Data Storage and Management

Project

On

Analysis of NoSQL Databases- Cassandra and HBase

Punit Lohani

x18127339

MSc Data Analytics 2018/19

Submitted to : Muhammad Iqbal

Abstract

The amount of data that are being used by the firms are continuously increasing at a faster pace and such type of data is commonly referred to as big data (Kota Tsuyuzaki, 2019). For such a big data storage and retrieval there is a requirement for the advanced technologies so as to serve the users in the best possible way. For this NOSQL databases have gained the popularity and attention for the effective and efficient way for the data storage and retrieval. To overcome the limitations of existing databases like MYSQL the firms are adopting the NOSQL databases as they are behaving like a backbone for the data management. Hbase and Cassandra are the popular NOSQL databases and in this project the comparison between their performances and the results are the primary objective. In order to proceed the platform has been made by installation of the respective databases along with the required configuration files for the environment where the tests can be performed.

Introduction

As the amount of data is exponentially increasing with time so there is a high requirement from the company's point of view to manage it in the best possible way so as to serve the customers with the effective and efficient retrieval of data. Earlier the data was managed by the firms with the help of relational database management system but with the introduction in the advancement of the internet technologies the task of handling the data became arduous and this leads to the invention of the NoSql that proves to be helpful in overcoming the limitations of the existing database. NoSQL database systems are distributed as well as non relational that are used for processing as well storage of enormous data across multiple servers. Strong consistency, high availability and partition tolerance are the some of the important characteristics of NoSql databases (Hossain, 2013). Strong consistency enables the users to view the same form of data despite of the updates being performed. High availability ensures that the data can be made available even when the sudden failure occurs in the nodes. And partition tolerance ensures that the system will continue to work as per the expectation in case of any failure related to the network. These important characteristics attracts the attention and contributes to the adoption by the firms across worldwide.

Four types of NOSQL database includes:

- **Key value store:** Key value store makes the use of hash table in which for a particular data there is an existence of unique key and the pointer. Example of such database includes Amazon S3.

- **Document Store**: This type of database is useful in storing each record and the corresponding data in a single document. The data can be encoded using XML, Jason etc. Example of such database includes Apache CouchDB (Kumar)

- **Graph based:** In this type of database the nodes and the edges are used for the storage and representation of data. The transformation of data from one model to another can be made with an ease with such type of database. Example of graph based is Neo4j.

- **Column based store:** In this type of database the data is stored in the columns which ensures faster searching and accessing of the data. Hbase and Cassandra are the popular examples of column based store database.

Key Characteristics

Hbase

Hbase is an column oriented database and it is scaled over hadoop. The architecture of Hbase is resemblant to that of Googles Bigtable. Hbase is used for swiftly accessing and storing the huge volume of data. (eduonix, 2016)

Some of the Key features of Hbase are mentioned below:

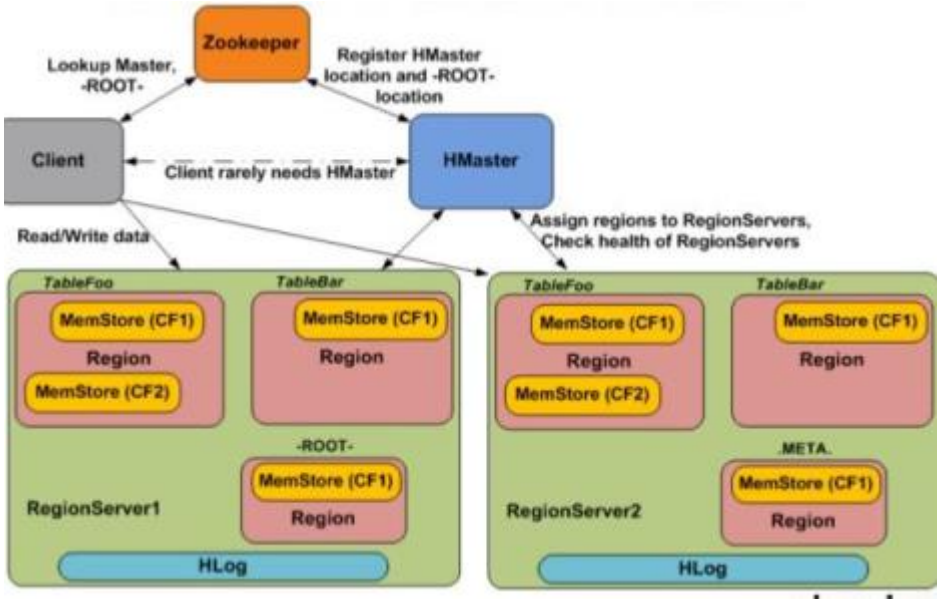
Whenever there is a requirement of high speed then Hbase is very useful as it provides the consistency in read as well as the write operations.

- Hbase provides the consistency between the read as well the write operations and this feature can be widely used where the speed requirement is the basic necessity.
- Hbase is ideal to store the semi as well as unstructured data.
- Hbase ensures the high availability of data by ensuring that even if the nodes in between encounters the failure, the system will work as per the expectation.
- It supports the feature of sharding which means that the data can be splitted automatically.
- Provides the support of automatic failover which means that it enables the movement of application to standby server in case of major failure.
- It helps in retrieving the data within the appropriate timeframe.
- For the effective processing of data , Hbase support the high level language.
- Java API are also supported by the Hbase so as to help the clients in accessing the required data with ease.
- Hbase highly supports the backup feature.

Cassandra

- The first important feature of Cassandra is its high reliability.
- Robustness in Cassandra architecture is the another important feature as it supports peer to peer with an outstanding characteristics which attracts the customers
- The nodes can be added or deleted in the cluster without any hassles.
- The throughoput of Cassandra is excellent which plays a significant role during the read and write.
- In between, if any of the node gets failed then also the data retrieval is possible through the other nodes. (Flair, 2019)
- Data can be replicated across various data centers which contributes towards the higher chance of getting the backups and recovery is possible whenever required.
- Cassandra is the first choice for most of the firms because of its ability to ensure data storage and retrieval for huge amount of records on the daily basis.
- As compared to the other databases the data model of cassandra is column oriented in nature.
- Another important characteristic includes that Cassandra is schema free.

