COMPARISON BETWEEN HBASE AND MONGODB BASED ON THEIR PERFORMANCE ON VARIOUS WORKLOADS

Siddharth Sharma
M.Sc. Data Analytics, School of Computing
X16148371@student.ncirl.ie

INTRODUCTION

With the introduction of high speed internet and smartphones the generation of data has grown exponentially. It has been predicted that by the year 2020 four Tera bytes of data will be consumed by each person on earth. This has lead the world to research and develop real-time (near-real-time) stream processors such as spark, storm, trident etc., and storage systems such as HDFS (Hadoop Distributed File System), HBase, MongoDB, Cassandra etc.; that can process this massive flow of data and store it in a structured way respectively. The storage units are expected to reach Yottabyte (10^{24}) very soon, they have already reached a new high of Peta (10^{15}). Research is going on to tackle this issue and there are systems like Hadoop to perform the necessary task efficiently. High performance analysis and storage is yielded by Hadoop because it performs to its full potential when operating in parallel and distributed database architecture. As we have humongous datasets from real-time data streams, Hadoop helps in processing and storage of such data. In this paper, we discuss two types of NoSQL databases i.e. HBase and MongoDB based on their performance.

HBase

HBase is a column oriented NoSQL database which provides real-time accesses to read & write enormous datasets. It has flexible structured hosting so it can manage large table updates. It can be accessed through Java API, HTTP API & supports number of languages such as C, C++, Java Script, Perl [11].

Key Characteristics of HBase:

- HBase is column oriented database and can process millions of rows and columns [8].
- It is horizontally scalable and is also known to be a column family-oriented database [1].
- HBase is generally used for recovering and storage of data [1].
- Any datatype can be stored irrespective whether it is an integer value, character or a float type value [1].
- HBase presents low latency and faster lookups in case of huge datasets.
- It is schema less data model [3].
- Best of OLAP (Online Analytical Processing) because it is column oriented [2].]
- High Availability.

MongoDB

It is also a NoSQL database model. MongoDB is a document oriented database which is useful to query huge datasets swiftly. It is accessed using JSON and supports number of languages such as Java, Python, Scala [8]. Their no support for usage of triggers.

Key Characteristics of MongoDB:

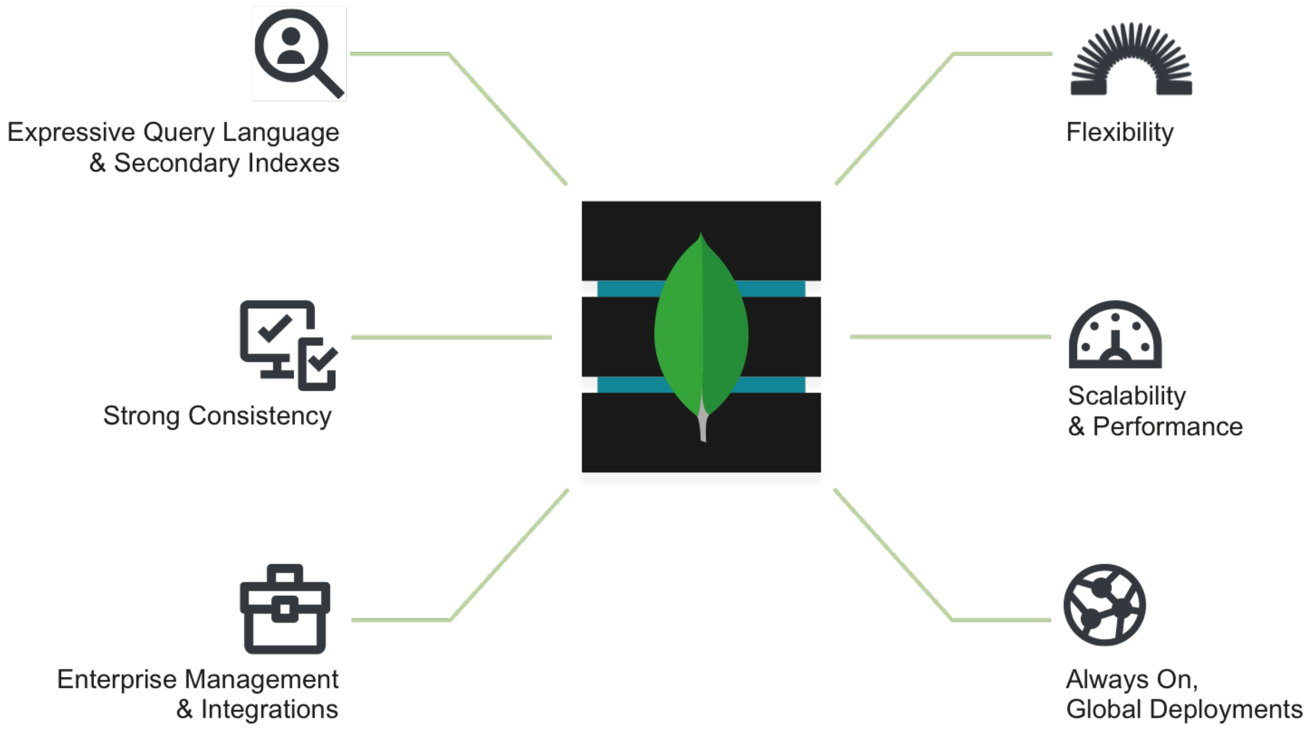

Figure 1[6]

- It is a document oriented data model.
- Master-slave replication is supported which helps in quick integration of data across various applications [3].
- MongoDB is also a schema free database [3].
- It has a distributed data platform which is a very useful in controlling various data centres at different geographical locations [4].
- Secondary Index is also supported by MongoDB which is kept in memory [3].
- As it can be accessed using JSON, updating of data & inserting data is quite easy [3].
- High Availability.

Database architectures of HBase and MongoDB

HBase Architecture and its components:

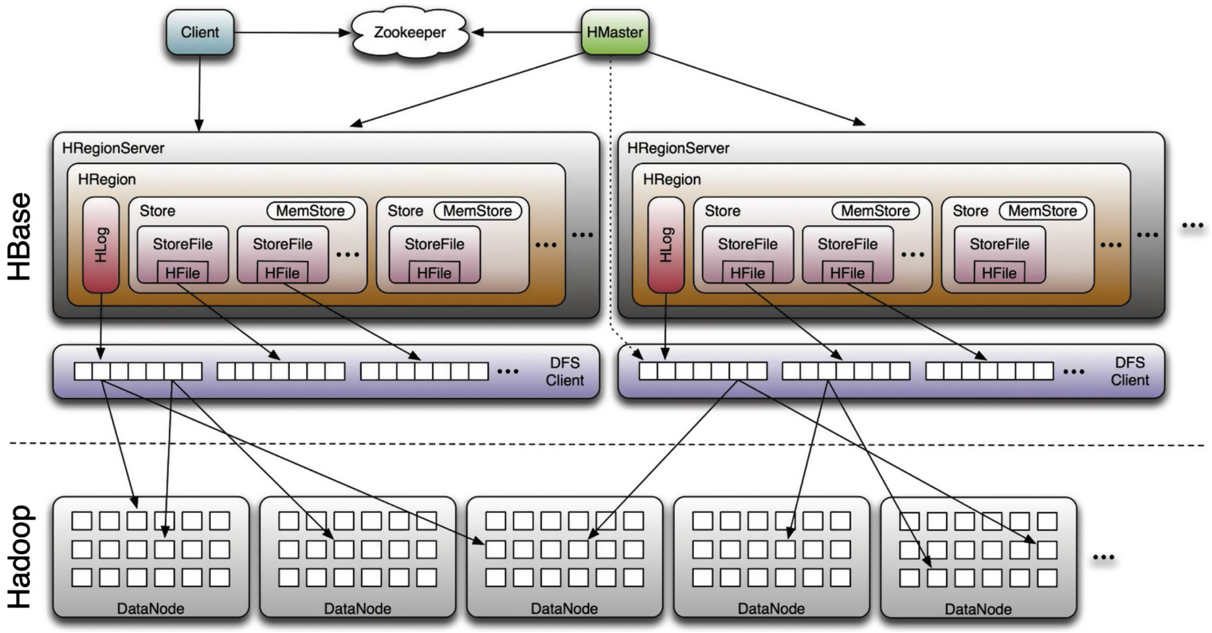

Figure [7]

HMaster

It is the master server and it supervises all the HRegion instances [5]. With the help of Zookeeper, it assigns regions to the region servers as shown in Figure 2. Its plays a very important part in maintaining the cluster nodes. It helps in controlling load balancing and maintains the state of cluster [2][5]. Any change that happens in metadata or the schema HMaster takes the responsibility of such alterations.

