Data Storage and Management

Project

X18129633

Div A

Sumit Jadhav

Table of Contents

Abstract:

There are two types of databases the traditional RDBMS and the NoSQL. These types of databases are used to store high volume data. These databases are being used by many applications like Google maps, Google book search Gmail to store the data. This project is based on the two most popular NoSQL databases which are HBase and Cassandra. This report is based on the analysis of the comparison and visualization of the benchmark test performed using YCSB framework.

Introduction:

To manage the vast amount of data from various sources such as social network, research, and development sector as NASA, the experiment based on hydron collider and the recent findings of black hole images are generated. This led to the rise in use of NoSQL database. Most of the database application use NoSQL for their big data storage option. These storages are built with the features which will overcome the drawbacks of the previous traditional RDBMS. The performance of these can be tested with the help of CRUD operations which are (Create, Read, Update, Delete). For this specific project, we have used Yahoo! Cloud Service Benchmark which is also known as YCSB, to make the formulation of databases like HBase and Cassandra. A test was conducted for the benchmarking the scalability of both of these databases under workloads A and B.

Key Characteristics of HBase:

1. **Atomic Read and Write**:

   When the system is running a read or writes process then the rest of the operation of reading and writing is prevented. This process is called atomic read and write. Atomic read and write operation are offered by HBase are on a low level.

2. **Atomic Sharding**:

An HBase table is made is regions and is handled by region serves, and these servers are distributed throughout the data nodes. HBase gives us an automatic and manual splitting of these regions to smaller regions and when the threshold size is meet then it reduces I/O time and overhead.

3. **Consistency**:

   HBase is proven to be very consistent in its performance. The semantics of ACID is followed by HBase. This has an outcome over the write performance and HBase is focused on high read performance.

4. **Rowkeys**:

   There are mainly three primary Rowkeys in HBase which are: Get, Put, Scan. In this data is prioritized and retrieved in a very smooth fashion. The application pattern is coherent with results in higher read performance. When the rows go under scan the rows are arranged alphabetically as per their respective Rowkeys. By this method in a single database, all the related and applicable database are collected.

5. **Scalability**:

   HBase provides the service of scalability. HBase supports scalability in linear and modular form. We can say that HBase is linearly scalable.

Key Characteristics of Cassandra:

1. **Peer to Peer Architecture**:

   Some databases work on master-slave architecture in Cassandra and some work on peer to peer architecture. In peer to peer architecture, several

units are able to communicate with each other. Which leads us to an advantage of no single point of failure. Cassandra has robust architecture and with some exceptional characteristics.

2. **Elastic Scalability**:

   With this elastic scalability, we can easily scale up or scale down the desired cluster. There is integrated flexibility of adding and deleting any size of nodes from the desired cluster with no disturbance. Because of this Cassandra has a very high throughput and also have the highest number of nodes.

3. **High Availability and Fault Tolerance**:

   Cassandra has very high availability and fault tolerance due to having a ring structure, multiple data centres, and adjustable replication and because of this, we can retrieve data from any node.

4. **Column-Oriented**:

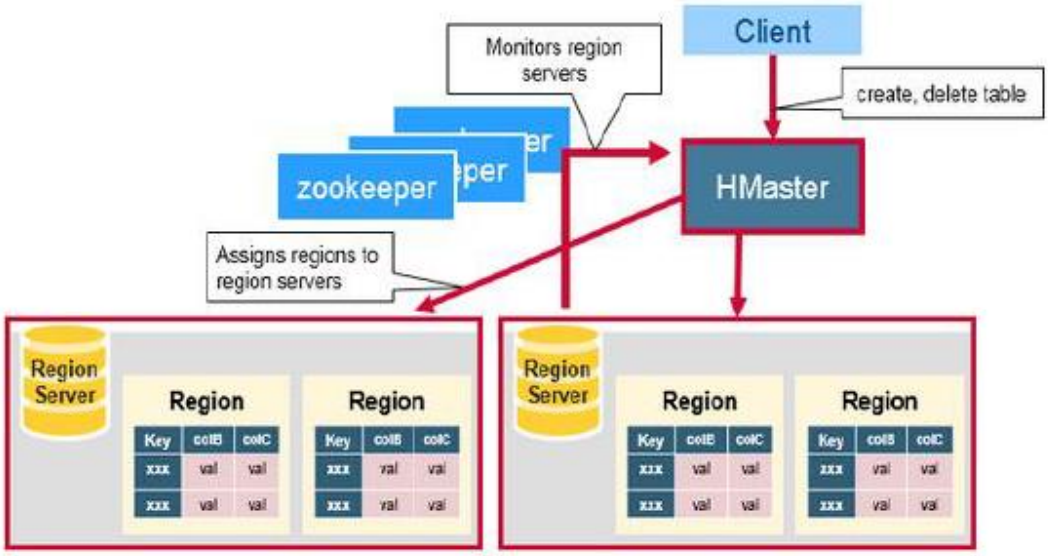
   Cassandra's data model is column oriented. In the Cassandra data model, the columns are stored according to their respective column names. Therefore, there is a number of columns are present in a row.

5. **Open Source**:

   Cassandra being more powerful and reliable it is available for free to users. Therefore, Cassandra could be integrated with many open source projects like Hive, Pig, Hadoop.

