Benchmarking HBase and MongoDB

Abstract

NoSQL databases have been very efficient while dealing Big Data because of BASE (Basically Available, Soft state, Eventual consistency) principle that it follows rather than (Atomicity, Consistency, Isolation, and Durability) properties which relational databases follow. The objective of this experiment is to compare the performance of two of the most widely used NoSQL servers namely Apache HBase and MongoDB using Yahoo Cloud Serving Benchmarking (YCSB) tool for two different workloads and four different operation count and to comprehend how these two databases perform under different combination of workload and operation count.

According to the findings from this experiment, it can be concluded that MongoDB is a better database in terms of low average latency as well as high throughput. (Zack, 2016)

Introduction

Three Vs (Volume, Variety and Velocity) that are defining properties of Big Data. Volume refers to the amount of data, variety refers to types of data and velocity refers to the speed at which data is generated. All three properties of data have increased at an enormous speed in a few years and so as the challenges to cope up with this ever-increasing data. Relational databases have been doing the job of storing the data very well up till the last couple of years but, now the modern-day requirements because of big data is not fulfilled by relational databases and use of relational database in distributed environment becomes very inefficient and difficult so came NoSQL databases that provides solutions to all these challenges of Big Data. (Rouse, 2013)

Though there are many NoSQL databases around Cassandra, HBase and MongoDB have been very popular. In this experiment comparison of MongoDB and HBase is done using YCSB and testharness tool to find out how these two databases perform under different workload for different operation counts.

Key Characteristics

HBase

HBase is a column-oriented database which runs on the top of HDFS and provides dynamic database schema. Following are the key characteristics of HBase:

- Availability

HBase offers failover and recoverable Local Area Network and Wide Area Network with the help of master server which controls monitoring of region servers and metadata of cluster.

- Sharding

Whenever data becomes too large to handle in a region HBase uses automatic and configurable splitting of blocks into smaller blocks thus reducing input/output time and overhead.

- Failover

Data in HBase is distributed and replicated over multiple regions which facilities recovery of data with the help of HDFS and RegionServer in case of failover.

- Parallel Processing

Parallel processing of big volume of data can be achieved in HBase with the help of convenient base classes of Hadoop MapReduce jobs.

- Java API

HBase provides simple to use Java API which can be used for client-side access.

- Real time processing

With the help of Bloom filters and Block cache real-time queries can be processed.

- REST web service and Thrift gateway

HBase supports REST web service and Thrift gateway which enables Protobuf, XML and binary data encoding options for non-java clients.

- Scalability

HBase supports both modular and linear scalability.

- Consistency

Strictly consistent read and write operations enables the use of HBase for high-speed requirements.

- Faster Lookups

HBase offers random access with the help of Hash tables and stores data in indexed HDFS files which enables faster lookups. [1] (TEAM DATAFLAIR, 2018)

MongoDB

MongoDB is a document-oriented schema-less database which uses JSON like schema known as BSON to store data. Following are the key characteristics of MongoDB:

- High Performance

MongoDB is a high-performance database which uses embedded data models to increase the efficiency of input/output activities in the system. Support of indexes in MongoDB enables faster query processing.

- Rich Query Language

MongoDB has rich query language for CRUD operations, Parallelly processed queries, Text searches and Queries related to geographical location.

- High Availability

MongoDB uses replica set which keeps the same data in multiple servers that increase data availability and enables automatic failover.

- Horizontal Scalability

MongoDB distributes data in smaller sub zones across multiple clusters providing horizontal scalability.

- MongoDB Indexing

Indexes can be created in MongoDB to improve search queries performance.

[1] https://hbase.apache.org/

- MongoDB Management System (MMS)

MMS is a tool that tracks and monitors database server & machine. It analyses hardware metrics of the machine where MongoDB is to be deployed and helps optimizing deployment as well as has a feature of issue alert which helps in resolving the issue before it affects the MongoDB server.

- Schema Less

As MongoDB is a schema-less database, there is no need to map objects/fields in application to objects/fields in the database and can be easily stored BSON format. [2] (Arora, 2017)

Database Architectures

HBase Architecture

HBase tables are bifurcated into multiple regions wherein each region stores multiple rows of the table. HBase architecture consists of three important components HMaster, Region server and Zookeeper.