Hbase Architecture



HBase is a column oriented database as it stores the records in the form of columns. The column oriented data is very useful if the huge data is encountered as it helps in retrieving the data within the appropriate timeframe. (Sarpale, 2019)

The architecture of the HBase consists of a single master node which is the Hmaster and many regional servers. These regional servers are important for the purpose of serving the regions. The Hmaster is responsible for forwarding the request to the corresponding regional servers that are raised by the clients.

There are basically three components of the Hbase Architecture and these are mentioned below:

1) HMaster

   Hmaster which is also known as the master server is responsible for allocating the various regions to the region servers and for this process it takes the help of the zookeeper. For the proper balancing of loads across the regions, the region servers are assigned to each regions by the Hmaster. The Hmaster closely monitors the loads in every region and If the load is found higher in any them, then it is distributed and allocated to the region server that are not occupied in order to effectively manage the load. Whenever the client raises the new

requirement related with either change in the schema or data operations then it first reaches to the Hmaster and then the operations as well as taken actions are taking accordingly. Additionally, the other responsibilities of Hmaster include handling the failures, working on the creation as well as deletion operations.

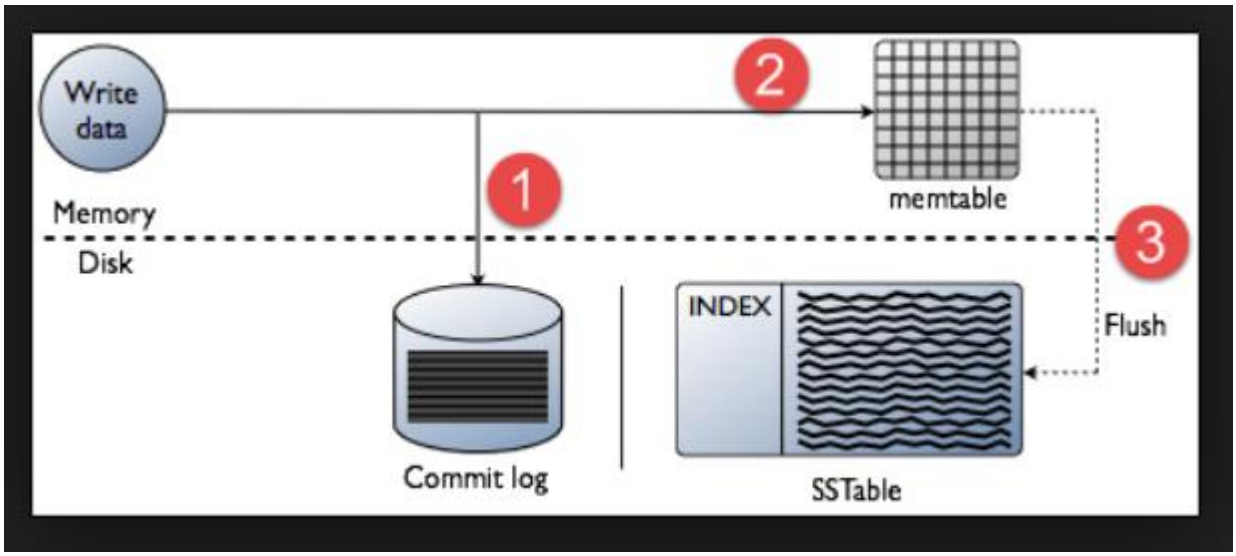
2) Region Server

The region servers are the working nodes that are responsible for handling the requests that are raised by the clients. These requests include read operations, write operations accompanied by updation and deletion of data. The region server components includes (janbasktraining, 2018):
-block cache generally contains the read data.
-MemStore takes care of writing cache and storage of the latest data that is being in the waiting list for the purpose of disk writing.
-Write Ahead Log is used for the storage of the new data.
-Hfile is the file that is used for row storage in the form of key values to the disk.

3) Zookeeper

Zookeeper is one of the important part of the Hbase architecture. Basically it acts as an interface between the client and the Hmaster by maintaining the proper information between them. The information that are related to regional servers such as information about the nodes and the availability are maintained by the zookeeper. The various services provided by the Zookeeper includes establishment of the communication between the client and the region servers. Additionally, it keeps a track on the failure that are encountered in the server.

Cassandra Architecture



In Cassandra the distribution of data is made across the nodes in the cluster. The nodes are interconnected and are independent of each other. The read and write requests are accepted by each nodes that are present in the cluster. If in between any of the node fails then also the read write operation are in active state as rest of the nodes continues the task. If the request for the data comes from the client's end and few nodes responded with the incorrect value then Cassandra detects and present the correct value to the client and in the back end it replaces the outdated value with the correct value. In the background the communication between the nodes takes place through the gossip protocol and it also helps in the detection of faulty nodes if encountered in the cluster. (tutorialspoint)

Some of the key components of the Cassandra are:

Node- The Cassandra data are stored in the nodes.

Data Center- Several data nodes makes the data center.

Cluster- Several data center makes the cluster.

Commit log- In Cassandra, Commit log is used to move the database back to the usable form.

Mem-table – After the commit log, in the Mem-table the temporal storage of data takes place.

SStable- When the data in the Mem-table reach the limit or becomes full, it is sent to SStable.

Bloom Filter- It is used after every query to quickly verify the identity of an element and perform check to figure out if the data belongs to a set or not.

Write operations- All the write operations in Cassandra are performed in the commit log and it is used in case of failure recovery. The data is then temporarily moved to the Mem-table and when the Mem-table becomes full then the data are sent to the SStable.

Read Operations- In the read operation as per the request, the mem table provides the value to Cassandra and then the check is performed to identify the correct ss table that is holding the requested data.

In addition there are two types of strategies in Cassandra i.e. simple strategy and network topology strategy. Presence of single data centers makes the simple strategy, while on the other hand presence of more than one data centers makes the network topology strategy.((guru99, 2019)).

Performance Test Plan

Creation of instance

The first step in order to proceed is the creation of instance for the two Nosql databases on the cloud service provided by the institution. For accessing the instance Putty has been used. For executing the test and installing purpose creation of hduser was created as a superuser.

Database

For the evaluation of the performance Cassandra and Hbase were installed.

Benchmarking tool

Yahoo Cloud service benchmark (YCSB) version 0.11.0 has been used for testing the performance of the two databases.All over there were the presence of six workloads out of which two workloads have been taken for the project i.e. workloada and workloadb.