Database Architecture:

HBase Architecture:



HBase has mainly three components such as:

a. HMaster
b. Regions
c. Zookeeper

HBase provides low latency random read and writes operations on the top of the HDFS system. In HBase tables are dynamically distributed by the system whenever it becomes too large to handle. The most simple and foundational unit is HBase is region. HBase architecture has a single master node and multiple slave nodes.

a. HMaster:

HMaster is a lightweight process which handles and assigns regions to the region server in the cluster of Hadoop for the main purpose of load balancing. Creation of column family in regards to the table is some of the metadata operations for which the HMaster node is accountable.

There are several responsibilities of HMaster such as:

i. It monitors and manages the Hadoop cluster.
ii. It helps to perform administrations of CRUD operations such as providing an interface for creating, updating and deleting the tables.
iii. Operations such as DDL and also handled by the HMaster node.
iv. Whenever the desirable change is required by the client in the cluster these operations are also handled by the HMaster node.

b. Regions:

Each node present in the Hadoop cluster is executed by the region server process. The tables which are present widely throughout the region are known as regions. Functions related to data are handled by the regions only. Also, communication with the client is also done by the regions.

c. Zookeeper:

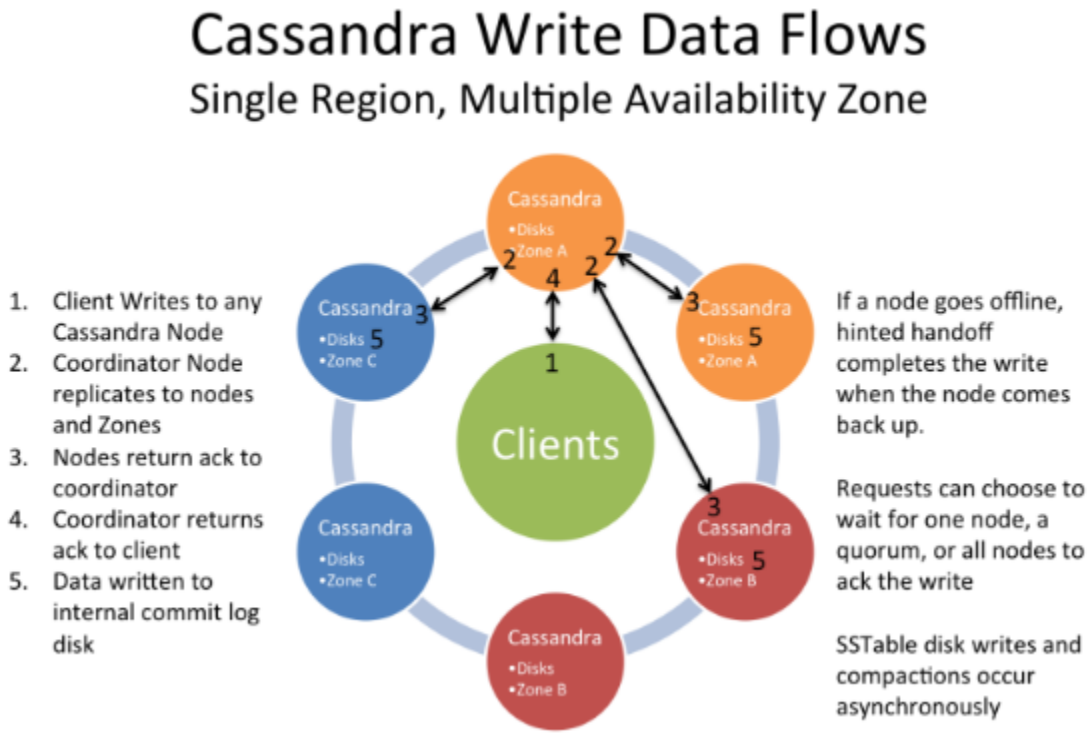
Zookeeper is the centralized monitoring server which performs operations like maintaining the configuration information and also provides the distributed synchronization. HMaster and region server is mainly registered with the Zookeeper. Client strictly needs access to Zookeeper if it wants to connect to region server and HMaster. In the case of node, failure Zookeeper triggers the error message and begins to repair the nodes.

Zookeeper also performs additional operations such as:

i. Tracking the server failures and breakdown

ii. Tracking network partitions.
iii. Establishing communication channels with a region server.
iv. It also provides ephemeral nodes.

Cassandra Architecture:



The architecture of Cassandra is typically based on the considerations of system and hardware failure. Cassandra is made to handle big data workloads across multiple nodes. Cassandra consists of peer to peer distributed across its nodes and it is consistently distributed among all the other nodes. As all the nodes present in the cluster has the same role and every node is independent and interconnected with each other at the same time.

Each node present in the cluster has is able to accept read and write requests whether the data is actually located in the cluster or not. Whenever a node goes down the read and writes operation from that node are stopped.

In Cassandra the node act as a replica for a given piece of data. If one of the nodes returns as out of date value then Cassandra will return the most recent value available to it. Cassandra also uses gossip protocol in the background which allows communication between the nodes.

Comparison Between the Cassandra and HBase for scalability, availability, and reliability.

HBase:

1. HBase is known for its linear scalability.
2. HBase also compromises on its availability.
3. HBase provides a high degree of reliability.

Cassandra:

1. Cassandra is highly available as it has the replication property which can handle node failure.
2. As the data is presented in a replicated format the most recent value is sent by Cassandra in case of the out of date value is detected.
3. Cassandra is made in the consideration of hardware failure. So whenever a node failure occurs there is no loss of data, as well as the repair of the failed node, is called.

