Data Storage and Management Project B

on

MySQL and Cassandra

Shikhar Srivastava
x18106960

MSc Data Analytics – 2018/9

Submitted to: Michael Bradford

MySQL and Cassandra

Shikhar Srivastava
x18106960

December 19, 2018

Abstract

We are in an era where data has become a new fuel for every corporate giant. Companies are using data in several different ways to benefit. As everybody now is learning the importance of data, volume of data is growing drastically every year. This in turn raises a new point of deciding where to store this data so that it can be retrieved and processed efficiently as per requirements. There are a lot of databases in the market today, but the question about which one to choose depends on a lot of factors. One of such factors is How well a database performs? keeping in mind the amount of data one has to store or might process. This project studies the performance of 2 databases, MySQL and Cassandra through a benchmarking tool called YCSB. YCSB is widely used to test performance of databases so that an organization can make a logical decision of choosing a certain database. Two YCSB workloads are run for both databases to compare their performance graphically.

1 Introduction

As data is growing day by day, companies are getting more interested and concerned about how well their data can be stored, processed and retrieved while maintaining an optimized performance. Strategy towards achieving this starts right from choosing a correct database from a list of databases available from many different vendors present in the market today. One of the important steps in choosing a database is to test its performance keeping in mind an approximate volume of data which they will be handling. This is important because if an organization chooses a database without performing this test, they might end up with slow processing and data retrieval and this is something which no organization want because need today is fetching and processing data as fast as possible. It is sometimes hard to choose a correct database as there are a lot of vendors present in the market and each one having a distinct feature as compared to others. So, the best method is to test the performance of all chosen databases through a benchmarking tool. Project is using a benchmarking tool called YCSB which can be used to test performance of databases.

Two databases, MySQL and Cassandra are picked for which the project is aiming to carry out performance tests. YCSB provides metrics like the overall throughput of database, its average latency along with operation counts that can be obtained for four different kinds of workloads present in YCSB which are mentioned in Table 1 below. Project uses Workload A and Workload E for carrying out performance analysis on two databases mentioned above. Both YCSB workloads are run for four different operation

Serial No.	Workload	Brief Summary
1	Workload A	This workload performs 50% read and 50% write operations.
2	Workload B	Workload performs 95% read operations and 5% write operations.
3	Workload C	Performs 100% read operations.
4	Workload D	It reads latest records inserted in the database.
5	Workload E	This workload queries short ranges of records.
6	Workload F	Here records are read, modified and then modified records are inserted back to the database.

Table 1: YCSB Core Workloads GitHub (2010)

counts i.e. 25000, 50000, 100000, 200000. For each database, each workload and each operation count performance metrics like throughput, average latency w.r.t. insert, read, update operations are measured through YCSB. These metrics can further be used to analyze database performance. Output from YCSB is further re-arranged in excel and fed in Tableau to check these performance metrics by building informative graphs and finally drawing fruitful conclusions from these graphs.

2 Characteristics of Chosen Data Storage Management Systems

All data storage management systems are categorized by a set of features on which their different processes are based. One way in which MySQL and Cassandra differ is that MySQL is a relational database management system while Cassandra is not. Both of these databases are characterized by different characteristics. In this section well look at some of the characteristics of our two databases, MySQL and Cassandra.

2.1 Cassandra Characteristics Hewitt (2010)

Cassandra is an open source distributed database which was first developed at Facebook. Cassandra is now widely used by many reputed and large companies as their database. Few of the characteristics of Cassandra are as follows:

2.1.1 Distributed and Decentralized

Cassandra works as a distributed file system, which means that can run on multiple machines. It is engineered to not only run on multiple machines but also work in an optimized way across several data center racks. One characteristic of Cassandra which distinguishes it from other distributed databases is that it is decentralized. This means that Cassandra does not follow 'master-slave' node approach, in it every node is identical.

So, when Cassandra is scaled up, there is no need to set up a master or a slave. Instead, Cassandra uses gossip protocol to keep track of dead and alive nodes.

2.1.2 Elastic Scalability

There are basically two types through systems can be scaled up or down. Vertical scalability refers to directly adding or removing hardware as per the requirements. Horizontal scalability means to add or remove a machine. In Cassandra, Elastic Scalability in Cassandra refers to special type of horizontal scalability. This type of horizontal scalability means that the cluster should smoothly scale up or down without any major disruption. If a machine is added in Cassandra, it smoothly blends in the cluster and starts participating by getting some amount of data. Similar process is followed by Cassandra if a machine is move dout of the cluster. Suppose, a machine is removed from a cluster, data is automatically distributed to other machines.

2.1.3 High Availability and Fault Tolerance

A systems' availability is measured in terms of ability to process or fulfill requests. There should be no downtime if a system is highly available. Machines in a highly available system are on multiple networks and softwares know how to work in clusters. Cassandra is highly available and data ca be replicated to other data centers if some failure happens with no down time.

2.1.4 Schema-Free

In Cassandra you don't need to spend money on expensive data modelling tools to model your data. It just asks you to define an outer container which is called 'keyspace'. Keyspace holds column families and some configuration details. You don't need to build a schema in Cassandra.

2.1.5 Performance

Cassandra is super fast and it was designed to work on multiprocessor-multicore machines. It can run on many machines across many data centers and can scale up to hundreds of terabytes smoothly and seamlessly. Cassandra is famous for performing exceptionally well under heavy workloads.