Workloada: This workload mainly comprises of 50% read and 50% write operations. In total four test were performed for both Cassandra and Hbase.

Workloadb: This workload comprises of 95% read operation and 5% write operation. In total four test were performed for both Cassandra and Hbase.

In total four runs were performed for 50000,100000,200000,400000 respectively. The output result were recorded for the analysis purpose. For evaluating the performance of the database number of records and the type of workload were taken into consideration.

Evaluations and Results

Workload A and Workload B has been taken.

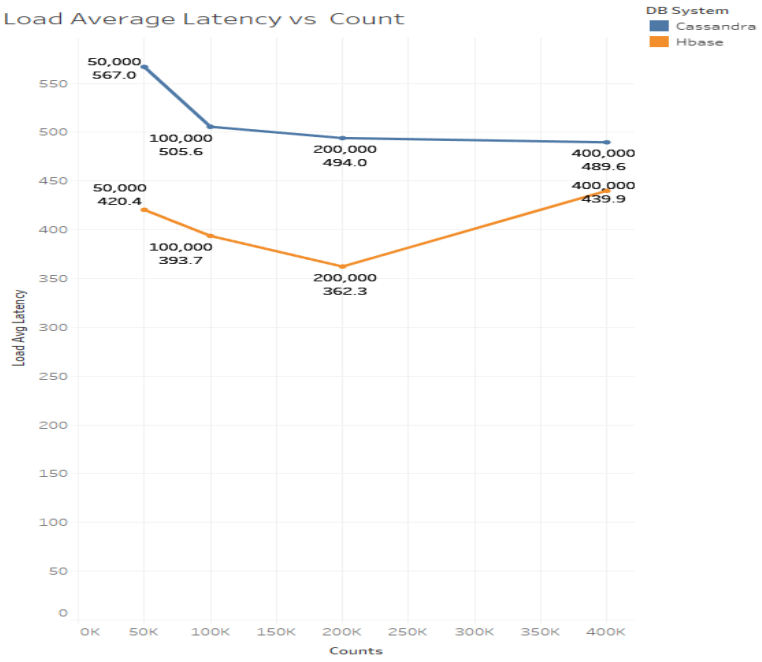
Database used are Cassandra and HBase.

The workload has been run for 50000,100000,200000,400000 counts.

Workload A results are shown below:

Load average latency vs Counts

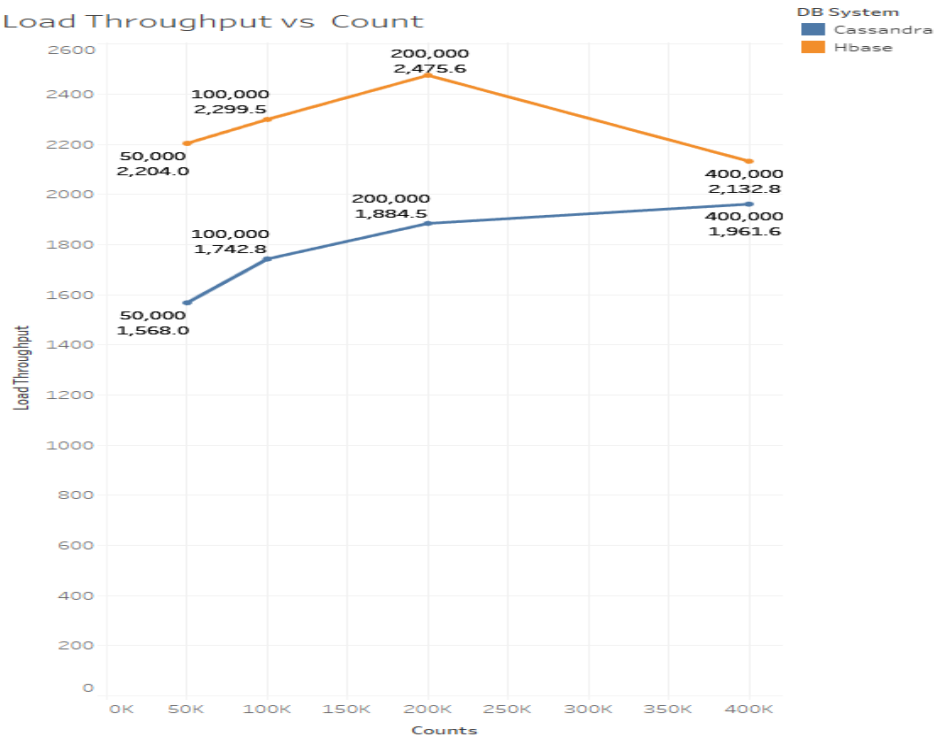
DB System	Workload	Counts	Load Avg Latency
Hbase	A	50000	420.43068
Hbase	A	100000	393.71528
Hbase	A	200000	362.28183
Hbase	A	400000	439.919765
Cassandra	A	50000	566.95966
Cassandra	A	100000	505.58498
Cassandra	A	200000	493.9707
Cassandra	A	400000	489.6322175



The above graph represents the comparison between the Load average latency and the Counts between the Cassandra and Hbase. It has been taken for the workload A. It can be seen that the load average latency in hBase is lower as compared to Cassandra.

