 8 Performance evaluation for Cassandra

9. Deficiencies and Future improvements

Cassandra does not provide secondary indices. This can be time consuming when given the query, application would have to iterate over primary index until it finds match. This can be jeopardizing while dealing with large amount of data. Issue can be mitigated by maintaining separate table to stores secondary indices and keep it in synch with old table.

Cassandra as with most other NoSQL databases does not provide support for joins. Instead it gives emphasis on row duplication and denormalization techniques. One way to avoid this problem is denormalize join queries before actually applying them to database tables.

As far as current implementation is concerned, Cassandra does not add significant compression. However from future perspective this feature will play an important role as number of users increase. Compression will not only increase storage capacity but will also bring in significant time saving in terms of read and replication latency. [7][5]

10. Performance comparison

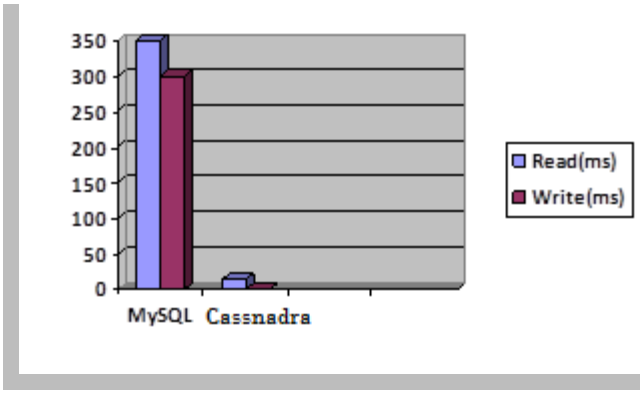

Figure 10 MySQL vs. Cassandra (Data size 50 GB+)

Chart above shows performance comparison of MySQL vs. Cassandra. From chart size it is obvious that there is significant performance improvement when switched from MySQL to Cassandra. [5][2]

Although read time is slightly more for Cassandra compared to write operation, it plays pivotal roles for offering high availability and low downtime during frequent maintenance and update operations.

11. Cassandra Features – Utility in real time applications

Cassandra provides data independence where applications do not worry about where data is located. This is useful for applications which are designed to server diverse geographical areas without have to deal with communication latency. [2] [6]

Cassandra is designed to be highly scalable database system. This is beneficial for many social networking websites to which require managing terabytes/petabytes of data for analytical and statistical processing. [6]

Cassandra is useful for cloud applications where is provides easy deployments, fault tolerance and ability to scale towards large infrastructures as data need grows. [6]

One of the distinctive features of Cassandra is that its writes are much faster than reads. This feature can be harnessed by many write intensive applications such as files/music upload, event logging, system monitoring, emails and private messaging facilities on social networking websites. [5][6][7]

12. Improvements over RDBMS [6]

Ability to scale to larger size with no significant processing time penalty as data grows (Terabytes/Petabytes)

Higher performance for both read/write operations

Tunable data consistency to offer persistence and protection with additional capability to adjust it up according to underlying application

Flexible schema design with ability to store any kind of structured, semi-structured or unstructured data

Data replication and consistency facility with low down time during maintenance with support to data center and cloud applications

13. Business corporations using Cassandra database

Twitter

Twitter is a social networking website which allows users to post messages up to 140 words size. Main challenges with this application are scalability, diversity and consistency across geographically diverse applications. Cassandra helps to take care of these issues with built in fault tolerance, scalable architecture and low downtime maintenance capability. [2][5]

Digg

Digg is another social network which allows users to vote any internet article up or down. Due to large volume of users posting their feedbacks Digg has expected problem of handling and managing large volume of data. Cassandra provides highly scalable architecture with no single point of failure and recovery. Replication and consistency mechanism help to improve availability and reduce access time. [1][2][5]

Formspring

Formspring is a social networking website which allows subscribed users to ask/answer questions or put forward their opinions about any topic they choose. Cassandra is utilized by Formspring technical team to count number of responses and active users (followers, following). Formspring is indeed getting benefit by the usage of Formspring which currently hosts 26 million users with 10 million average responses per day. [2]

14. Conclusion

Thus we presented the features of Cassandra distributed database management system and benefits of it using in real world enterprise applications. This database system is different from traditional RDBMS applications based on MySQL technology. It does not offer table joins but relevant functionality is offered by replicating rows across data table. From various use cases describes in this survey paper it is evident that Cassandra is excellent choice for businesses which are looking for features such as high availability, consistency, low downtime, fault tolerance, high scalability in terms of both users and data. These features are empirically tested by the creators of Cassandra on production data spread across geographically diverse areas with volumes in terms of several terabytes. With continuous development going on under Apache software foundation, Cassandra is expected to evolve as most prominent and powerful distributed database management system.

15. Acknowledgements

I would like to thank Prof. Judy Qiu, Assistant Professor of Computer Science and Informatics at School of Informatics and Computing at Indiana University for providing us with an idea to write a survey paper on one of the many areas of Distributed System. I also appreciate the help from Ila Jogaikar, an Associate Instructor for her valuable ideas throughout the design and development of this survey paper. Also thanks to ACM for providing this template to modify.

16. References

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

[2] http://en.wikipedia.org/wiki/Apache_Cassandra

[3]http://www.ibm.com/developerworks/opensource/library/os-apache-cassandra/index.html

[4] http://www.quora.com/Cassandra-database

[5] http://www.slideshare.net

[6] http://www.odbms.org/download/WP-DataStax-Cassandra.pdf

[7]Avinash Lakshman, Prashant Malik Cassandra-A Decentralized Structured Storage System, ACM SIGOPS Operating Systems Review archive, Volume 44 Issue 2, April 2010

[8] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung - The Google File system, ACM SIGOPS Operating Systems Review - SOSP '03 Homepage Volume 37 Issue 5, December 2003