HRegionServer

It communicates with the client i.e. receives read and write operations [2][5]. It can directly contact the client and resize the region according to the region size thresholds [2]. Initially all the data is stored here. It can split, host and manage the regions directly without any permission needed form HMaster [5].

HRegion

This is the initial block where all the cluster is setup. It has all the column families and distributed tables building the cluster [5].

Zookeeper

It acts as a streamline monitoring server and contains all the information about the various configurations, naming's and renders distributed synchronization [2][5]. It is the medium through which a client can communicate with the region server. Zookeeper contains ephemeral nodes which help the HMaster to check available servers, these nodes can further help in tracking failures and network partitions [2][5].

Mongo DB Architecture and Components:

MongoDB allows various organisations to utilize multiple storage engines with a single deployment. MongoDB's flexible architecture allows storage of different types of data in a secure way and is consistent in performing such operations on large scale. MongoDB is the most popular and upcoming database because it is combing the capabilities of RDBMS (Relational Database Management Systems) with the features of NoSQL database which will be fruitful for real-time data streaming management.

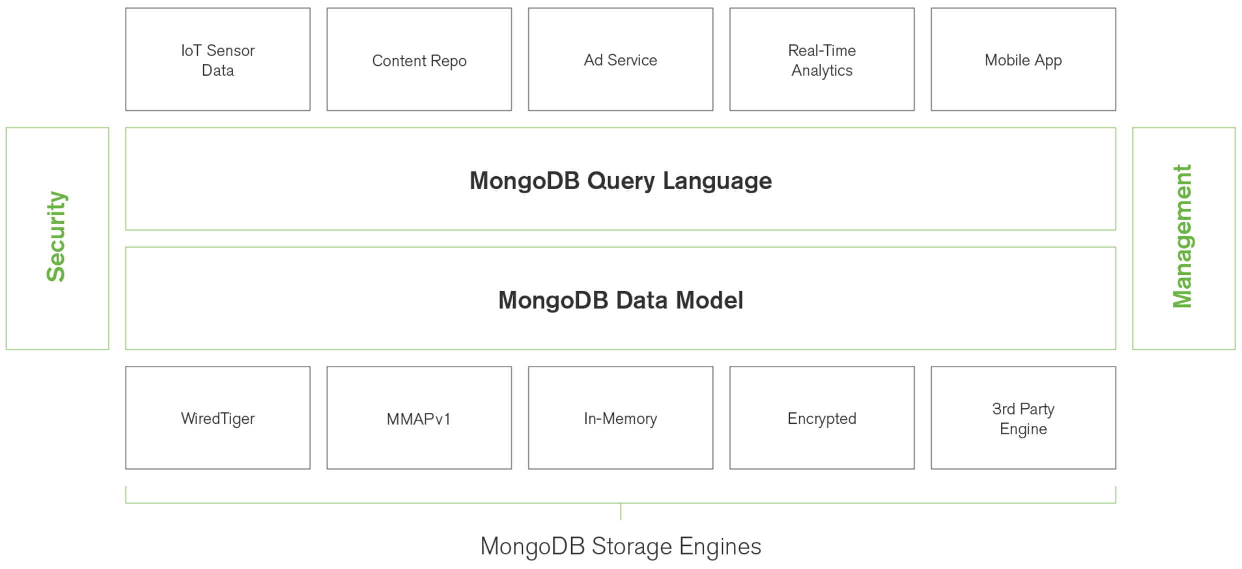

Figure 3[6]

Four types of Storage Engines are supported by MongoDB:

WiredTiger storage engine is used for all round performance and storage efficiency [6].

MMAPv1 is an improved version of the original storage engine used in pre-3.x MongoDB releases.

In-Memory storage engine provides extreme throughput and predictable latency as per the requirement of various organisations.

Encrypted storage engine helps in securing highly sensitive data with top-notch encryption.

Scalability, Reliability and Availability

Scalability is a measure to analyse the performance of the system on which the operations have to be performed and end results need to be produced. Reliability is a measure to check the failure of nodes at any level of the architecture. Availability is a measure to check capability of the system, whether it can perform a certain operation in time or not.