2.2 MySQL Characteristics

Following are some of the characteristics of MySQL:

2.2.1 InnoDB - Default Storage Engine Oracle (2011)

MySQL data storage engine was changed and upgraded to InnoDB from MySQL version 5.5. It was the most notable advancement in MySQL back then. InnoDB storage engine strictly follows ACID properties for transactional operations and also comes with unique properties which ensure high performance and scalability. InnoDB was re-architected for MySQL 5.5 so as to take advantage of crash recovery, referential integrity and high user concurrency in an optimal way.

2.2.2 MySQL Replication Oracle (2011)

No organization can bear loss of data on any way. To tackle this threat they set up different methods like data replication, data duplication etc. MySQL comes with a data replication feature utilizing which the admins can create multiple replicas of their database in a simple way. This also increases the scalability of database beyond single instance.

2.2.3 Security Oracle (2011)

Number of threats to database are increasing every day and database professionals' most important job is to protect and guard organizations' database. MySQL provides several levels of security features to tackle and help organization protect their data. It comes with powerful mechanisms to ensure that only authorized users can access the database. If required, MySQL blocks users from entering into the database. SSH and SSL connections are provided to make sure that secure connections are established. MySQL also comes with a capability which check data accessibility to users so that users can only see data they are authorized to.

2.2.4 Internals MySQL (2018*a*)

MySQL is coded in C and C++ and is tested with broad range of different compilers. It uses B-Tree disk tables which comes with index compression.

2.2.5 Cross Platform Availability MySQL (2018*a*)

MySQL is flexible to use across a variety of platforms. From Oracle linux and ubuntu to Microsoft Windows and MacOS, MySQL can easily be used on any commonly used platform. Platforms on which it can be used are: Oracle/RedHat Enterprise Linux, Fedora Linux, Ubuntu Linux, Debian Linux, SuSE Linux, Oracle Solaris, Microsoft Windows, Apple MacOS, FreeBSD and many more.

3 Database Architectures

3.1 Cassandra Architecture Hewitt (2010)

3.1.1 System Keyspace

Cassandra is not a relational database but like Oracle has its tablespace called SYSTEM, Cassandra too has a keyspace called SYSTEM where it stores configuration details and metadata about the cluster to help in carrying out operations.

3.1.2 Peer-to-Peer

Unlike other databases which use multiple machines, Cassandra does not follow the traditional master-slave model. Instead, it follows peer-to-peer model, which means that there is no master and no slave, each node present in Cassandra clusters are identical. This improves Cassandras' system availability and it can be easily scaled up and down as required.

3.1.3 Gossip and Failure Detection

Each node in Cassandra has state information about other nodes. Cassandra works on gossip protocol based on which every node present in Cassandra clusters kind of communicate with each other. The gossip protocol runs every second. For an example, a signal is triggered if a node notices that another node for each it is having state information comes back online.

3.1.4 Anti-Entropy

It is often observed that where ever gossip protocol is present, anti-entropy is also present there. Anti-entropy is a mechanism which keeps the replicas in different nodes up to date. It means that this mechanism ensures that replicas in each node are updated to the newest version.

3.1.5 Hinted Handoff

This is a very interesting phenomena in Cassandra. Suppose, if user has written a write request, but the node for which the write request was created has failed somehow. In this case, another node receives the write request which is actually not meant for that node, so, this node saves a hint to send it later to the node for which the write operation was actually written for after it back online.

3.1.6 Staged Event-Driven Architecture SEDA

Normally, what happens is an operation starts in a thread and end within the same thread. But in Cassandra, an operation might start from one thread and end in within another thread. An operation is divided in stages and execution is decided by the thread pool.

3.1.7 Other Components

Other key aspects in Cassandra architecture are: Bloom Filters, Tombstones, Cassandra Daemon, Hinted Handoff Manager.

3.2 MySQL Architecture Nellutla (2017)

Like all other relational databases, MySQL also comprises of three layers: Application layer, Logical Layer and Physical Layer.

3.2.1 Application Layer

This is the layer where users an clients interact with MySQL. This layer consists of Database Administrators, Clients (APIs) and Users (who interact with MySQL through query). Refer Figure 1.

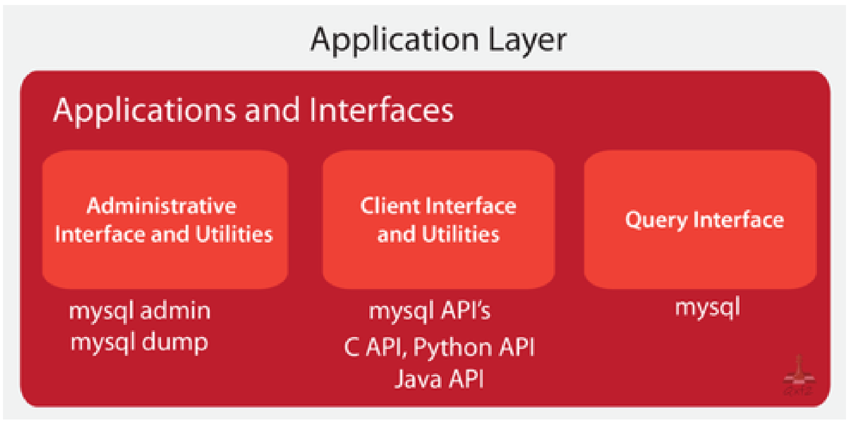
Application Layer
Applications and Interfaces
Administrative Interface and Utilities
Client Interface and Utilities
Query Interface
mysql admin
mysql dump
mysql API's
C API, Python API
Java API
mysql

Figure 1: Application Layer

3.2.2 Logical Layer

Logical layer is the layer which takes data from the application layer. This layer consists of Query Processor, Transaction Management, Recovery Management, Storage Management. These all systems together process the input from application layer and feed it to the physical layer. Refer Figure 2.