1. HMaster

HMaster is the master process of HBase responsible for assigning regions to region server and track health of regions across regions servers.

Few other responsibilities of Hmaster are listed below:

- Manage load balancing of the regions by shifting over strained regions to less busy regions.
- Create/Delete table and column family as well as deal with changes in schema and metadata.
- Supervises failover in HBase.

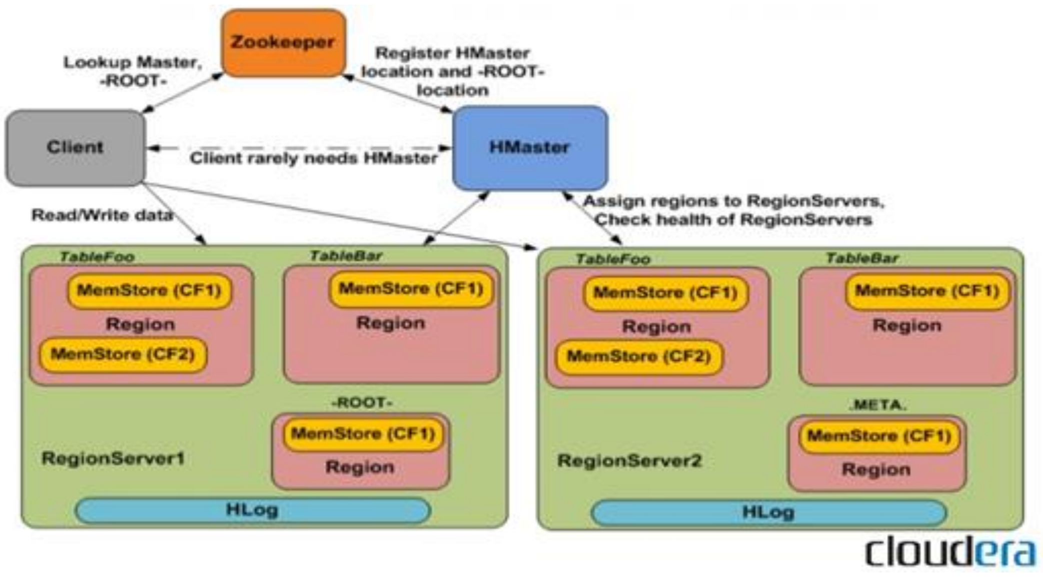

Fig. HBase Architecture [3]

[2] https://docs.mongodb.com/manual/introduction/#key-features
[3] https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295

2. Region Server
   Region server handles all the data related requests from the client. Region server consists of the following components:
   - MemStore:
     It is the write cache which stores the data that are not yet stored in the table and once data is stored in a table the data is flushed from it.
   - Block Cache:
     It is the read cache which data which are most frequently used and whenever the block cache is full recently used data is deleted.
   - HFile:
     HFile is the actual place in HDFS where all data is stored in HBase.

3. Zookeeper
   Zookeeper helps the client communicate with region server and manages all the configuration information and maintains synchronization. It manages server crashes but loading then onto other servers that are working properly. It keeps track of all region servers by keeping information about which datanode is stored in which region server. [4] [5]

MongoDB Architecture

MongoDB is designed by blending capabilities of normal relation database and innovation of NoSQL to meet the requirement of the versatile and big volume of data.