Load Throughput vs count

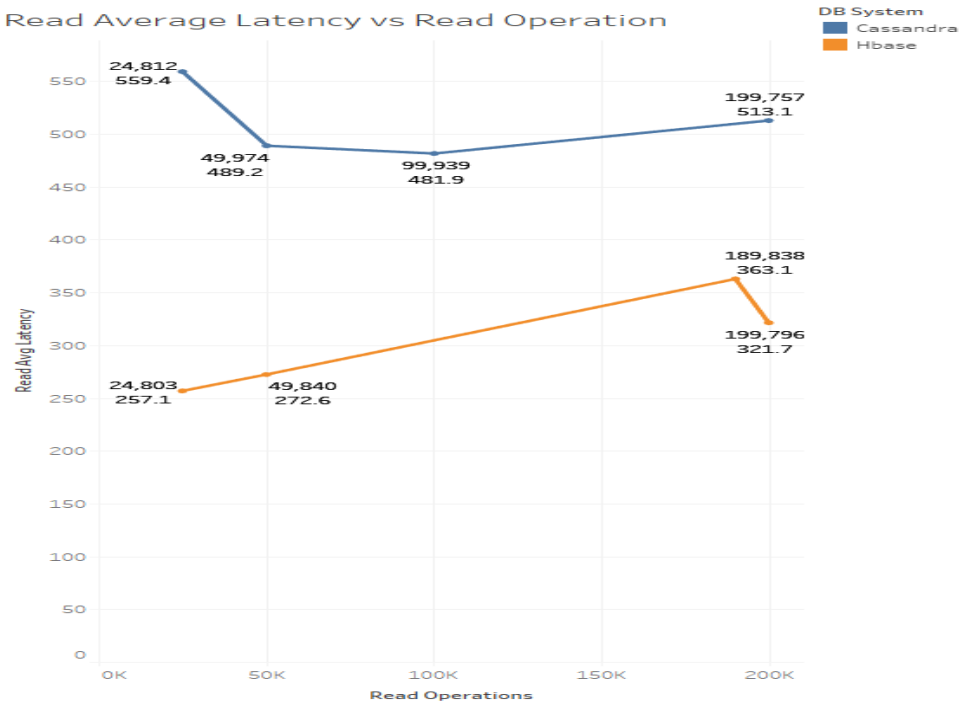
DB System	Workload T	Counts	Load Throughput
Hbase	A	50000	2204.002468
Hbase	A	100000	2299.537793
Hbase	A	200000	2475.645834
Hbase	A	400000	2132.798712
Cassandra	A	50000	1568.037131
Cassandra	A	100000	1742.828262
Cassandra	A	200000	1884.51681
Cassandra	A	400000	1961.630507



The above represents the comparison between the load throughput and counts between the Cassandra and Hbase. It can be seen from the graph that the load throughput of HBase is higher a compared with the Cassandra.

Read average latency vs Read operation

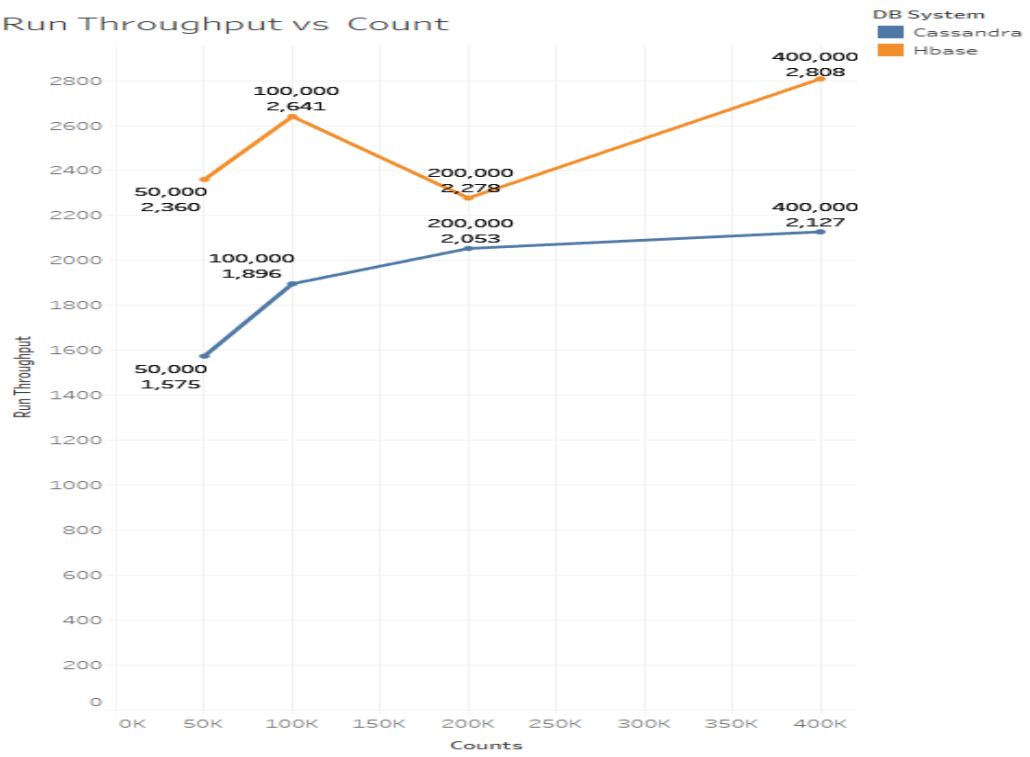
DB System	Workload T	Read Avg Latency	Read Operations
Hbase	A	257.0726525	24803
Hbase	A	272.5695225	49840
Hbase	A	363.0761386	189838
Hbase	A	321.6645478	199796
Cassandra	A	559.3831211	24812
Cassandra	A	489.2256373	49974
Cassandra	A	481.8724422	99939
Cassandra	A	513.0734492	199757



The above represents the comparison between the read average latency and read operation between the Cassandra and Hbase. In case of Hbase it can be seen that after the load of 400000 was applied there was lower read average latency while in case of Cassandra the lower read average latency can be observed on increasing the load.

Run throughput vs Counts

DB System	Workload Type	Run Throughput	Counts
Hbase	A	2360.383326	50000
Hbase	A	2640.682352	100000
Hbase	A	2277.670854	200000
Hbase	A	2808.002808	400000
Cassandra	A	1574.703956	50000
Cassandra	A	1896.381704	100000
Cassandra	A	2053.303766	200000
Cassandra	A	2127.478512	400000