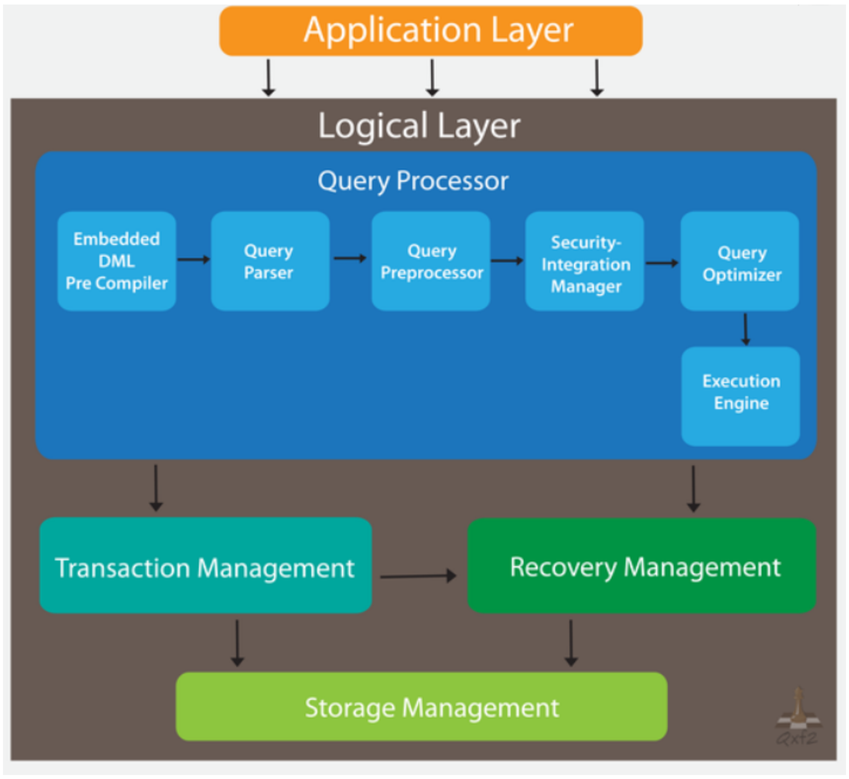

Figure 2: Logical Layer

3.2.3 Physical Layer

This layer is responsible for storing and retrieving data in MySQL. MySQL storage layer comes with a flexibility of adding or removing a storage engine from a running MySQL server which is not present in other relational databases.

4 Scalability, Availability and Reliability of Chosen Data Storage Management Systems

Scalability, Availability and Reliability are important aspects of data storage management systems. These parameters define how flexible and robust a database is. Under this section we'll discuss above mentioned parameters for our chosen databases, MySQL and Cassandra.

4.1 Scalability

4.1.1 MySQL Data Storage Management System MySQL (2018*b*)

Depending on operational and availability requirements, MySQL offers different types of architectures which are: Data Replication, Clustered and Virtualized Systems and Shared-Nothing, Geographically-Replicated Clusters. As we move from the first type of architecture to third type, cost and complexity increases exponentially. There is downtime if we want to scale out our MySQL database system when data replication architecture is used. For second and third type of MySQL architectures there is no downtime if we want to scale out the system. This means that if we need to add a new node to MySQL system and assume that data replication architecture is used, then there will be a downtime until the new node has not been successfully deployed in the system, whereas, for second and third architectures, there is no downtime and the new node can be added on the go while the system is online.

4.1.2 Cassandra Data Storage Management System Hewitt (2010)

Cassandra is known for its elastic scalability which follows dynamic horizontal scaling technique. In this, if a new node/machine is added in the cluster, Cassandra automatically detects it and starts to it work. It works in a similar fashion if there is a need to scale down the system. User does not need to restart the process. So, there is no downtime in Cassandra while scaling up or down the system by adding or removing a node.

4.2 Availability

4.2.1 MySQL Data Storage Management System MySQL (2018*b*)

As discussed above, based on operational and availability requirements, MySQL offers different types of architectures which are: Data Replication, Clustered and Virtualized Systems and Shared-Nothing, Geographically-Replicated Clusters. Availability is defined as ability to take requests if there is some failure in the system. Availability of MySQL is very low when it is used with data replication architecture, whereas, if the other two architectures are used, its availability increases drastically. So, if second or third architecture is implemented, MySQL can still process requests if there is a failure like host failed, or operating system failure.

4.2.2 Cassandra Data Storage Management System Hewitt (2010)

Availability of Cassandra is also very high. Users can replace the failed nodes with no downtime. Data in Cassandra can be replicated to different geographically located data centers and if any of the data centers fail, it can still process requests with no downtime.

4.3 Reliability

4.3.1 MySQL Data Storage Management System

Reliability of a database is very important which is based on the performance of database. It is observed that MySQL is not always consistent performance wise. Although, from above two points we can see that reliability of MySQL is good if Clustered and Virtualized Systems or Shared-Nothing, Geographically-Replicated Clusters architectures are used, but, it is low as compared to other leading data storage and management solutions.

4.3.2 Cassandra Data Storage Management System

Cassandra on the other hand is considered to be a highly reliable data storage and management system. It is very flexible in scalability and does not follow master-slave model and still Cassandra is known for its good performance for high workloads. It is considered to be more reliable than MySQL.