Security measure For HBase:

Initially, Hadoop was developed without the measure of security, user authentication, and data privacy model. This was the main reason for poor security measures in Hadoop. Eventually, security measures like authentication and authorization were applied but this wasn't enough to make the system secured. HBase is not very strong to defend on it own hence it is weak against injection attacks.

The architecture of HBase is a master-slave and because of it, the recovery time of the affected node is very time-consuming. As HBase consists of region server high-level monitoring is very difficult as the region server supports macro monitoring. Also, HBase does not support exception handling as well so it is more prone to a single point of failure.

On the positive side, HBase has authentication layers as token-based authentication and HBase uses Kerberos for authentication purposes. An encryption algorithm is also used for the protection of data as well.

Security Measures for Cassandra:

1. **Authentication based on internally username/passwords**:

   Cassandra authentication is role-based which means that roles can be with superuser, non-superuser, and login privileges. The internal authentication is used to authenticate the user privileges which grants access to create keyspace and Tables.

2. **SSL Encryption**:

   Cassandra provides a secured connection between the user and the cluster as well as between nodes and the database cluster. Enabling SSL encryption helps us to encrypt the data and transfers securely without any compromise.

3. **Authentication and authorization based on JMX username and password**:

   Java management extension provides gives standard gateway of managing the resources. By this, we can manage the permission of the roles and restrict their access. This type of authentication uses two different files, one for password and one for access.

Literature Survey:

NoSQL database are getting a standard fata platform for many applications which make a heavy use of internet and data. While being in the market situation we can observe that there are mainly used databases are Cassandra, HBase and Mongodb. NoSQL highly relies on distributed system.

The main purpose of using NoSQL is that it doesn't limit the storable data types. As HBase provide scalability options and can be worked with Hadoop it benefits the handling of big data easily.

Compared to RDBMS the structure of unstructured data makes it more reliable and make it possible to adapt with natural scalability. The execution speed of this databases can be increased by adding the higher configuration to the machines.

Basically, NoSQL works on basic availability, soft state and eventually consistent standards. [10]

A Benchmarking framework such as YCSB helps to perform benchmarking tests on distributed system. This benchmarking is done on the criteria of scalability, availability, performance and replication with respect to data. To perform these tests, we have to enter some kind of workload. These workloads have different analysis such as:

Workload A: This consists of 50 percent read and 50 percent update.
Workload B: This consists of 95 percent read and 05 percent update.
Workload c: This consists of 100 percent read.
Workload D: This reads only the latest version of the workload.
Workload E: This consists only of query records.
Workload F: This consists of reading and modifies records.

Performance and Test Plan:

The complete analysis is done on OpenStack instance with a virtual terminal session. The instance is accessed with a key pair which was generated at the time of instance.

Instance name : X18129633-CA4

IP address: 192.168.100.143

1. **Test Harness**:

   Test Harness has a test execution engine and test scripts. It's an automated process which performs multiple tests which defines the different test scenarios and compares their performances. The automated test present in test harness helps us to compute the performance and we can compare the

output values of the tests which are performed. Test Harness has several advantages such as:

a. Complete automation of test to be performed.
b. Test results generated are easy to read and easy to compare.
c. Increased productivity due to automation.
d. Supported debugging.

2. **Yahoo! Cloud Storage Benchmark**:

YCSB was used in this project to calculate the performance of workloads of NoSQL database such as HBase and Cassandra. YCSB was developed for benchmarking purposes of multiple systems which has the same configuration system as well as to able to compare the output values from the workloads.

YCSB has mainly 2 components:

i. YCSB client which is a workload generator.
ii. The workloads which generate and executes a set of workloads.

3. **OpenStack**:

This is a type of cloud operating system which provides a service of computation by web services channel. It is an open source platform which helps by virtual servers. An instance is created on the OpenStack platform which is accessible by a local terminal by ssh.

4. **Virtual Machine**:

The created instance was connected with a virtual machine using proxy SSH command.

5. **Databases**:

   Cassandra and HBase were used to perform the performance tests. Analysis of the performance of this database is compared after when the tests are performed.

6. **Benchmarking**:

   The benchmark tests were done on Yahoo! Cloud Storage Benchmark platform in the mentioned open stack instance. To check the performance testing workloads A and B were used. There are 6 different types of workloads A to E.

   Workload A: This consists of 50 percent read and 50 percent update.
   Workload B: This consists of 95 percent read and 05 percent update.
   Workload c: This consists of 100 percent read.
   Workload D: This reads only the latest version of the workload.
   Workload E: This consists only of query records.
   Workload F: This consists of reading and modifies records.