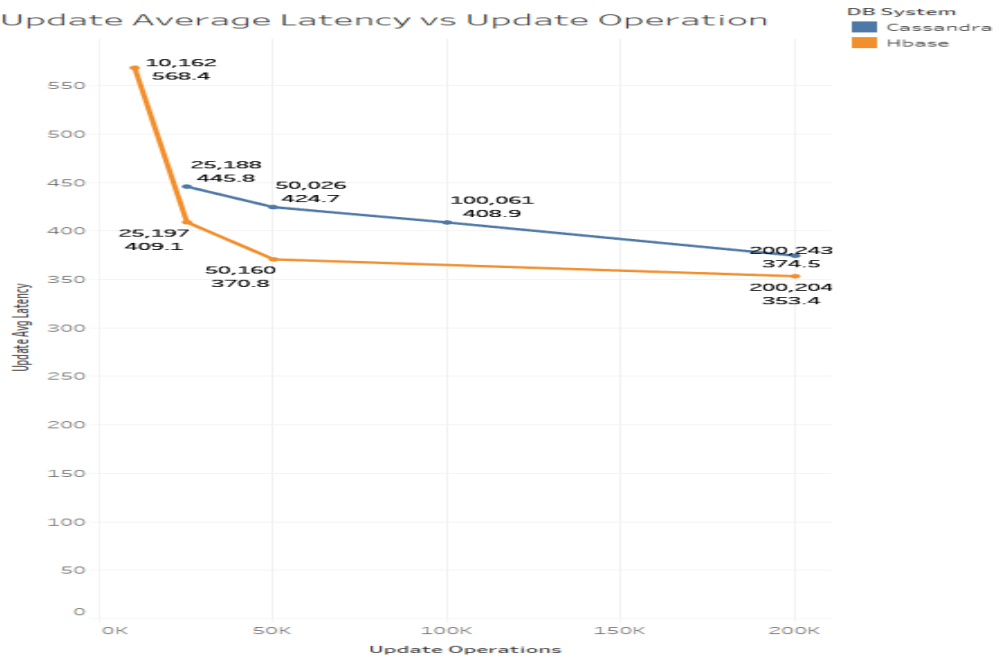


The above represents the comparison between the Run throughput and Counts

between the Cassandra and Hbase. This has been performed for the Workload A for both the databases.

Update Average latency vs Update operation

DB System	Workload Type	Counts	Update Avg Latency	Update Operations
Hbase	A	50000	409.1070366	25197
Hbase	A	100000	370.7563198	50160
Hbase	A	200000	568.3781736	10162
Hbase	A	400000	353.3540888	200204
Cassandra	A	50000	445.8298793	25188
Cassandra	A	100000	424.6649342	50026
Cassandra	A	200000	408.8640329	100061
Cassandra	A	400000	374.5189095	200243

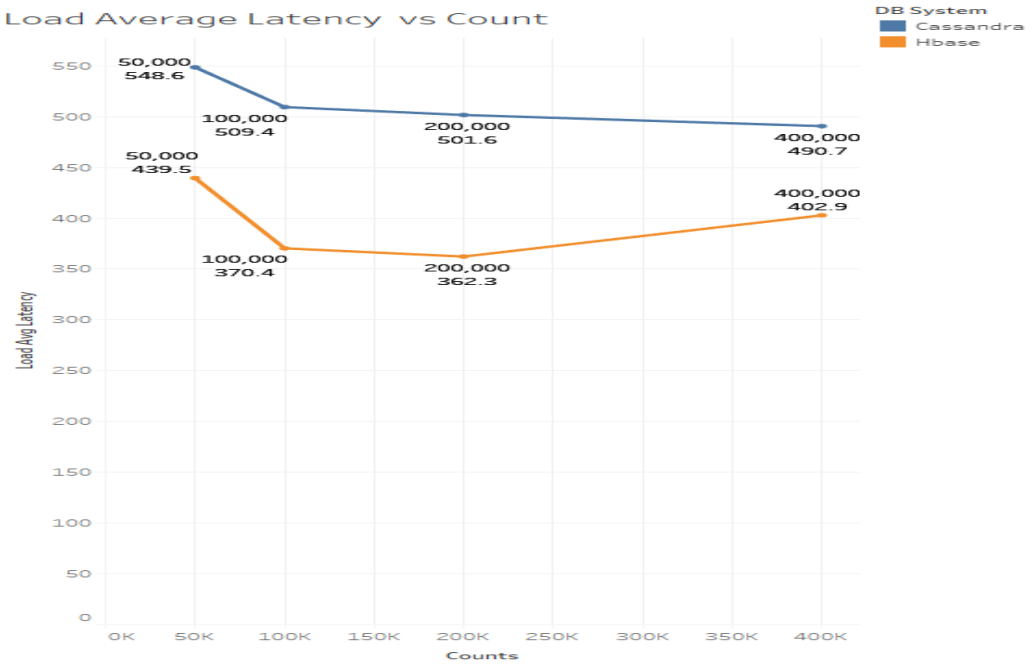


The above represents the comparison between Update Average latency vs Update operation between the Cassandra and Hbase.

Workload B result are represented below:

Load average latency and Counts

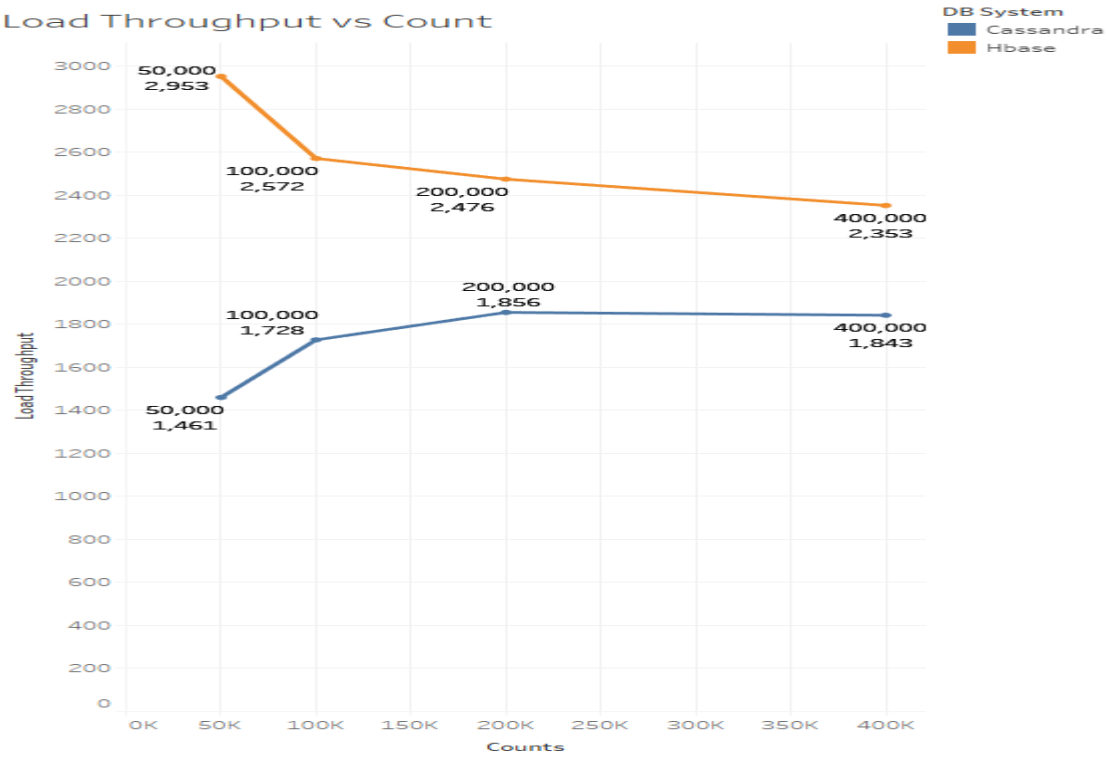
A	B	C	D
DB System	**Counts**	**Workload Type**	**Load Avg Latency**
Hbase	50000	B	439.53168
Hbase	100000	B	370.3759
Hbase	200000	B	362.28183
Hbase	400000	B	402.8651125
Cassandra	50000	B	548.55188
Cassandra	100000	B	509.39392
Cassandra	200000	B	501.615655
Cassandra	400000	B	490.6758502