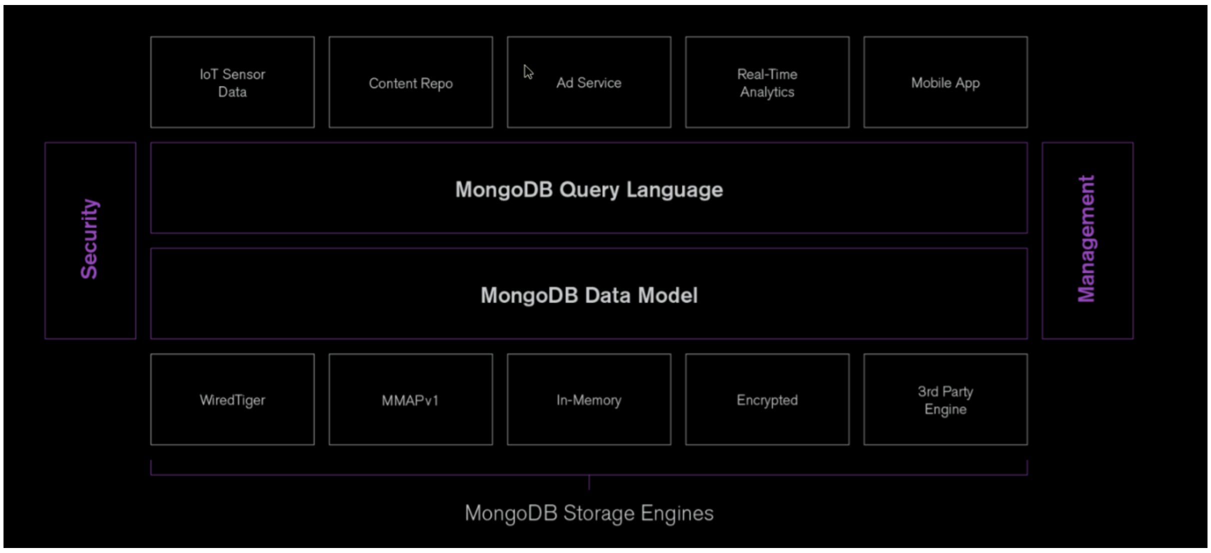

Fig: Flexible Storage Architecture [6]

MongoDB consist of four storage engines which can be easily configured and used as per the requirement. These storage engines can be used on same replica set which provides MongoDB query language, MongoDB data model, security and scaling tools across applications each using different storage engines. MongoDB stores data in binary JSON format, Data from different applications/client is encoded and decoded with the help of the driver. MongoDB has MongoDB Management Systems (MMS) which is used to track the health of all the servers and machine and

[4] https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295
[5] https://www.tutorialspoint.com/hbase/hbase_architecture.htm
[6] https://jira.mongodb.org/secure/attachment/112939/MongoDB_Architecture_Guide.pdf

help in optimizing deployment of instances on the server. MongoDB has various security configuration which enables clients/users to use MongoDB securely.

MongoDB has another architecture called as nexus architecture which is designed to blend best of both relational databases and NoSQL databases and have tried to keep the ease of query language and indexes as in relational database and have combined data availability, redundancy and failover which are modern requirements for the growing big volume of data. This combination of both types of the database makes MongoDB easy to use and integrate it with existing infrastructure and applications.

MongoDB's architecture is designed in a way that it can be used in multiple types of application such as IOT applications, Mobile apps, Real-time analytics application and many more and has special storage engines as per the application type, for example, In-Memory storage engine can be used in real time analytics applications where high performance with low latency is required. [7] [8]

Security

HBase

To protect data access from unauthorised users/ ransomware attacks HBase uses Kerberos, Simple Authentication and Security Layer (SASL) along with Access Control List (ACL).

1. Kerberos

To create a secure connection between the client and HBase server Kerberos is used, it uses secure-key cryptography protocols such as AES, 3DES to check the identity of the client and create a connection. Once the client is validated the communication between client and server can also be encrypted to guarantee privacy and data integrity. Following are the three key steps involved in creating a secure connection between the client and the HBase server.

- Kerberos Authentication: In the first step the client needs to validate its identity to Kerberos authentication server after which user receives a Ticket Granting Ticket (TGT).
- Kerberos Authorization: Once the client receives a TGT, it can request for service and if the TGT is valid it receives a service ticket along with session key for the service.
- Service Request: Now to access the service client authenticates itself using the service ticket at the server.

2. SASL

As HBase runs upon HDFS and requires the help of zookeeper for its operation, there is a need for a secure connection session between the three of them which is done with the help of SASL.

3. ACL

Though the authentication of users is done using Kerberos, there is no control on role level i.e. any user can read, write or delete any table so to authorizes restricted access to specific users HBase uses ACL. However, by default, this feature is disabled in HBase and need to be enabled by changing in configuration file Hbase-site.xml. Once this feature is enabled there are mainly three commands to restrict access to specific users which are:

- grant: It is used to give specific rights to users such as read, write, create, execute, create and admin.
- revoke: It is used to revoke rights from the user.

[7] https://www.quora.com/What-is-the-MongoDB-architecture
[8] https://jira.mongodb.org/secure/attachment/112939/MongoDB_Architecture_Guide.pdf

- user_permission: It is used to list all user permission for a table.