HBase

It is a column based database architecture which is horizontally scalable as it can handle billions of rows and millions of columns, providing row level consistency on a higher level. It provides immediate consistency as it is schema free.

MongoDB

It is a document based database architecture. It enhances the insertion, deletion, update functionalities. It uses master-slave replication method for amalgamation of data which is better for durability of the database. This functionality tells about the reliability of MongoDB i.e. if one replica fails, secondary replica can be used to continue the operations.

Performance Test Plan run HBase and MongoDB

The tool used for benchmarking the storage architectures HBase and MongoDB is YCSB (Yahoo! Cloud Service Benchmark). It is used for checking scalability and performance of the storage databases. I have used an automation tool - testharness which helps in running different operations on HBase and MongoDB sequentially with the help of the script (runtest.sh) provided to us. I have used different workloads, operation counts for the evaluation of the storage databases i.e. MongoDB and HBase based on their performance. OpenStack Ubuntu Virtual Machine server that was provided to estimate the difference between both the databases: MongoDB & HBase on an instance adding a floating IP and then accessing the Virtual Machine through Terminal (Mac OS). The following operations were performed on the OpenStack Virtual Machine:

1. Insert Operation
2. Read Operation
3. Update Operation
4. Clean-up Operation

The performance of the storage databases depends on the configuration of the machine. There is the configuration I have used to perform benchmarking test.

Device Specification
- ➢ Machine Name: Apple MacBook Pro
- ➢ RAM: 16 GB 1600 MHz DDR3
- ➢ Processor: 2.2 GHz Intel Core i7
- ➢ Operating System: macOS High Sierra

Virtual Machine
- ➢ Operating System: OpenStack
- ➢ RAM: 4GB

Database Requirements
- ➢ HBase: Hadoop & HDFS
- ➢ MongoDB: Hadoop & HDFS

Benchmarking Tool:
- ➢ YCSB – v0.11.0

Workloads

Workload A:

- ➢ Read/update ratio: 50/50
- ➢ Default data size: 1KB records (10 fields,100 bytes each, plus key)
- ➢ Request Distribution: zipfian

Workload B:

- ➢ Read/update ratio: 95/5
- ➢ Default data size: 1KB records (10 fields,100 bytes each, plus key)
- ➢ Request Distribution: zipfian

Operation and Read counts

- ➢ Operation Count 100000
- ➢ Operation Count 200000
- ➢ Operation Count 500000
- ➢ Operation Count 600000

Evaluation and Results

Different scenarios' have been used to perform the benchmarking test.

Test 1: Relating both the storage databases on Average Latency versus the number of record read operations for Workload A and Workload B

Result:

Parameters used for workload A:

Workload A	
readproportion	0.5
updateproporation	0.5
Scanproportion	0
Insertproportion	0

Table 1

Read Operation:

We have compared the average latency against the record read operations for both the databases: MongoDB and HBase for workload A. As shown in the Figure 4, Average Latency of HBase and MongoDB increases as the number of Operation count increases. MongoDB is delivering an improved result than HBase. Therefore, MongoDB is a better option than HBase for record read operations when tested on Workload A

	HBase		MongoDB	
Operation Count	Read Operations	Average Latency	Read Operations	Average Latency
100000	49975	289.0208904	49946	115.2645057
200000	99971	340.9320403	99969	116.2377237
500000	249637	435.6508611	250204	155.1440025
600000	300028	463.1106963	300296	371.0843534

Table 2

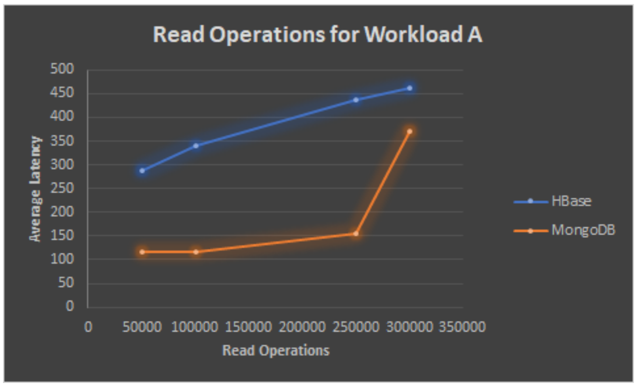

Figure 4

Parameters used for Workload B:

Workload B	
readproportion	0.95
updateproporation	0.05
Scanproportion	0
Insertproportion	0

Table 3

Read Operation:

We have compared the average latency against the record read operations for both the databases: MongoDB and HBase for workload B. As shown in the Figure 5, Average Latency of HBase and MongoDB increases as the number of Operation count increases. MongoDB is delivering an improved result than HBase. Therefore, MongoDB is a better option than HBase for record read operations when tested on Workload B.