The above graph represents the comparison between Load average latency vs Counts between the Cassandra and Hbase. It is representing that how the load average latency gets affected when the counts are increased.

Load throughput vs count

A	B	C	D
DB System	Counts	Workload Type	Load Throughput
Hbase	50000	B	2953.337271
Hbase	100000	B	2572.016461
Hbase	200000	B	2475.645834
Hbase	400000	B	2353.398013
Cassandra	50000	B	1460.792334
Cassandra	100000	B	1728.339584
Cassandra	200000	B	1856.045139
Cassandra	400000	B	1842.895363

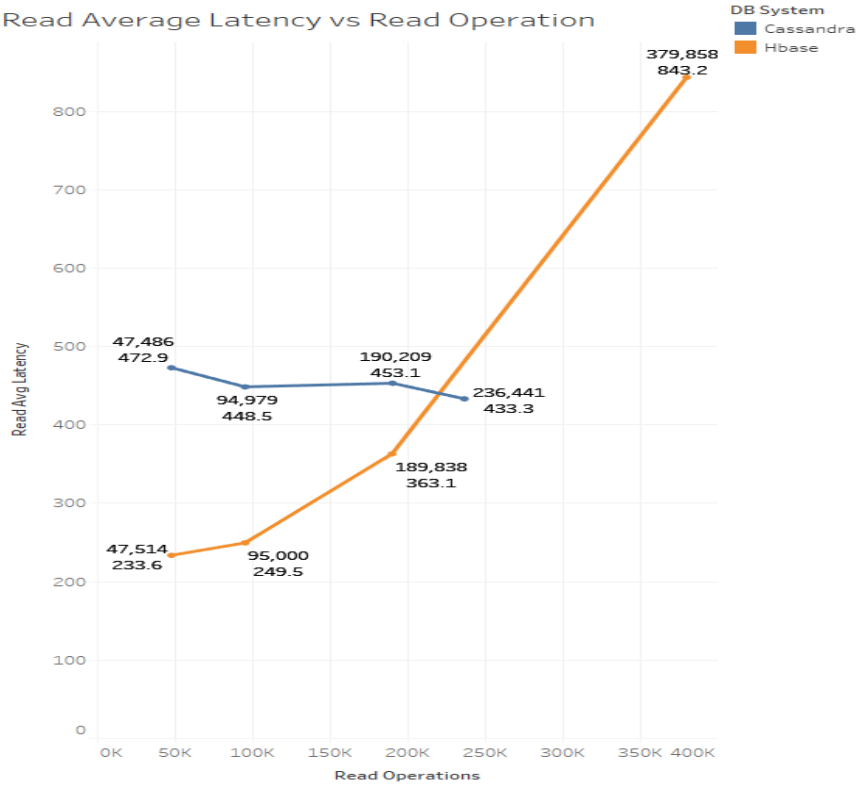


The above graph represents the comparison between Load throughput and counts

between the Cassandra and Hbase

Read average latency vs Read operation

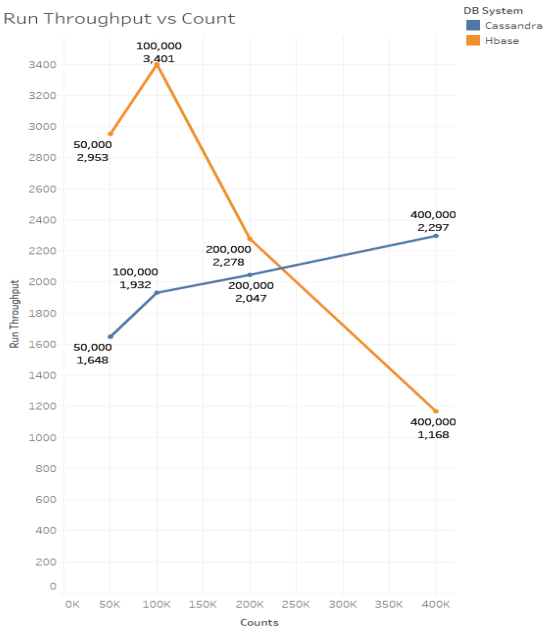
A	B	C	D	E
DB System	Counts	Workload Type	Read Avg Latency	Read Operations
Hbase	50000	B	233.643263	47514
Hbase	100000	B	249.5337684	95000
Hbase	200000	B	363.0761386	189838
Hbase	400000	B	843.2209931	379858
Cassandra	50000	B	472.8901992	47486
Cassandra	100000	B	448.4824119	94979
Cassandra	200000	B	453.0877298	190209
Cassandra	400000	B	433.2805013	236441



The above graph represents the comparison between Read average latency and Read operation between the Cassandra and Hbase. In case of Hbase as the number of counts increases, the read average latency is also increasing, while in case of Cassandra with the increment in the count, the read average latency is descreasing.