5 Learning from Literature Survey

The advertisement of various database vendors showing how well their database performs as compared to other database is nothing but the benchmark results. Benchmark results are widely used by database vendors to advertise their database where they show performance comparison between their database and other databases. Transaction Processing Performance Council `www.tpc.org` is a committee which defines transaction processing and database benchmarks and delivers trusted results to the industry. Currently, major members of TPC are corporate giants like Teradata, Oracle, Intel etc. The TPC benchmarks are divided into four categories: TPC-C, TPC-H, TPC-R, and TPC-W. Among all four of the benchmarking tests, TPC-C is the most commonly used benchmark test. TPC-C pwerforms transactions of an order entry environment Callan (2005). Not sure whether this test is still carried out nowadays or not, because there are other good softwares and tools like YCSB which can perform the same task.

Nowadays, a number of benchmarking tools are available in the market which can be used to carry out the benchmarking tests for various types of databases. NoSQL or 'Not Only SQL' databases are also prevalent in the market today and many companies are using them as well. These databases can also be tested using a benchmarking tool. One recent study actually tests few NoSQL databases using a popular benchmarking tool called YCSB(Yahoo! Cloud Service Benchmark). NoSQL databases which were used for the study are HamsterDB, LevelDB, STSdb 4.0, RavenDB and BrightstarDB. YCSB test was performed on all of the four databases and results were for following parameters: 'Speed of insertion of records per second', 'Database size expressed in MB', 'Time of insertion of generated records expressed in seconds', 'Maximum utilization of operational memory expressed in MB', 'Speed of reading of records per second', 'Database size expressed in MB', 'Reading time of inserted records expressed in seconds' and 'Maximum utilization of operational memory expressed in MB'. After performing tests on above mentioned parameters, it was concluded that HampsterDB performed the best while BrightstarDB performed the worst Krsti & Krsti (2018).

6 Performance Test Plan

We all know that data is increasing, and increasing exponentially. There are a number of vendors present in the market today and with exponentially increasing data, a lot companies are moving towards NoSQL databases from traditional relational databases because for their ever growing data, they need an infrastructure which is highly scalable, fast, flexible and fault tolerant. NoSQL databases come with such characteristics. But simultaneously, there are a lot of NoSQL database solution present and it sometimes becomes very confusing to choose from a wide range of products because each vendor boast about their own product.

To handle this situation in a more logical way, companies should use benchmarking tools to test performance of databases. YCSB (Yahoo! Cloud Service Benchmark) is one of the most popular benchmarking tools available and its free. It comes with some pre-loaded workloads types which can be run for different operation counts as per requirements. MySQL and Cassandra are chosen for this project to perform a benchmark test. We'll test these databases to get insights about their performance. As discussed about YCSB above, it is a reliable benchmarking tool and commonly used by many companies to carry out benchmark tests. We'll be using YCSB to test performance of our two chosen databases and out of the four core workloads which YCSB provides, for this project I decided to pick following workloads, Table 2:

Serial No.	Workload	Brief Summary
1	Workload A	This workload performs 50% read and 50% write operations.
2	Workload E	This workload queries short ranges of records.

Table 2: Workloads Used for this Project GitHub (2010)

For each database, both workloads were run individually for four different operation counts i.e. 25000, 50000, 100000, 200000. The benchmark test was run 3 times for both databases and an average value of three run was taken to get a better picture. Although for each database and each workload test for all four operation counts mentioned above were run, **but unfortunately for Cassandra, test for operation count 200000 failed in all 3 runs. YCSB showed "[INSERT-FAILED]" every time for this operation count and didn't insert all the records in Cassandra 'usertable' for which reason is unknown. However, few records were inserted. So, latency figures might not be accurate for Cassandra for operation count 200000**

The test was carried out on an openstack instance, for which details are given below in table 3:

Specification	Value
Instance Name	x18106960-DSMPROJBNEW
Instance IP	192.168.101.143
Instance Floating IP	87.44.4.85
RAM	4GB
Disk	40GB
VCPUs	2 VCPU

Table 3: Openstack Instance Configuration

WinSCP was used to import YCSB output files from openstack ubuntu OS to local system. High level steps followed to run the test in order from start to end are:

Create an openstack instance — Associate a floating IP address to it — Install MySQL (version 5.7.24) — Install Cassandra (version used here is 2.2.9) — Install YCSB (latest version 0.14.0) — Install Testharness — Edit opcounts.txt file in testharness folder and enter opearation counts 25000, 50000, 100000, 200000 — Edit testdbs.txt file in testharness folder and enter jdbc and cassandra2-cql — Edit workloadlist.txt file in testharness folder and enter workloada and workloade — Edit runtest.txt file in testharness folder and add/change necessary values like add threads — Edit db,properties in /home/hduser/ycsb-0.14.0/jdbc-binding/conf and change password if password is set other than 'password' (This is MySQL password) — Run YCSB test 3 times through testharness — Import YCSB output files in local system through WinSCP — Modify and import data in an excel spreadsheet (Average of all 3 runs was taken) — Import Excel spreadsheet in Tableau to draw graphical representations of the output from YCSB — Draw Conclusions

7 Evaluation and Results

As mentioned in section 6, YCSB output files were imported from openstack ubuntu to local machine with the help of WinSCP. These files were further re-arranged and to feed it in Tableau. As YCSB was run three times, average values of all metrics was calculated and then this result was fed into Tableau.

through YCSB, following parameters were measured for all four record operations: Overall throughput of database, Average Latency for READ operations, Average Latency for UPDATE operations, Average Latency for INSERT operations, Average Latency for SCAN operations.