	HBase		MongoDB	
Operation Count	Read Operations	Average Latency	Read Operations	Average Latency
100000	95008	244.5405334	95051	113.0331401
200000	190189	261.2537739	190134	103.30823
500000	474925	359.6745549	475010	144.3153155
600000	569966	379.2734777	569999	212.8460699

Table 4

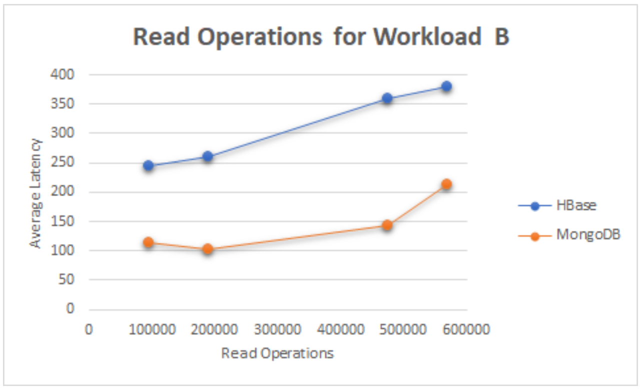

Figure 5

Test 2: Relating both the storage databases on Average Latency versus the number of record update operations for Workload A and Workload B

Result:

Parameters used for Workload A

Workload A	
readproportion	0.5
updateproporation	0.5
Scanproportion	0
Insertproportion	0

Table 5

Update Operation:

We have compared the average latency against the record update operations for both the databases: MongoDB and HBase for workload A. As shown in the Figure 6, Average Latency of HBase and MongoDB decreases as the number of Operation count increases. HBase is delivering an improved result than MongoDB. Therefore, HBase is a better option than MongoDB for record update operations when tested on Workload A

	HBase		MongoDB	
Operation Count	Update Operations	Average Latency	Update Operations	Average Latency
100000	50025	368.7067466	50054	1258.198845
200000	100029	327.6869408	100031	901.0443662
500000	250363	332.8552781	249796	810.4262278
600000	299972	323.4565826	299704	1197.650592

Table 6

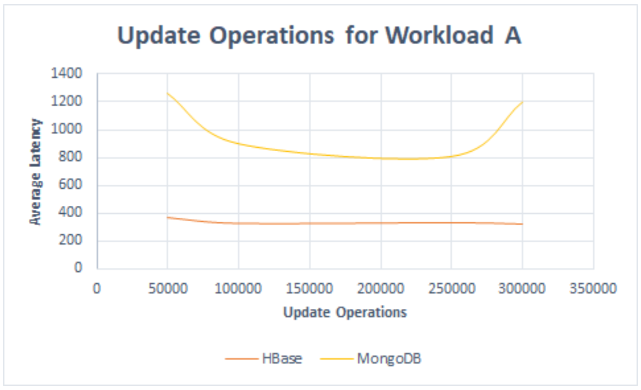

Figure 6

Parameters used for Workload B

Workload B	
readproportion	0.95
updateproporation	0.05
Scanproportion	0
Insertproportion	0

Table 7

Update Operation:
We have compared the average latency against the record read operations for both the databases: MongoDB and HBase for workload A. As shown in the Figure 7, Average Latency of HBase decreases and for MongoDB it decreases as the number of Operation count increase. HBase is delivering an improved result than MongoDB. Therefore, HBase is a better option than MongoDB for record update operations when tested on Workload B.