7. **Operational Counts**:

   We have considered 5 operational counts for testing the above-mentioned database (HBase and Cassandra) such as:

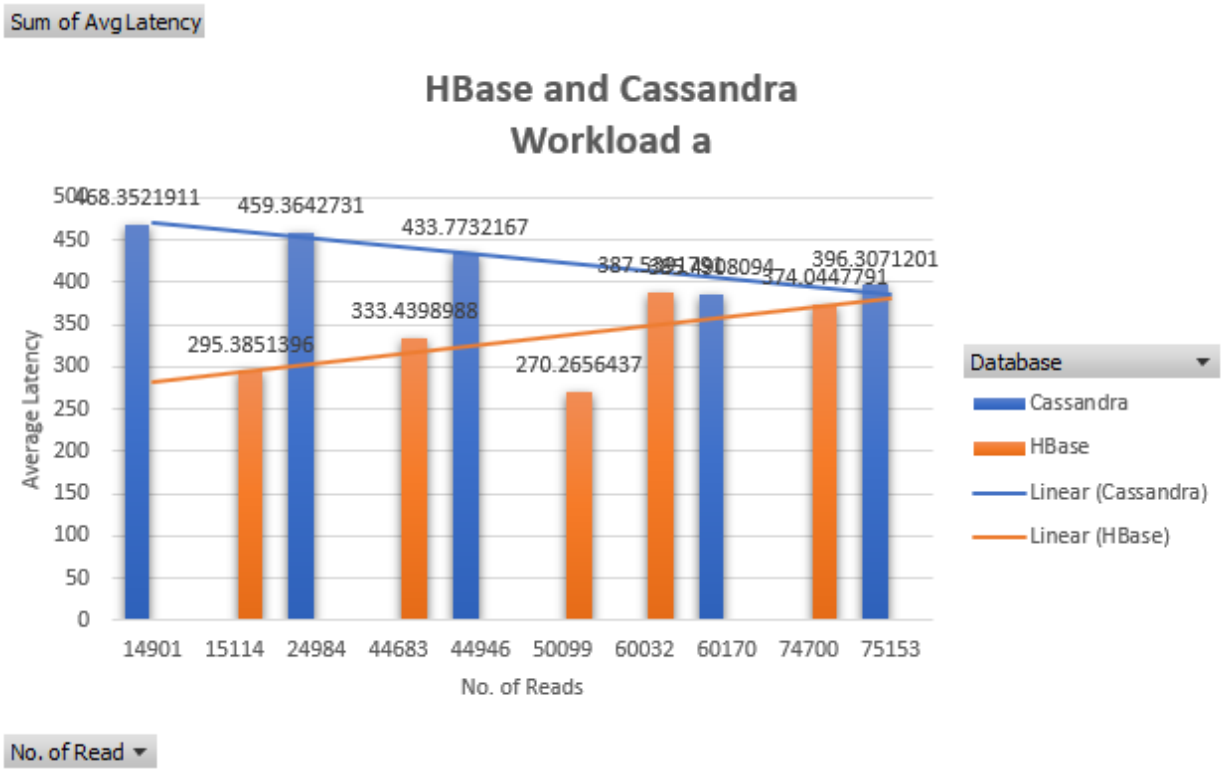
   - 30000
   - 50000
   - 90000
   - 120000
   - 150000

   Each of the outputs was tested for both workloads A and Band the average output is taken for the performance analysis.

Evaluation and Results:

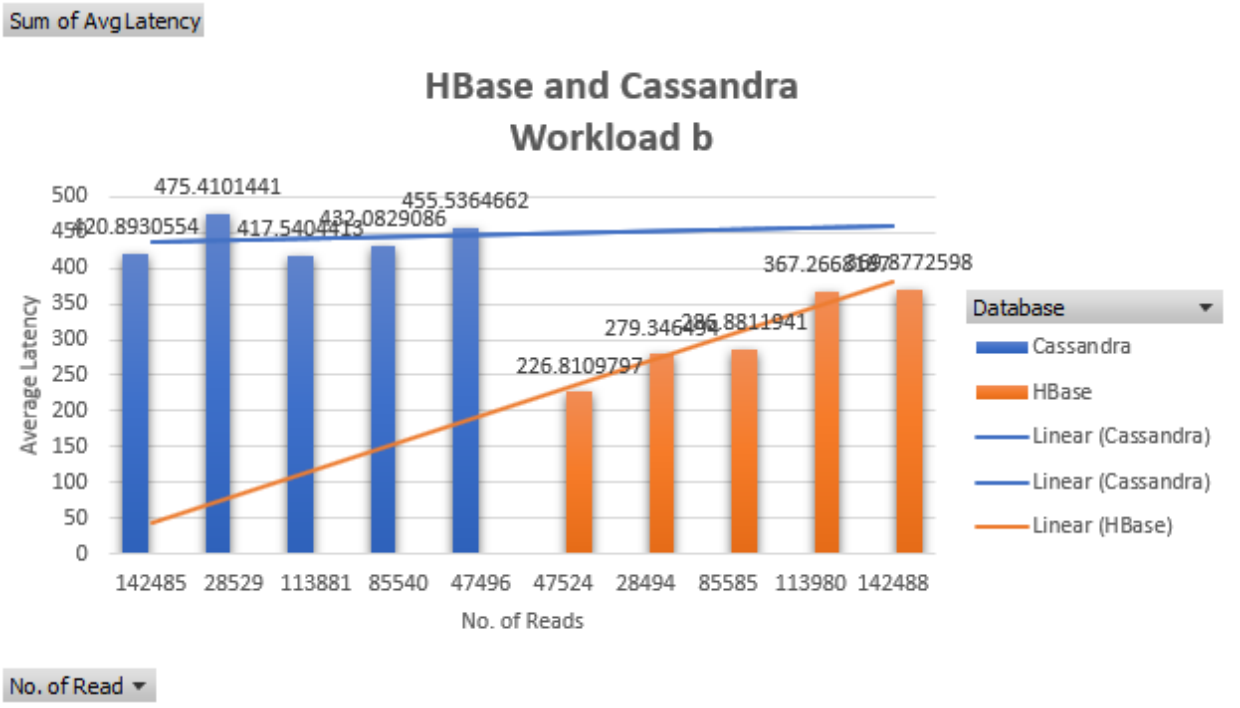
The mentioned databases (HBase and Cassandra) were gone through a run with variable operational count. 5 operational count are considered for the execution of tests with workload a and workload b. after successfully running the comparisons are made for the performance of Cassandra and HBase.