HBase also provides a configuration in Hbase-site.xml file to create a secure connection in Thrift and Rest gateway. (Data Flair Team, 2018) (UpGrad, 2018)

MongoDB

In MongoDB by default configuration related to security is not done and it is very vulnerable to use MongoDB as it is without any security configuration and hence MongoDB provides a list of actions as security checklist required to protect data in MongoDB.

Security Checklist

- Enable Access Control and Enforce Authentication
  MongoDB has a default authentication framework that can be used to validate a user/client based on the credential before they can access the service of the server or an external authentication framework can also be used for the same.

- Configure Role Based Access Control
  Restrict the access of user as per the operations the user should be able to perform, for that MongoDB suggest creating an admin user then give all users least privileges and then assign them only privileges they need to perform their operations.

- Encrypt Communication
  Configure TSL/SSL connection between mongos and mongod components of MongoDB server as well all between clients/application and MongoDB server to prevent unauthorised access and ransom attacks on the server. MongoDB provides libraries as per the operating system on which MongoDB server is deployed.

- Limit Network Exposure
  MongoDB recommends deploying the MongoDB cluster on the secured network only and give access to the network to authorised clients only.

- Audit System Activity
  MongoDB provides a facility where database administrators can track all operations and changes in the system to take necessary actions if and when required. Auditing facility can be enabled in MongoDB using command auditDestination and other configurations can be done as per the requirement such as generating the report to console or JSON file or syslog or BSON file.

- Run MongoDB with a dedicated user
  MongoDB recommends running MongoDB processes using a dedicated authorised operating system user with all required permissions. [9]

HBase vs MongoDB in terms of security

As initially HBase was designed to be used in private cluster in the secured authorised environment it is difficult to incorporate security into the complex framework. Initial versions of HBase didn't have even the HBase authorization and authentication which is available now but, that too is disabled by default and need configuration changes to enable them. The Common Vulnerabilities and Exposures

[9] https://docs.mongodb.com/manual/administration/security-checklist/

(CVE) database which is the repository of all known public vulnerabilities and exposure has given HBase score of 6.3 in the medium risk category.

MongoDB also has issues related to security because of its origin in private data centres and MongoDB now is providing configurations such as access control and encryption to avoid ransom attacks and prevent a data breach. CVE has given MongoDB score of 6 in the medium risk category.

So, both HBase and MongoDB both seems not that secure, and both have vulnerabilities and exposures but there are configurations available in both databases through which precautionary actions can be taken to prevent a data breach and ransom attacks. (UpGrad, 2018)

Literature Survey

Many experiments and research have been done before to compare various NoSQL databases under various environments and workloads, below are few experiments on a comparison of NoSQL databases.

In the paper (Suman Tiwari, 2017) authors have compared three different types of NoSQL databases namely HBase, MongoDB and Neo4J with the help of YCSB in different scenarios and concluded that MongoDB is to be used in systems where the data is changing with requirement and data is loosely coupled and not to be used for transactional systems whereas HBase is recommended for systems where real-time and random read/write access on huge volume of data and for HBase also it is recommended to not use for transactional systems and Neo4J is to be used on systems involving data centre management, fraud detections and real-time recommendations while it is not to be used projects involving big volume of data.

In the paper (End Point Corporation, 2015) benchmarking of Apache Cassandra, Couchbase, HBase, and MongoDB using YCSB for different workloads using YCSB is done. To mitigate instance performance variability each combination database, workload and number of nodes were ran for three times and then the average of results was taken, and the following observations were made.

- For load process where bulk loading of data was done total throughput was compared for all the databases across nodes and in this test, the throughput(operations/sec) for Cassandra was better compared to other databases for a different number of nodes while MongoDB had worst throughput compared to other three.
- For read mostly, balanced write/read, ready-modify-write similar trend was observed.
- For insert-mostly workload performance of Cassandra was better than others and the performance of Couchbase improved as the number of nodes increased.
- For all the workload as expected average latency of Cassandra was least compared to other databases while MongoDB had highest average latency.

In the paper (Ali Hammood, 2016) authors have compared three different types of NoSQL databases namely HBase, MongoDB and Cassandra with the help of YCSB for different workloads. Authors examined throughput in operation/sec and average latency in milliseconds for different loads and observed following things:

- In load phase in which a million records were loaded Cassandra had the highest throughput and MongoDB had the lowest average lowest.
- For read-update workload, HBase performed better compared to other two databases in terms of both average latency and throughput with the lowest average latency and highest throughput.
- For read heavy workload MongoDB performed well compared to others but there was no difference in latency of Cassandra and MongoDB.