	HBase		MongoDB	
Operation Count	Update Operations	Average Latency	Update Operations	Average Latency
100000	4992	619.6141827	4949	1301.845019
200000	9811	524.7925798	9866	1676.097203
500000	25075	496.8593021	24990	2255.312965
600000	30034	448.6232603	30001	1936.39662

Table 8

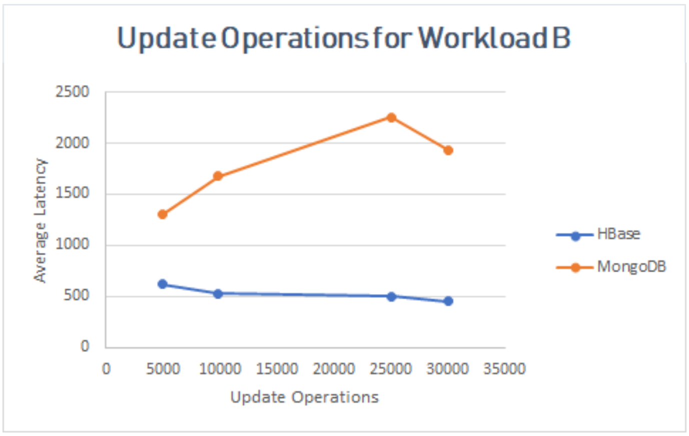

Figure 7

Test 3: Relating both the storage databases based on the overall throughput for Workload A and Workload B

Result:
Parameters used for Workload A

Workload A	
readproportion	0.5
updateproporation	0.5
Scanproportion	0
Insertproportion	0

Table 9

Overall Throughput:
We have compared the Throughput for both the databases: MongoDB and HBase for workload A. As shown in the Figure 8, overall throughput of HBase and MongoDB decreases as the number of Operation count increases. HBase is delivering an improved result than MongoDB. Therefore, HBase is a better option than MongoDB for overall throughput when tested on Workload A

	HBase	MongoDB
Operation Count	Throughput	Throughput
100000	2853.63696	1436.306968
200000	2770.581961	1944.18252
500000	2546.239713	2055.152061
600000	2487.366252	1267.009604

Table 10

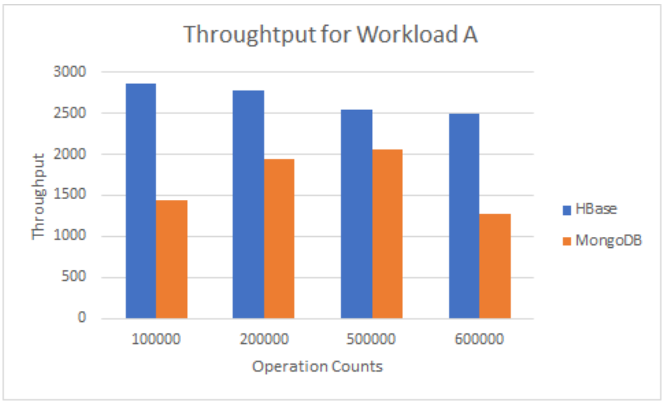

Figure 8

Parameters used for Workload B

Workload B	
readproportion	0.95
updateproporation	0.05
Scanproportion	0
Insertproportion	0

Table 11

Overall Throughput:
We have compared the Throughput for both the databases: MongoDB and HBase for workload A. As shown in the Figure 9, overall throughput of HBase and MongoDB decreases as the number of Operation count increases. MongoDB is delivering an improved result than HBase. Therefore, MongoDB is a better option than HBase for overall throughput when tested on Workload B.

	HBase	MongoDB
Operation Count	Throughput	Throughput
100000	3560.492772	5527.304886
200000	3240.17821	5351.456934
500000	2662.180006	3938.465416
600000	2559.552249	3292.88572

Table 12

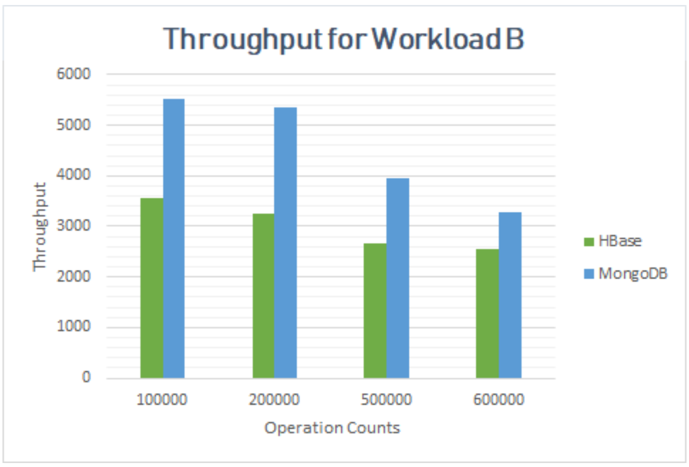

Figure 9

<u>Test 4:</u> Comparing both the storage databases for 95th Percentile Latency and 99th Percentile Latency for Workload A and Workload B