Run throughput vs count

A	B	C	D
DB System	**Counts**	**Workload Type**	**Run Throughput**
Hbase	50000	B	2953.337271
Hbase	100000	B	3400.666531
Hbase	200000	B	2277.670854
Hbase	400000	B	1168.476832
Cassandra	50000	B	1648.315422
Cassandra	100000	B	1931.546009
Cassandra	200000	B	2046.726772
Cassandra	400000	B	2296.923271

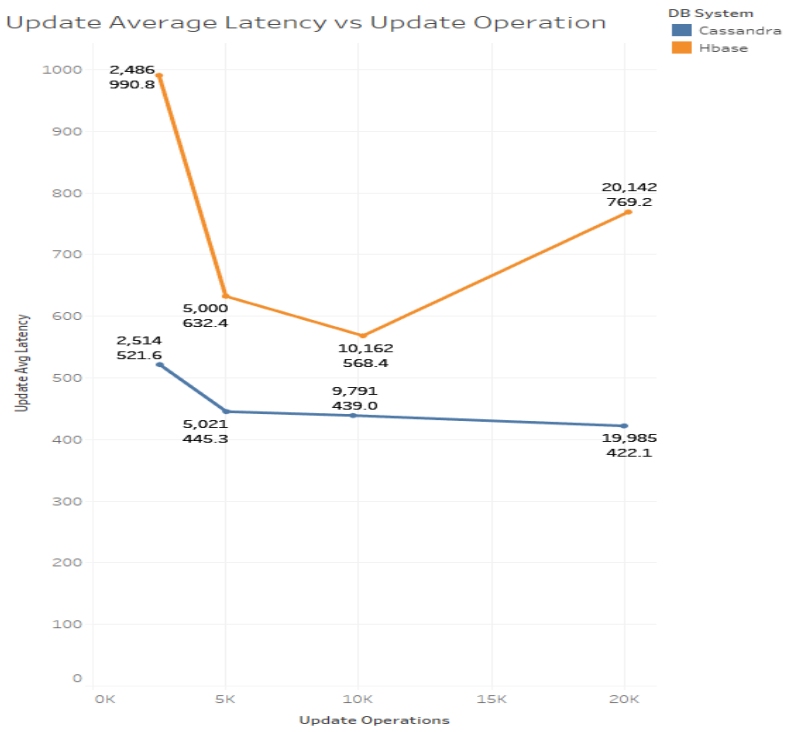


The above graph represents the comparison between Run throughput and count

between the Cassandra and Hbase. For Cassandra it can be seen that with the increase in the counts the the run throughput is increasing. While in case of HBase the when the count in increased , there can be seen the drop in the run throughput.

Update average latency vs update operation

A	B	C	D	E
DB System	Counts	Workload Type	Update Avg Latency	Update Operations
Hbase	50000	B	990.8246179	2486
Hbase	100000	B	632.4308	5000
Hbase	200000	B	568.3781736	10162
Hbase	400000	B	769.2061861	20142
Cassandra	50000	B	521.6284805	2514
Cassandra	100000	B	445.3164708	5021
Cassandra	200000	B	438.9945869	9791
Cassandra	400000	B	422.1370028	19985



The above graph represents the comparison between Update average latency and update operation between the Cassandra and Hbase. In case of Cassandra, with the increase in the counts, the update average latency is decreasing while the combination can be observed in case of HBase.

Database Systems Capabilities

1)Security of Hbase and Cassandra

1) Cassandra security features

Below are some of the authentication security features that are provided by the Cassandra:

- Client can authenticate with the database by the help of the various types of security protocols. In cassandra such type of protocols are included eg; LDAP, Kerbos and SSL . Hence this feature is supported by this database
- Methods for the authentication of servers are supported by the Cassandra that includes the servers communicating with each by ensuring the security of the environment.
- In Cassandra the account details of the database are stored either in the database or in the file which ensures that storing of the credentials are taking place in the secured manner. Hence this database supports this feature.
- Role based security ensures that the database is accessed by the authorized person and through authorized login. This feature is also supported by the Cassandra.
- For the highly sensitive domains encryption of database is very important and in this the database is made encrypted for a particular disk. This feature is supported by Cassandra.
- In order to make sure that the logs does not consume high amount of space the feature of flushing the log plays an important role. This feature is supported by Cassandra. (quabase, 2015)

2) Hbase security feature (quabase, quabase, 2015)

- Client authentication with the database is very important aspect and this feature is available in Hbase in the form of kerbos and SSL.
- To make sure the security with respect to the environment in which the operations are being performed it is important for the various database servers to make an authentication with one another in the distributed database. This feature is supported by the Hbase, eg shared key file.
- The account details must be stored in a secure manner by the database and this feature is provided by Hbase .
- Role based security ensures that the database is accessed by the authorized person and through authorized login. This feature is supported by the Hbase. Default and custom roles are some of the role based options that are provided by Hbase.

- For the highly sensitive domains encryption of datybase is very important and in this the database is made encrypted for a particular disk. This feature is supported by Hbase.
- In order to make sure that the logs does not consume high amount of space the feature of flushing the log plays an important role. This feature is supported in Hbase. This is done using configurable event logging and fixed event logging.

2) Transaction Management

Cassandra

ACID transaction along with rollback and locking mechanism is not supported by Cassandra but surely offers atomic, isolated and durable tractions. (Rana, 2017)

Since Cassandra is NoSQL database therefore it does not support joins and foreign key references.

Isolation, Atomicity, Durabilty are the key aspects of Cassandra.

Write operation is atomic in Cassandra which means that insertion or updates for 2 or more rows into the same partition are treated as in one write operation, whereas deletion and writing operations are performed with row level isolations.

Finally writes in Cassandra are durable because of their node mechanism as data is stored on multiple node which will help in prevention from data loss, even if due to some hardware issue or system crash there will no loss of data as before any failure mem-table are flushed into the disk and commit log is replayed on restart to recovery any lost writes.

Hbase

HBase supports ACID properties of RDBMS in a limited way ,it has no mixed read/write transactions.

In Hbase whenever the write transaction initiated ,it will retrieve the next highest transaction number which is referred as Write number whereas whenever a read transaction is initiated, it will retrieve the transaction of last committed transaction which is referred as read point. (Hbase, 2019)

Key value pair created with its transaction are called as memstore timestamp in HBase.

However, on comparison with the RDBMS, Hbase does nt exactly support the transactions that are ACID compliant when it comes to the point that the rows are multiple and comparing across the tables.