- For insert mostly operation HBase had better average latency and there was a marginal difference between MongoDB and HBase but, Cassandra had better throughput whereas MongoDB had very high latency and low throughput.

Performance Test Plan

In order to compare the performance of HBase and MongoDB, the following steps were performed.

- A virtual machine was set up on an OpenStack cloud platform with the configuration of 4GB ram, 40GB disk size and 2 virtual processing unit and Ubuntu as the operating system.
- Hadoop 2.9.2 was downloaded and configured so that it can be used with HBase.
- HBase 1.4.8 was downloaded and configured on top of HDFS.
- MongoDB 4.0.3 was downloaded and configured to run mongod process as background process listening for connections on localhost.
- A new table named as usertable with column family cf1 was created in HBase.
- A new table named as usertable was created in MongoDB.
- YCSB 0.15.0 was downloaded and unzipped and a folder named output was created inside ycsb folder.
- Testharness was downloaded to automate the test runs for different workloads and for different operations on different databases and following steps were followed:
    a. Two workloads workloada which has an equal amount of read and update operations and workloadd which has read-mostly operations were chosen for the test. The configuration for workloads was done in workloadlist.txt file of testharness.
    b. Four operation counts hundred thousand, two hundred thousand, four hundred thousand and half a million were selected for the test which was configured in opcounts.txt file of testharness.
    c. In testdbs.txt name of two databases HBase and MongoDB were configured and respective files to clear/truncate data from tables were updated. Once all configuration was done testharness script was run for three times and the output files were stored in output folder of ycsb.
- Once all configuration was done testharness script was run for three times and the output files were stored in output folder of ycsb.
- Average of all three tests for each operation count, database type and workload were taken to negate variability in results.
- Graphs were plotted to help analyses how these two databases performed for different workloads and operation counts.

Evaluation and Results

YCSB provides its results in properly formatted text files. For each workload the overall throughput, operation (read/update) specific average latency along with operation specific count was gathered. An aggregated average was calculated for the three sets of outputs for each workload, database type, and operation count and the following graphs were plotted.

1. Workload A

I. Average latency vs Read Operations

From the graph plotted below for the workload A which is a good mix of read and update operations following observations can be made:

- Average latency for HBase for read operations has kept on increasing with increase in operation count.

- Average latency for MongoDB has also increased with increased with increase in operation count but not as much as it has for HBase.
- It is considered that smaller the average latency better the performance of the database, from the trend line in below plot it can be clearly observed that MongoDB has outperformed HBase in workload A in terms of average latency for read operations.

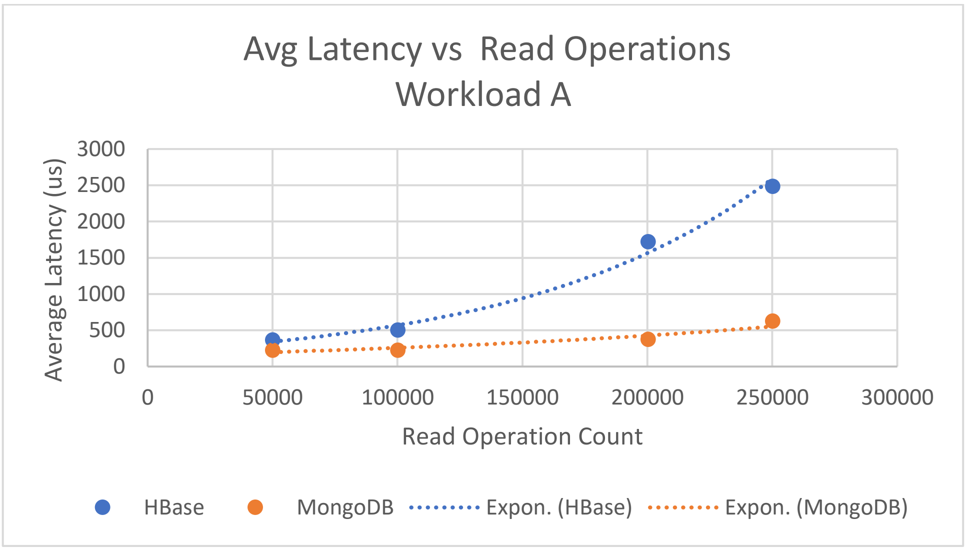

Fig: Avg latency vs Read Operations (Workload A)