<u>Result:</u>

Parameters used for Workload A

We have compared 95th Percentile Latency and 99th Percentile Latency for both the databases: MongoDB and HBase for workload A. As shown in the Figures below, 95th Percentile Latency and 99th Percentile Latency of HBase decreases and MongoDB increases as the number of Operation count increases. But MongoDB is delivering an improved result than HBase. Therefore, MongoDB is a better option than HBase for 95th Percentile Latency and 99th Percentile Latency tested on Workload A for extreme operation count values as seen in figure 11. Whereas for 95th Percentile Latency and 99th Percentile Latency of lower operation count values HBase gives a better result as seen in figure 10. So, a definite conclusion can't be made whether HBase is Better or MongoDB.

Workload A	
readproportion	0.5
updateproporation	0.5
Scanproportion	0
Insertproportion	0

	HBase		MongoDB	
	95thPercentileLatency	99thPercentileLatency	95thPercentileLatency	99thPercentileLatency
100000	482	948	247	463
200000	439	714	245	397
500000	435	640	245	2269
600000	434	692	2027	5047

Table 13

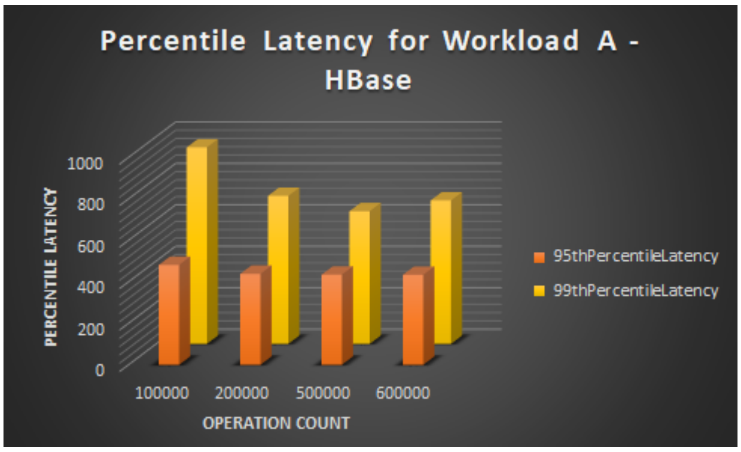

Figure 10

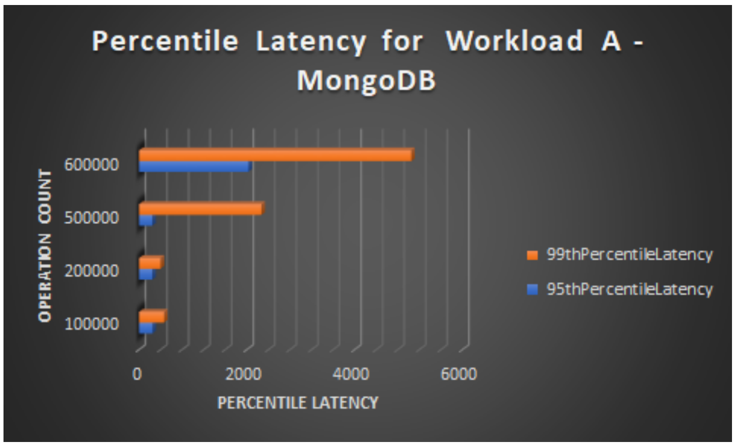

Figure 11

Parameters for Workload B

Workload B	
readproportion	0.95
updateproporation	0.05
Scanproportion	0
Insertproportion	0

Table 14

We have compared 95th Percentile Latency and 99th Percentile Latency for both the databases: MongoDB and HBase for workload B. As shown in the Figures below, 95th Percentile Latency and 99th Percentile Latency of HBase and MongoDB increases as the number of Operation count increases. But MongoDB is delivering an improved result than HBase. Therefore, MongoDB is a better option than HBase for 95th Percentile Latency and 99th Percentile Latency tested on Workload B.