Average Latency Vs Read operation for Workload A:



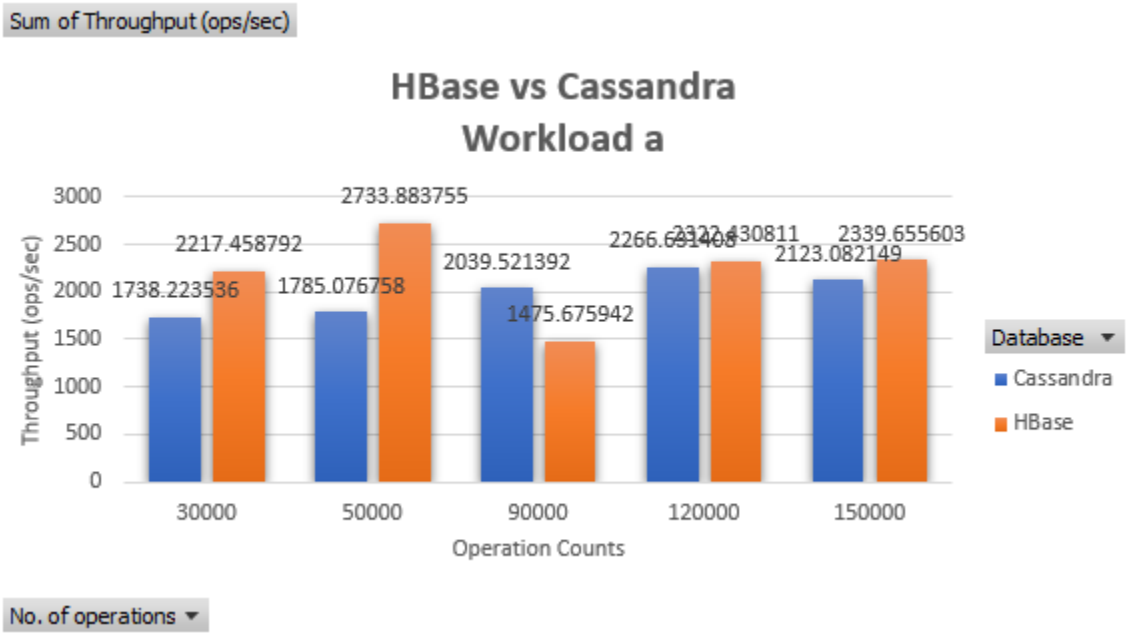
The above graph shows the graphical representation of the number of reads with average latency for the workload a. Here we can clearly see that for HBase the latency has increased as the number of read operations were increased. Here we get to see the linear latency of Cassandra as it has gradually decreased with less number of read operations and for the HBase it has gradually increased.

Average Latency Vs Read operation for Workload B:



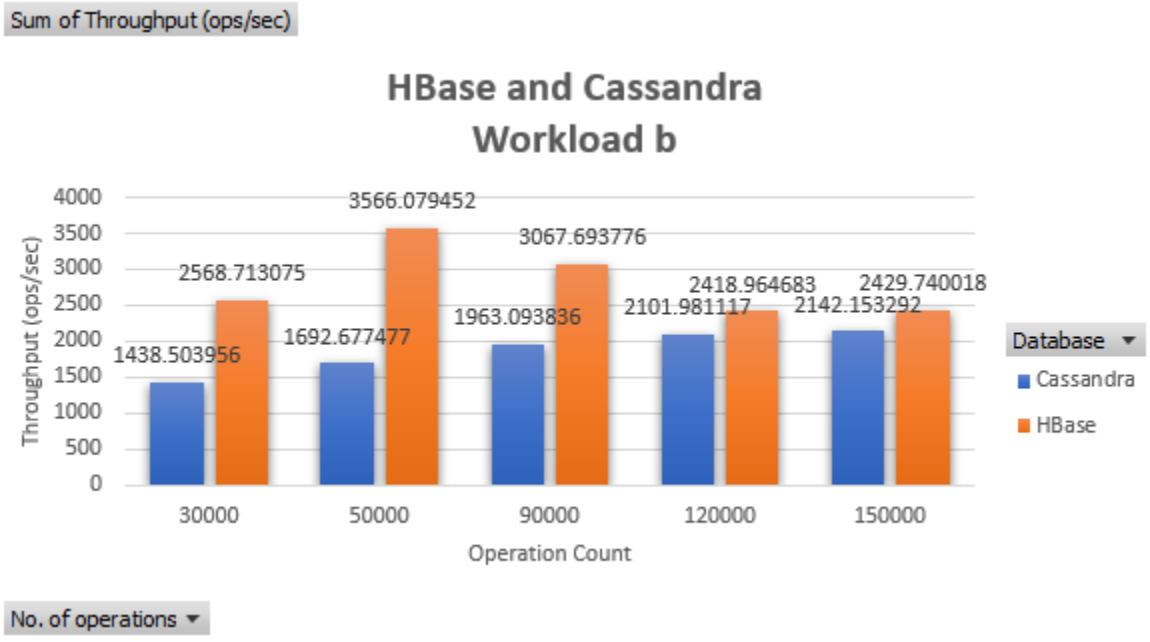
This is the comparison of average latency number of read operations for workload B. Here we can see that Cassandra is able achieve max latency throughout the operation counts as for the HBase we can see a steady increase by the time. For workload B Cassandra is able to achieve a linear latency as compared to HBase.