II. Average latency vs Update Operations

From the graph plotted below for the update operations following observations can be made:

- There is not much difference in the average latency of the first three operations count but for the last operation count, the difference is quite significant.
- Average latency for MongoDB has kept on increasing with increase in operation count.
- Performance of MongoDB for update operations in workload A is better as compared to HBase for first three operation count but for the last operation count, HBase has performed better with marginal less average latency.

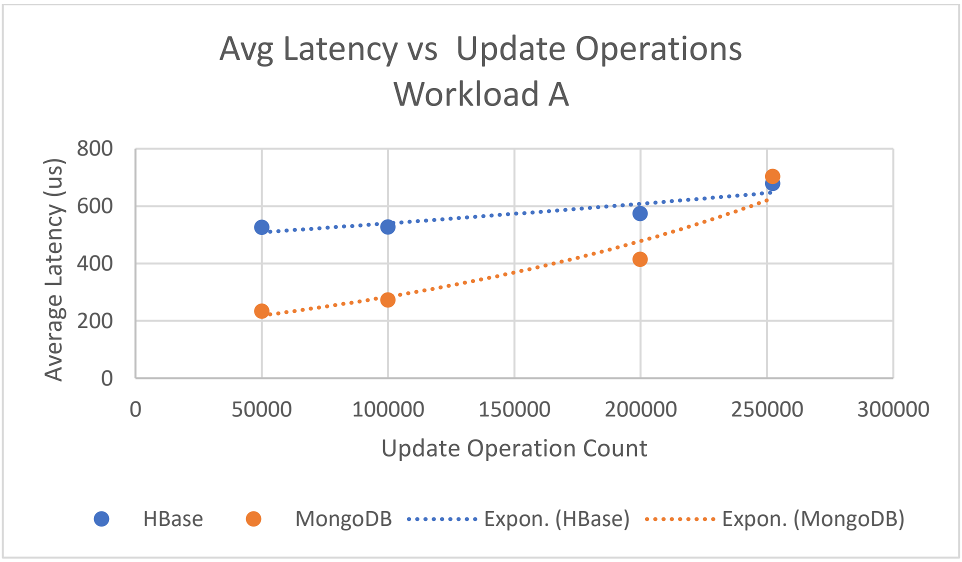

Fig: Avg latency vs Update Operations (Workload A)

III. Total Throughput vs Total Operations

From the graph plotted below for the update operations following observations can be made:

- Throughput for MongoDB has descended with an increase in operation count.
- Throughput for HBase has also decreased with an increase in operation count.
- It is considered that higher the throughput better the performance of the database, from the trend line in below plot it can be clearly observed that MongoDB has outperformed HBase in workload A in terms of total throughput for all operation counts.