RELATED RESEARCH

NoSQL database systems are distributed as well as non relational that are used for processing and storing the huge and enormous amount of data across several servers. Strong consistency, high availability and partition tolerance are the some of the important characteristics of NoSql databases (Hossain, 2013). These characteristics are very important because they makes the NoSQL database advantageous as compared to the existing databases. There are many types of NoSQL databases , here as a part of the project, Cassandra and Hbase have been taken into the consideration. Cassandra is basically useful in managing the large amount of data that is structured in nature. High throughout can be easily supported by Cassandra , moreover providing the excellent performance and the scalability. (Lakshman, 2009)

In this research the author has taken YCSB-0.5.0 version for the use. As YCSB comes with six different workloads from A to F. Here in the analysis A,B,and C have been taken. The creation of the in the experiment has been created by YCSB. In total 1 million records were used. Each records in the field contains 10 fields and the size of the record was 100 bytes. Out of the databases taken each of the database had 1 milllion records and the comparison have been made between the throughput and the average latency. While taking the workload A, the Cassandra proves to better than Hbase while comparing the throughput. The latency of Hbase was lower average latency which denotes that the Hbase performs well when it comes to higher loads. Because of the write optimized design of the Hbase the overall performance proved to be better than the Cassandra.

Now in another run, workload b has been taken , in this scenario the average latency as well the throughput of Hbase was found to be higher. Here the Cassandra performance was found to be good. Now on taking the workoad c, Cassandra peformed beter than Hbase. So on a concluding note it was observed that the Cassandra and Hbase both works well under the influence of heavy loads because of their optimized design, moreover while perfoming the under the read operation the performance of Hbase was found to be less satisfactory as compared with Cassandra. (Saran, 2016)

YCSB was developed by Yahoo is used when the databases needs to be tested. YCSB is quite helpful when there is a need to compare and test the multiple databases. As a part of the initial setup for the experiment Before the collection of value for response time and throughput the YCSB was ran around 20 times. The database were tested on the basis of number of nodes,cores and replication factor.

As a part of the experiment, both the Cassandra and hbase were hosted on the different machines the number of cores taken for Cassandra and hbase were 8 and 8 respectively and the throughput were 10

and 22 ops/sec respectively. As per the initial findings it is seen that the Hbase took the advantage of having the number of core on comparing with the others. Cassandra was slower than Hbase.

The comparison shown between the databases includes the number of nodes, replication factor and the throughput. The number of nodes for Hbase and Cassandra were 9 and the replication factor increased from 1 to 3. The replication factor equal to 1 signifies that in the cluster, corresponding to nodes there is only a single data. As the replication factor increased, the copies will be created. As the copies are increased this leads to the degradation to increase. On evaluating the performance, casssandra showed the behavior that performance was not decreased even when the increment in the replica took place. However there was no major impact seen in the case of HBase because of its architecture. As per the research conclusion, HBase and Cassandra were found to show the positive performance as compared to the other database. (Mary, 2017)

Conclusion and Discussion

On a concluding note it can said that the experiments have been carried out by taking two of the most impressive Nosql databases: Cassandra and Hbase by comparing the load throughput, load average latency, run throughput, read average latency, update average latency, read operations, update operations. The architecture as well as the discussion about the two databases have been discussed by keeping various factors into consideration. Yahoo Cloud service benchmark (YCSB) version 0.11.0 has been used for testing the performance of the two databases. All over there were the presence of six workloads out of which two workloads have been taken for the project i.e. workload A and workload B and the same have been illustrated above. After performing the research it can be seen that both Cassandra and HBase represented the different characteristics when the counts are increased from 50000 to 400000. Hence it can be observed that HBase and Cassandra

As for the future work point of view the experiment can be performed by taking higher data counts as well as changing the workloads which have been taken here. The experiment can be performed by taking other workloads from YCSB which can provide more informative result so as to compare the performance of these databases in an effective and efficient manner.

Bibliography

eduonix. (2016, 04 01). *eduonix*. Retrieved 04 22, 2019, from eduonix: https://blog.eduonix.com/bigdata-and-hadoop/use-hbase-nosql-db/

Flair, D. (2019). *Data Flair*. Retrieved 04 22, 2019, from Data Flair: https://data-flair.training/blogs/cassandra-features/

guru99. (2019). *guru99*. Retrieved 04 22, 2019, from guru99: https://www.guru99.com/cassandra-architecture.html

Hbase, A. (2019). *Apache Hbase*. Retrieved 04 22, 2019, from Apache Hbase: https://hbase.apache.org/acid-semantics.html

Hossain, A. B. (2013). *NoSQL Database: New Era of Databases for Big data Analytics- Classification, Characteristics and Comparison.* Daffodil: International Journal of Database Theory and Application.

janbasktraining. (2018). *janbasktraining*. Retrieved 04 22, 2019, from janbasktraining: https://www.janbasktraining.com/blog/hbase-architecture-main-server-components/

Kota Tsuyuzaki, M. O. (2019). *NTT*. Retrieved 04 22, 2019, from NTT: https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201212fa3_s.html

Kumar, G. (n.d.). *3pillarglobal*. Retrieved 04 22, 2019, from 3pillarglobal: https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases

Lakshman, P. M. (2009). *Cassandra - A Decentralized Structured Storage System.* https://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf.

Mary, R. K. (2017). *Comparative Performance Analysis of various NoSQL Databases: MongoDB,Cassandra and Hbase on Yahoo CLoud Server.* Bangalore: Imperial Journal of Interdisciplinary Research.

quabase. (2015). *quabase*. Retrieved 04 22, 2019, from quabase: https://quabase.sei.cmu.edu/mediawiki/index.php/Cassandra_Security_Features

quabase. (2015). *quabase*. Retrieved 04 22, 2019, from quabase: https://quabase.sei.cmu.edu/mediawiki/index.php/HBase_Security_Features

Rana, P. (2017, 02 01). *knoldus*. Retrieved 04 22, 2019, from knoldus: https://blog.knoldus.com/transaction-management-in-cassandra/

Saran, A. H. (2016). *A Comparison Of NoSQL Database Systems: A Study On MongoDB,Apache Hbase, And Apache.* Turkey: International Conference on Computer Science and Engineering.

Sarpale, G. (2019, 02). *researchgate*. Retrieved 04 22, 2019, from researchgate: https://www.researchgate.net/publication/330937764_A_Comparison_Of_NoSQL_Database_Systems_A_Study_On_MongoDB_and_Apache_HBase

tutorialspoint. (n.d.). *tutorialspoint*. Retrieved 04 22, 2019, from tutorialspoint: https://www.tutorialspoint.com/cassandra/cassandra_architecture.htm