Throughput for Workload A:



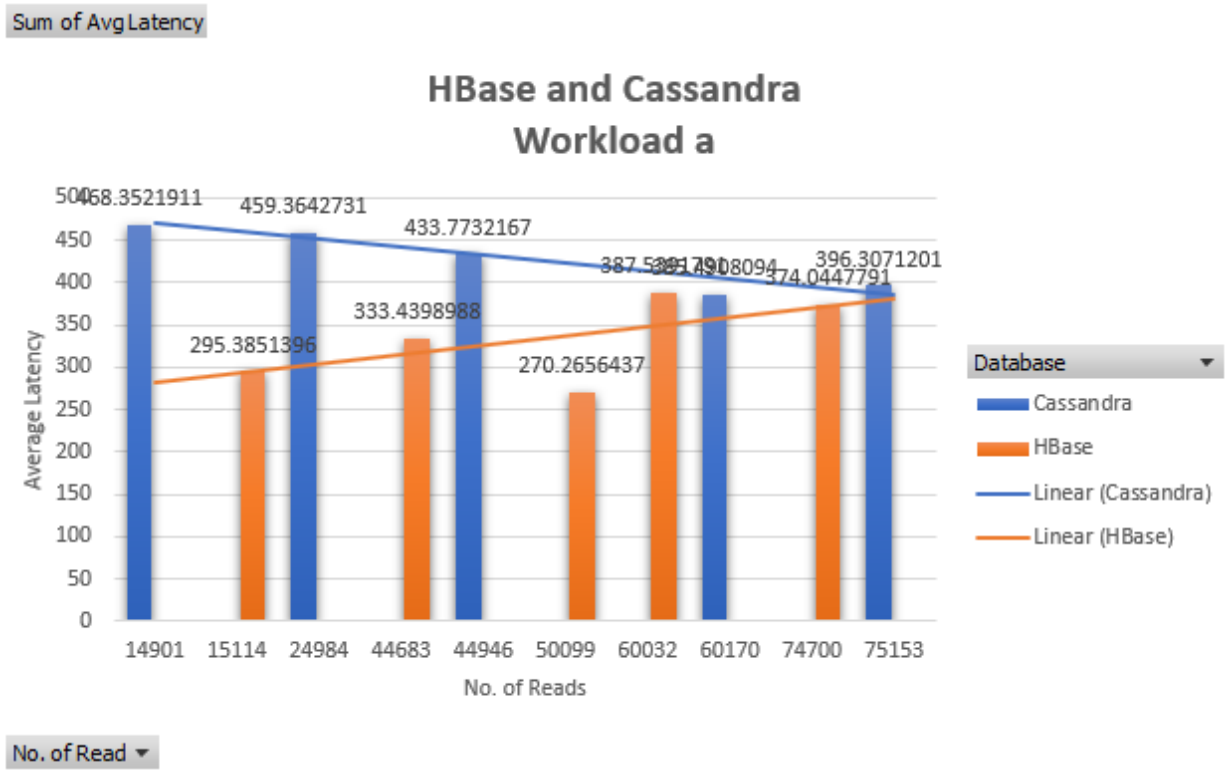
This is the comparison between the operation per second and the operation counts for workload A. HBase has higher throughput for operations per second as compared to Cassandra. Here we can see a variable behaviour of throughput for the operation counts.

Throughput for Workload B:



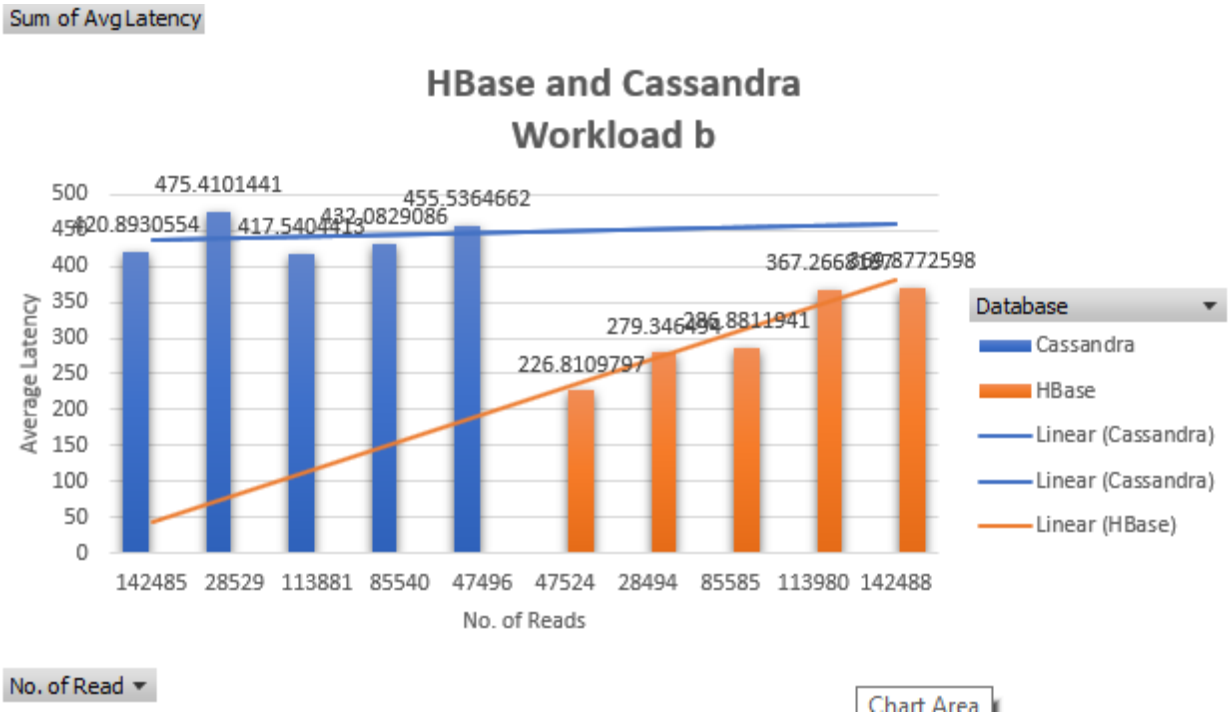
For workload B we can observe that the Cassandra has lesser throughput value as compared to HBase which results to Cassandra is better handling operation count which are gradually increasing.