7.1 Overall THROUGHPUT of both databases

Throughput of databases is defined as the operations performed by the database per second. Cassandra is believed to be very fast with throughput, which can also be seen by the plot built from YCSB output files.

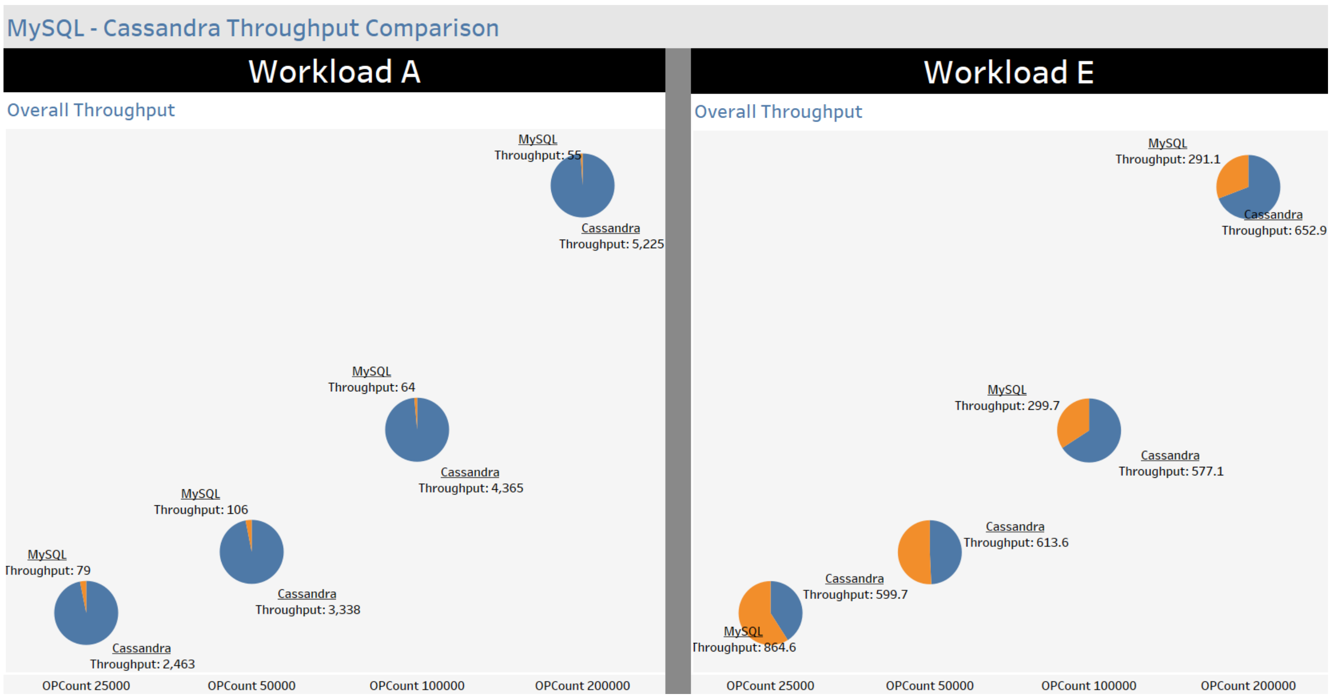

Figure 3: Throughput Comparison for both Workloads

Graph for comparing throughputs of both databases was plotted in 2 different worksheets and then those 2 worksheets were pulled in a dashboard. The dashboard is divided in two parts, Workload A and Workload E. Both parts are showing a plot of pie charts per workload. For Workload A, we can see that Cassandra is far more fast than MySQL, which means that READ and UPDATE operations are performed faster by Cassandra as compared to MySQL and Cassandra appears to be consistent in performance not matter what number of operation count is. For Workload E, where INSERTION and then SCANNING is done by YCSB, its is noticed that MySQL is giving higher throughput at operation count 25000 and then as the operation count starts to increase, throughput of Cassandra is much better than MySQL. So, what is believed that Cassandra is very fast seems to true as per this test.

7.2 Comparison by Average Latency for READ Operations

Latency measures the the time in which a database performs one operation. In this section, we'll see how MySQL and Cassandra behave if they just need to read records from a table.