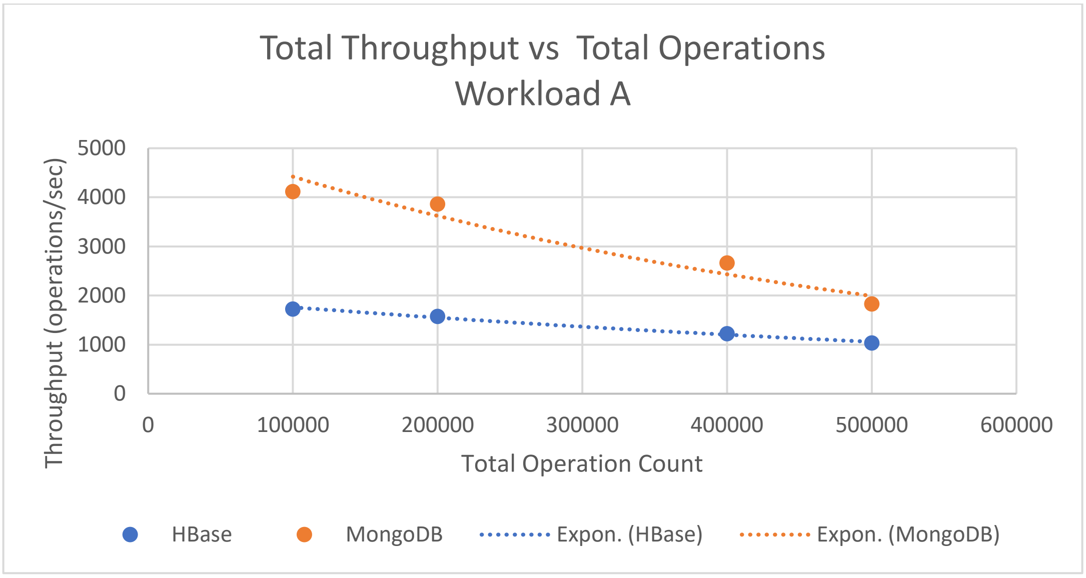

Fig: Total throughput vs Total Operations (Workload A)

2. Workload D

I. Average latency vs Read Operations

From the graph plotted below for the workload D which is read-mostly workload following observations can be made:

- Average latency for HBase for read operations has kept on increasing with increase in operation count.
- Average latency for MongoDB has marginally decreased with increase in operation count.
- From the trend in the below plot, it can be said that MongoDB has way better average latency then HBase for read operations.

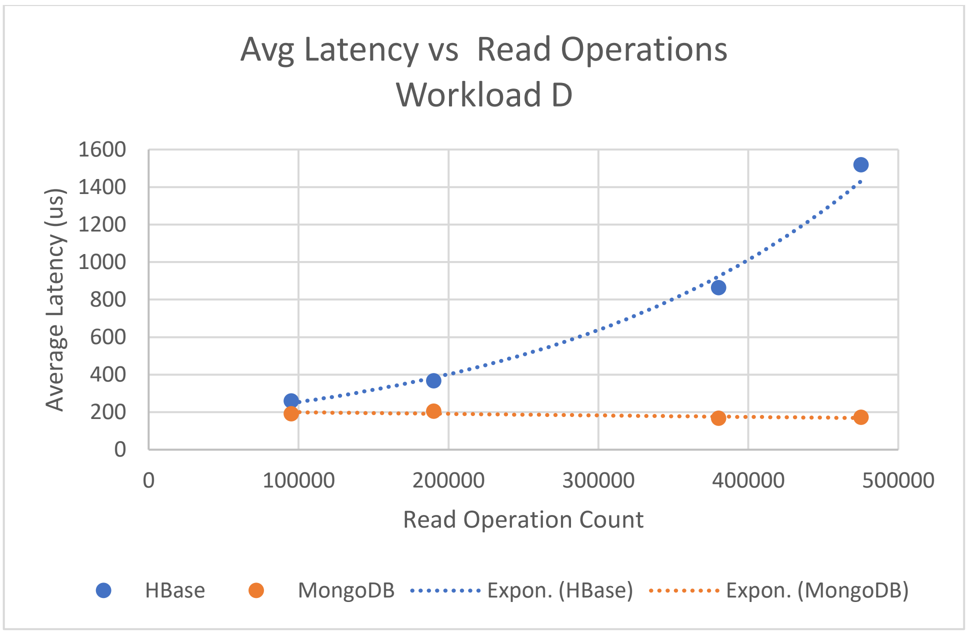

Fig: Avg latency vs Update Operations (Workload D)

II. Total Throughput vs Total Operations

From the graph plotted below for the total operations following observations can be made:

- Throughput for MongoDB has slightly increased with an increase in operation count.
- There is a substantial decrease in throughput for HBase with an increase in operation count.
- From the trend in the below plot, it can be inferred that MongoDB has better performance for workload D in terms of better throughput for all the four operation count.