Number of read operations Vs Latency for Workload A:



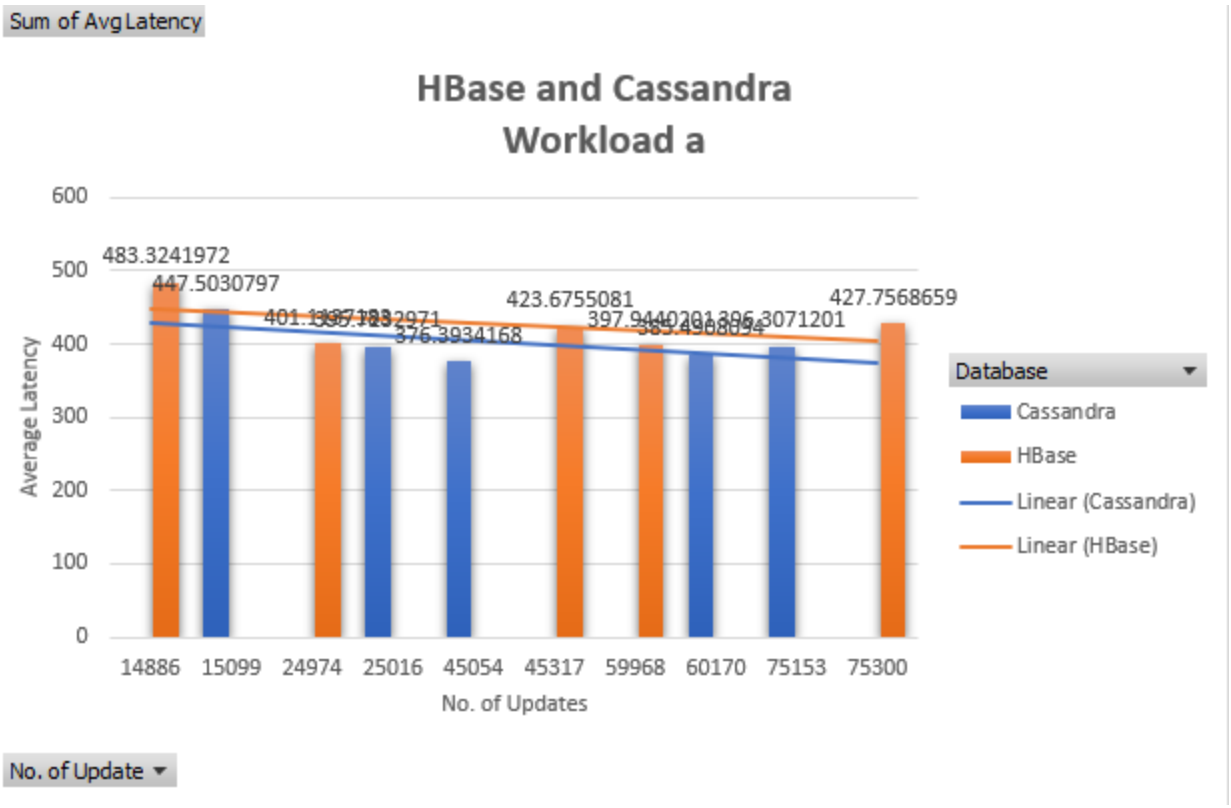
Here it is the graphical representation of the number of read operations with the average latency for workload a. We can see that the gradual decrease in the latency of Cassandra where the latency for HBase has increased.