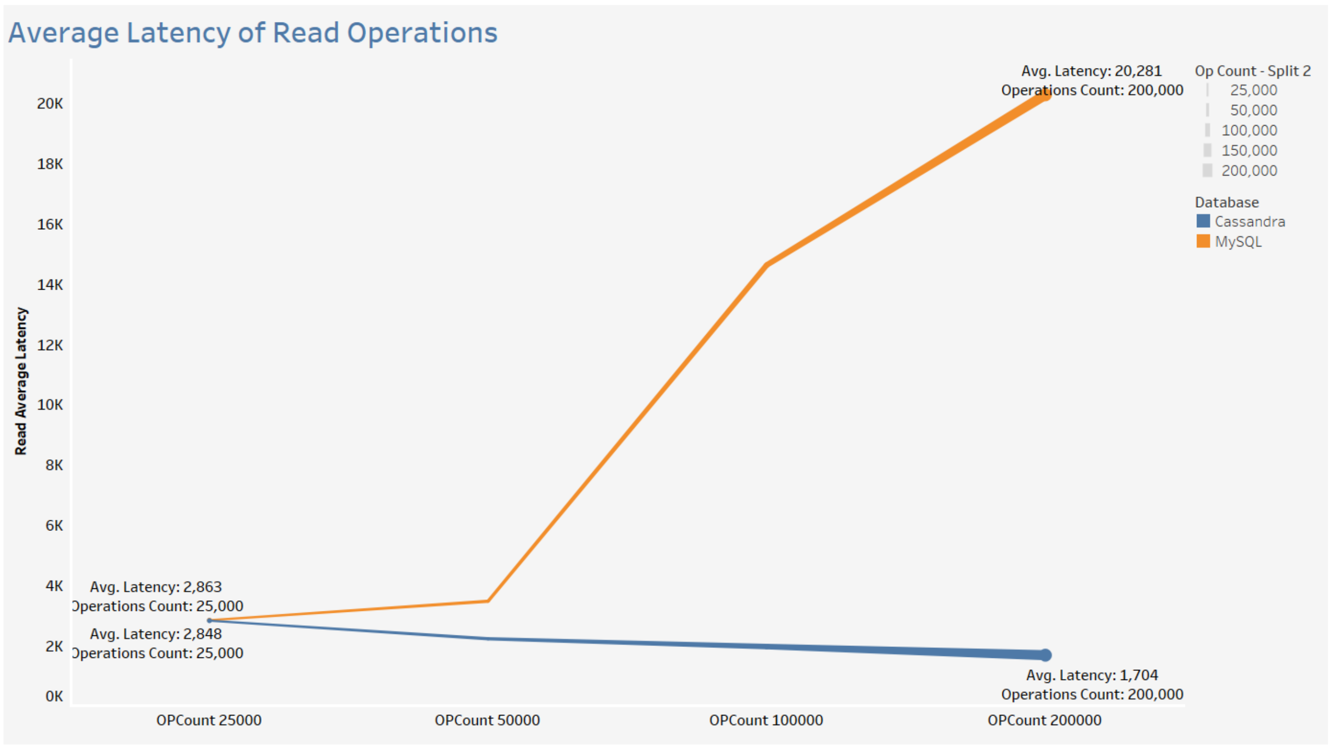

Figure 4: Average Latency for READ Operations

In the graph above, thickness of the line represents the operation count which means that as the operation count increases, the lines get thicker, that is why the lines are thickest at operation count 200000.

When both the databases are tested on how fast they can perform READ operations on a table, MySQL still lags behind. It can be seen that at operation count 25000 Avg. Latency for both MySQL and Cassandra are almost similar and at operation count 50000 MySQL lags a behind Cassandra. But, Cassandra is way more faster for operation count greater than 50000. Performance of Cassandra even boost at higher operation counts, which is impressive.

7.3 Comparison by Average Latency for UPDATE Operations

Now another important operation for a database is an UPDATE operation. We'll now check YCSB output of how well both databases perform on UPDATE operations.

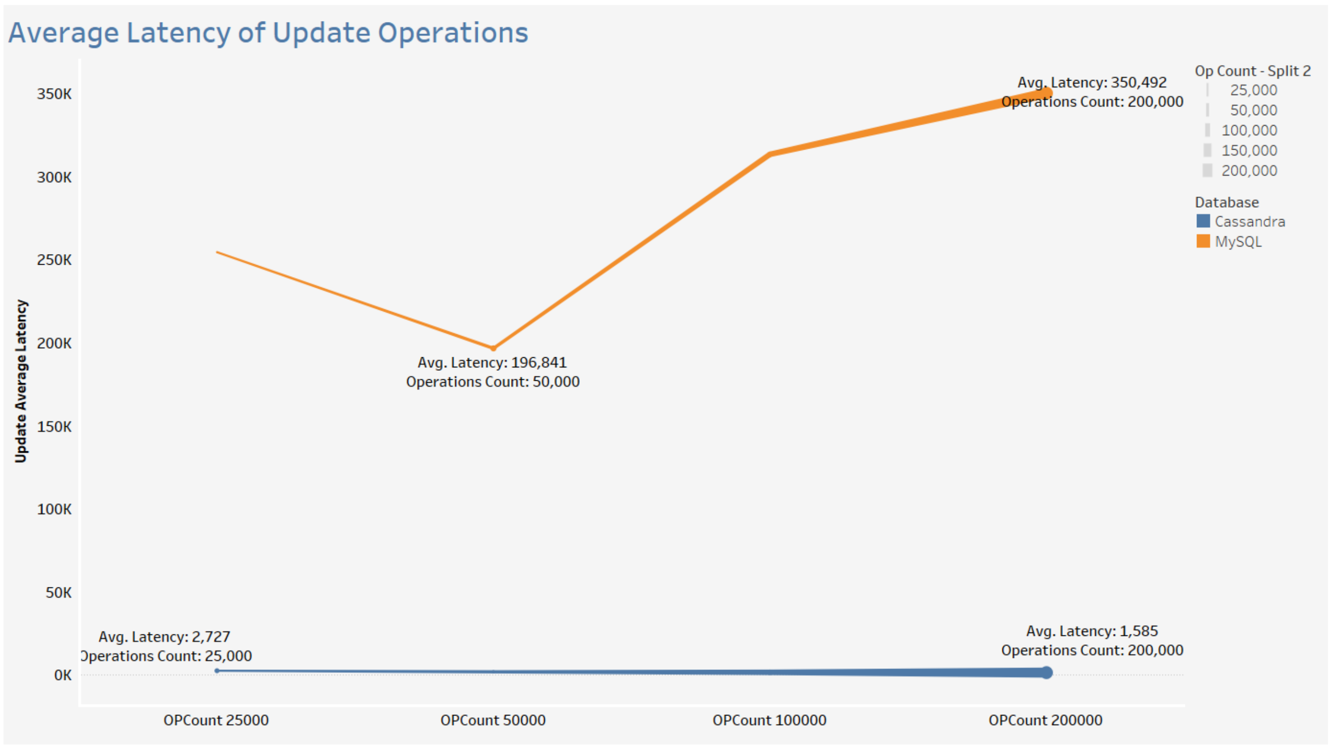

Figure 5: Average Latency for UPDATE Operations

Thickness of the line represents the operation count which means that as the operation count increases, the lines get thicker, that is why the lines are thickest at operation count 200000.