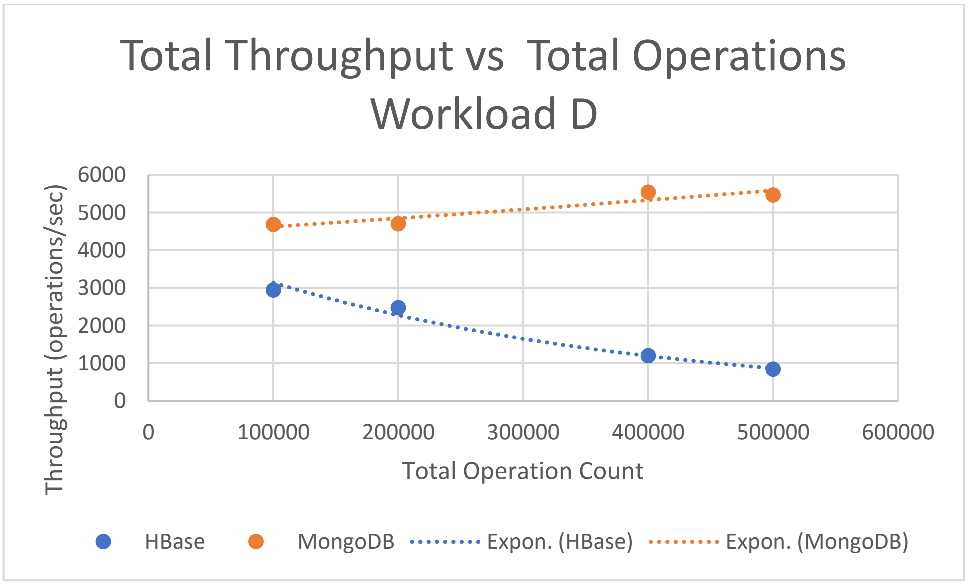

Fig: Total Throughput vs Total Operations (Workload D)

Conclusion and Discussion

In this experiment results were gathered for the comparison of MongoDB and HBase using YCSB with the help of testharness for two different workloads and four sets of operation count. For workload A and first two operation count average latency of MongoDB as well as HBase is almost same for read operations but, with an increase in operation count the difference in latency rate between MongoDB and HBase increase significantly and it can be concluded that MongoDB has better performance in terms of average latency for read operations. For workload, A average latency of MongoDB is better as compared to HBase for first three operations count but for last operation count, HBase has slightly better latency for update operations. For workload, A throughput for both databases decreased with increase in operation count and MongoDB had better throughput compared to HBase. For workload D MongoDB had exceptional average latency rate for all four sets of operation count whereas average latency of HBase kept increasing with increase in operation count, for workload D though the performance of both databases seems same for first operation count they actually difference in performance can be seen for the fourth operation count where the difference is almost 1200 us. For workload D throughput for MongoDB is better as well then HBase.

From all the findings through this experiment for all different sets of combination of workload, database type and operation count in each and every test MongoDB outperformed HBase be it in terms of low average latency or high throughput so, in conclusion for all the test that was performed in this experiment it can say that MongoDB is a better database in comparison with HBase.

Bibliography

Ali Hammood, ,. M. S., 2016. *A Comparison Of NoSQLDatabase Systems: A Study On MongoDB, Apache Hbase, And Apache Cassandra.* Turkey, International Conference on Computer Science and Engineering.

Arora, M., 2017. *WHAT ARE THE KEY FEATURES OF MONGODB?.* [Online]
Available at: https://www.tutorialsjar.com/key-features-of-mongodb/
[Accessed December 2018].

Data Flair Team, 2018. [Online]
Available at: https://data-flair.training/blogs/hbase-security/
[Accessed December 2018].

End Point Corporation, 2015. *Benchmarking Top NoSQL Databases Apache Cassandra, Couchbase, HBase, and MongoDB.* s.l.:End Point Corporation.

HBase, A., 2018. *Apache HBase.* [Online]
Available at: https://hbase.apache.org/
[Accessed 12 2018].

Rouse, M., 2013. *WhatIs.com.* [Online]
Available at: https://whatis.techtarget.com/definition/3Vs
[Accessed 12 2018].

Suman Tiwari, M. K. B., 2017. *Analysis of NoSQL Databases:Mongodb,HBase,Neo4J.* s.l.:International Journal of Engineering Trends and Technology (IJETT) –.

TEAM DATAFLAIR, 2018. *Data Flair.* [Online]
Available at: https://data-flair.training/blogs/features-of-hbase
[Accessed 2018 12].

UpGrad, 2018. *Apache Hadoop vs MongoDB: Which Is More Secure?.* [Online]
Available at: https://www.upguard.com/articles/apache-hadoop-vs.-mongodb-which-is-more-secure
[Accessed December 2018].

Zack, M., 2016. *Big Data: ACID versus BASE Data Stores.* [Online]
Available at: https://www.mindstick.com/Blog/11075/big-data-acid-versus-base-data-stores
[Accessed December 2016].