Number of read operations Vs Latency for Workload B:



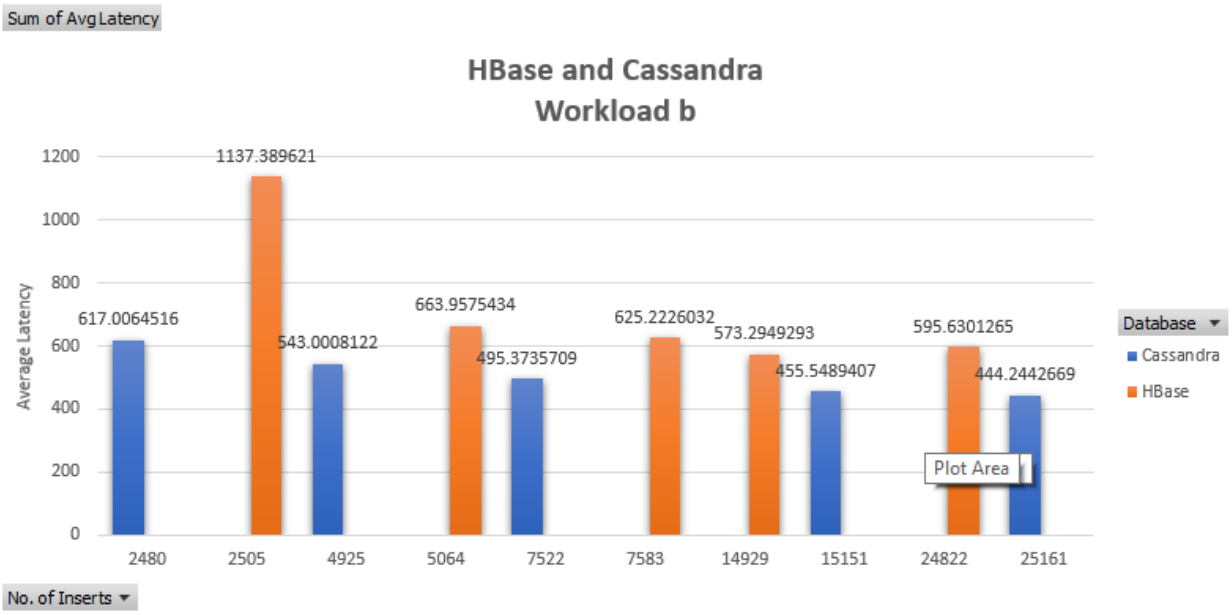
This is the graphical representation of number of reads compared with the average latency. Here it is observed that Cassandra has better latency rate than the HBase and certainly the linear latency is achievable for Cassandra much easily than that of HBase.

Number of read Update Vs Latency for Workload A:



This is the graphical representation of the number of update operation performed with compared to average latency of it. By this we can say that Cassandra is able to perform a greater number of update operation as compared to HBase.

Number of read Update Vs Latency for Workload B:



This is the graphical representation of total update operation performed with the average latency for workload B. by this we can see that Cassandra has much lesser latency than HBase.

Conclusion & Discussion:

In this project we have done the comparison of the two databases which are HBase and Cassandra. The result of the research is we can say that the HBase and Cassandra are better in performance as compared to other traditional databases. When the attention is drawn to the security constraints then HBase fail to provide enough security to the cluster and it is more prone to failure. Whereas Cassandra has provided enough security options which helps for data privacy and protection. After executing all the test on the YCSB platform, we have analysed and evaluated the results. After the test we were able to observe that Cassandra has a better performance status as compared to HBase for both of the workloads A and B. As this justifies that the test was sufficient enough to check all the constraints which leads us to a conclusion that Cassandra has a better latency in update and read

operations as well. We can also say that the overall throughput of Cassandra is comparatively better than HBase.

The main reason for Cassandra having much better performance because of the availability and the data replication which ultimately gave better output values than HBase. As the HBase has architecture of master slave which resulting impact on the test results as slave nodes has to perform functions autonomously.

References:

1. "HBase Overview", *www.tutorialspoint.com*, 2018. [Online]. Available: https://www.tutorialspoint.com/hbase/hbase_overview.htm [Accessed: 08- 05- 2018].
2. "Apache HBase", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Apache_HBase. [Accessed: 08- 05- 2018].
3. Srinivasa, K & G M, Siddesh & Hiriyannaiah, Srinidhi. (2015). Driving Big Data with Hadoop Technologies. 10.4018/978-1-4666-5864-6.ch010.
4. *Www2.cs.duke.edu*, 2018. [Online]. Available: https://www2.cs.duke.edu/courses/fall13/cps296.4/838-CloudPapers/ycsb.pdf. [Accessed: 08- 05- 2018].
5. "Securing Cassandra", [Online] https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureIntro.html. [Accessed: 08- 05- 2018].
6. Brewer EA (2000) Towards robust distributed systems. In: PODC. IEEE, Portland, Oregon, USA Vol. 7
7. Brewer E (2012) Cap twelve years later: How the "rules" have changed. Computer 45(2):23–29
8. Dezyre (2016) "Overview of HBase Architecture and its Components" [Online] Available: https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295 [Accessed: 08-May-2019]
9. Domaschka J, Hauser CB, Erb B (2014) Reliability and availability properties of distributed database systems. In: Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International. IEEE, Ulm, Germany. pp 226–233

10. Dory T, Mejías B, Van Roy P, Tran NL (2011) Comparative elasticity and scalability measurements of cloud databases. In: Proc of the 2nd ACM Symposium on Cloud Computing (SoCC). IEEE, Iasi, Romania Vol. 11
11. Lourenço, João & Cabral, Bruno & Carreiro, Paulo & Vieira, Marco & Bernardino, Jorge. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data. 2. 18. 10.1186/s40537-015-0025-0.
12. Tutorialspoint (2019) Cassandra-Architecture [Online] Available: https://www.tutorialspoint.com/cassandra/cassandra_architecture.htm [Accessed: 08-May-2019]