And yet again, Cassandra looks to be a winner. It can be seen from Figure 5 that MySQL in each operation count is not even close to Cassandra in performance terms. MySQL average latency for UPDATE operations is in six figures while Cassandra is lying down very close to the x-axis and is consistent performance wise as compared to MySQL. From the plot, it looks like operation count is having no effect on Cassandra for UPDATE operations.

7.4 Comparison by Average Latency for INSERT Operations

INSERT operations are also very important and can take a lot of time if database is not efficient in handling them.

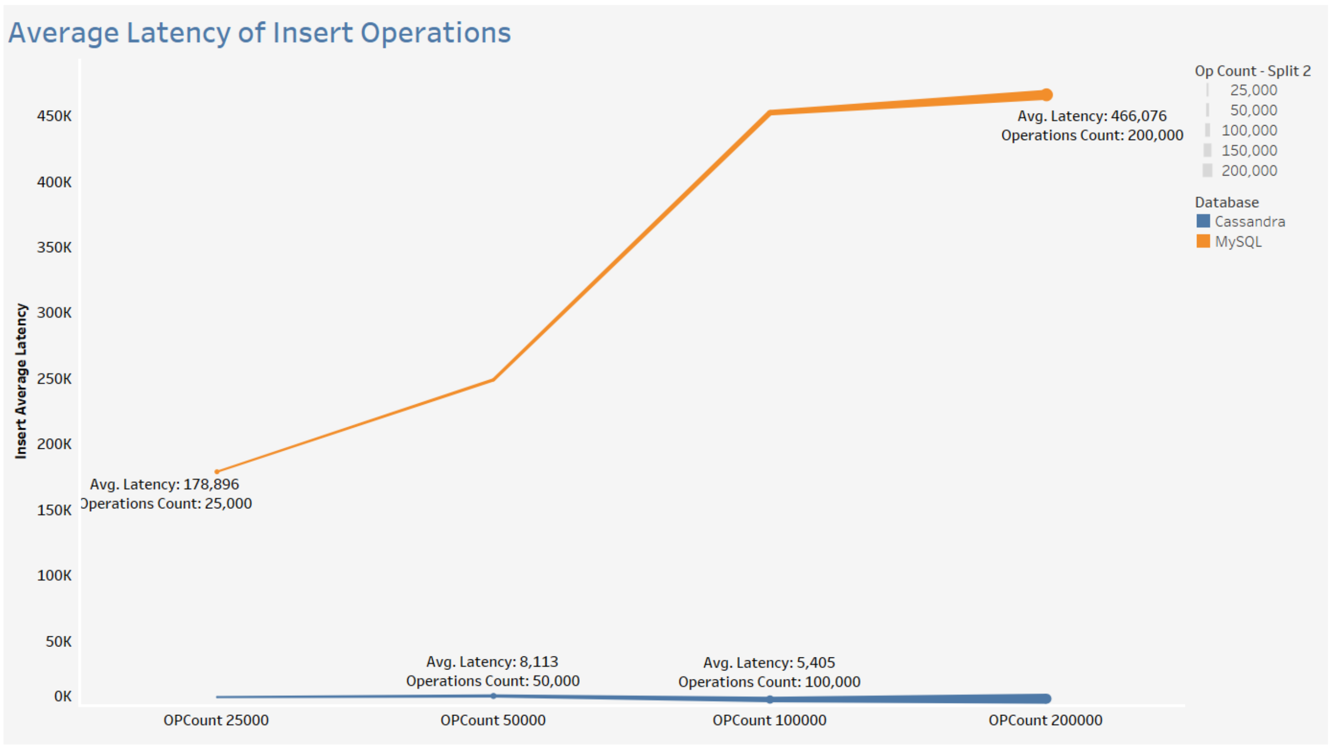

Figure 6: Average Latency for INSERT Operations

Thickness of the line represents the operation count which means that as the operation count increases, the lines get thicker, that is why the lines are thickest at operation count 200000.

Similar pattern can be seen here also. Average latency of MySQL is very high as compared to Cassandra and Cassandra seems to perform in an excellent manner for INSERT operations. If we notice, and look at UPDATE operations and INSERT operations plot, we'll see that Cassandra is even better in performance for INSERT operations as compared to itself for UPDATE operations.

7.5 Comparison by Average Latency for SCAN Operations

I think this is the most commonly and frequently used database operation which every person working in data comes across every day.