	HBase		MongoDB	
	95thPercentileLatency	99thPercentileLatency	95thPercentileLatency	99thPercentileLatency
100000	1175	5387	400	6591
200000	810	3635	386	14055
500000	712	3065	394	15519
600000	682	3091	2589	14455

Table 15

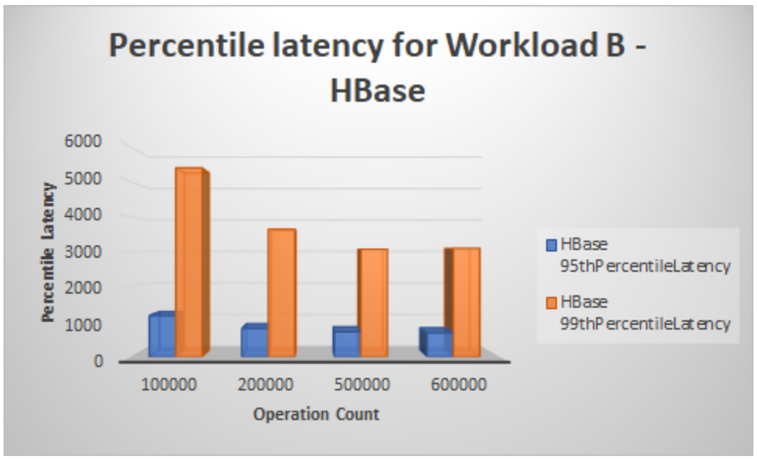

Figure 12

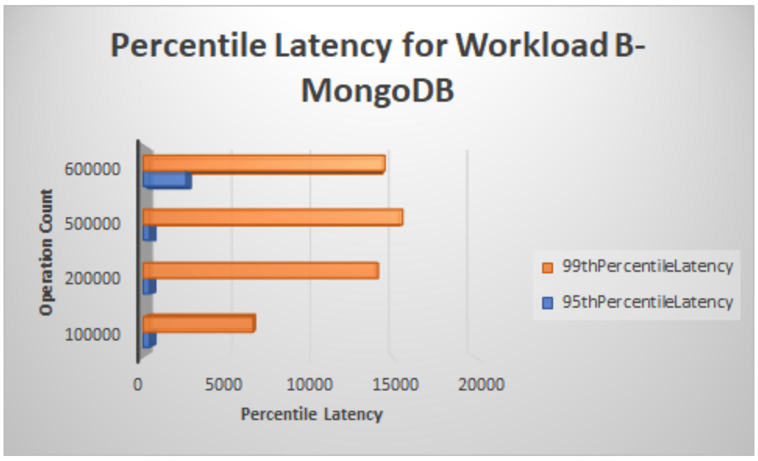

Figure 13

Conclusion:

As seen from the above evaluation for HBase and Mongo DB for Workload A and Work Load B. Conclusions can be made that MongoDB is a better option than HBase when tested on Workload A whereas for Workload B, MongoDB is a better option than HBase for record read operations. HBase is a better option than MongoDB for record update operations when tested on Workload A and Workload B HBase is a better option than MongoDB for overall throughput when tested on Workload A but for Workload B, MongoDB is a better option than HBase for overall throughput. Therefore, MongoDB is a better option than HBase for 95th Percentile Latency and 99th Percentile Latency tested on Workload A for extreme operation count values Whereas for 95th Percentile Latency and 99th Percentile Latency of lower operation count values HBase gives a better result. So, a definite conclusion can't be made whether HBase is Better or MongoDB but for Workload B MongoDB is a better option than HBase for 95th Percentile Latency and 99th Percentile Latency.

References:

[1] Available: https://www.javatpoint.com/what-is-hbase [Accessed on 11/12/2017].

[2] Available: https://www.tutorialspoint.com/hbase/hbase_overview.htm [Accessed on 12/12/2017].

[3] Available: http://www.oodlestechnologies.com/blogs/Comparison-Between-MongoDB-and-Hbase [Accessed on 13/12/2017].

[4] Available: https://www.mongodb.com/mongodb-architecture [Accessed on 14/12/2017].

[5] Available: https://www.guru99.com/hbase-architecture-data-flow-usecases.html [Accessed on 11/12/2017].

[6] Available: https://www.mongodb.com/mongodb-architecture [Accessed on 12/12/2017].

[7] C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sci. (Ny).*, vol. 275, pp. 314–347, 2014.

[8] Available: http://www.oodlestechnologies.com/blogs/Comparison-Between-MongoDB-and-Hbase [Accessed on 11/12/2017].