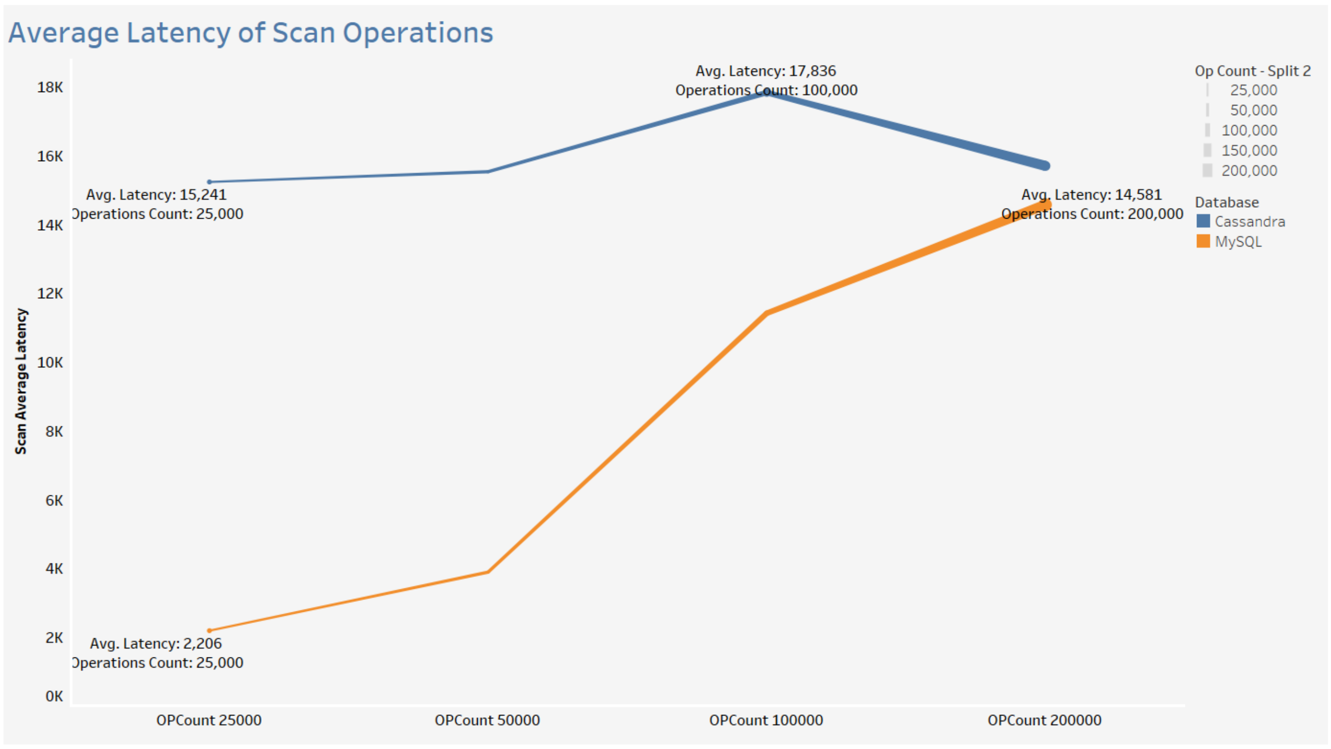

Figure 7: Average Latency for SCAN Operations

Thickness of the line represents the operation count which means that as the operation count increases, the lines get thicker, that is why the lines are thickest at operation count 200000.

This plot comes up with an interesting pattern. From the plot one can notice that performance of MySQL is better the Cassandra for SCAN operations. There is a huge gap between average latency of MySQL and Cassandra for lower Operation counts with MySQL performing way more better than Cassandra. But, aa the operation count increases, average latency of MySQL also increases, while the average latency of Cassandra is decreasing for high operation counts. At operation count 200000 both the databases are very close to each other and as the trend can be seen from above plot, Cassandra might perform better than MySQL at much higher operation counts, may be greater than 500000 or so.

8 Conclusion and Discussion

Data storage and management has become very important in today's world because almost all companies are facing data explosion especially big corporate giants. This massive amount of data needs to be stored at a place from where it can be processed and retrieved efficiently. Keeping this in mind, Information Technology is growing and providing interesting new solutions for data storage and management. There are a lot of different types of database solutions available in the market ranging from relational databases to NoSQL databases. It is a tough task to choose a database solution which can suit a particular organization from a heap of database solutions available. One has to decide after getting knowledge of what quantity of data will be handled by the database, how scalable it is and how it is in performance. Suppose this parameters are clear in ones head, the other most confusing thing is that every vendor talks best about their product and this becomes a challenge to decide which vendor to go with. This is when measuring performance of those databases comes into picture. The last step towards choosing a database is measuring its performance whcich can be done using several readily available benchmarking tool like YCSB.

Performance tests carried out in this project using YCSB suggest that Cassandra is a much better database as compared to MySQL and its performance is extremely good at high operation counts. Except SCANNING the database, Cassandra surpasses MySQL in other major database operations like READ, UPDATE and INSERT. MySQL however, performs better for SCANNING operation at low operation counts and SCANNING is the most common operation which every database professional performs. So, it depends on the requirement. If volume of data to be accesses and processed is low, one might use MySQL. But still, I'll conclude that Cassandra is a much better option to choose than MySQL.

References

Callan, S. (2005), 'Database benchmarking'.

GitHub (2010), 'Core workloads'.

Hewitt, E. (2010), 'Cassandra: The definitive guide'.

Krsti, L. J. & Krsti, M. S. (2018), 'Testing the performance of nosql databases via the database benchmark tool', *vojtehg* **66**(3), 614–639.

MySQL (2018*a*), 'The main features of mysql'.

MySQL, I. (2018*b*), 'High availability and scalability'.

Nellutla, I. (2017), 'Mysql architecture and layers'.

Oracle (2011), 'Top 10 reasons to choose mysql for web-based applications'.