

Name: Vaibhav Aher

Student ID X18104215

## Abstract

Every day a very large amount of data is being created, in the past few years the data generated are many times the data generated before. Traditional databases are not capable to handle and manipulate huge data. The solution for this is the non-relational database (NoSQL) which provides huge capacity with better performance for unstructured data. There are different types of non-relational databases and each one has its own different feature. Different databases give different performance depends on factors like different workloads, node capacity and the number of cores. To verify how the performance is for different databases, it is necessary to compare their performance for different workloads.

## Introduction:

This paper is about performance analysis of HBase and MongoDB which are most widely used non-relational database management systems. This study shows the performance comparison for the outputs of HBase and MongoDB which is obtained by applying different workloads. This comparison will help users to choose the database as per the requirements. These databases support big data applications. For performance analysis, YCSB (Yahoo Cloud Serving Benchmark) is used (Cattell, 2010). Horizontal scalability is One of the key features of NoSQL databases. Along with horizontal scalability, it also has the capability to manage a large amount of data. Due to this feature NoSQL databases are one of the leading cloud platforms

## Key Characteristics of Data storage Management systems

### MongoDB

MongoDB is a non-relational database management system in which data is stored in JSON format as a document [2]. One Document can have a different number of attributes. MongoDB performs better in de-normalized form of data. There is no need to pre-define the structure of data which we want to store because MongoDB is schema-less. Document of MongoDB uses JSON object to store the data, In JSON while storing the data there we have to pre-define the data type but MongoDB accepts it as a JSON object. Unlike traditional database It is necessary to define the column type, MongoDB has no such a tight bonding. Relational database are vertically scaled in that comparison another advantage of MongoDB is horizontally scalable, It maintains the performance under larger operations. Horizontal scalability obtains by sharding which boost capacity by adding extra instance whenever required. Sharding means mongo DB splits a single collection in to pieces of shard in to name and value pairs across multiple machines (MongoDB, n.d.)(Intellipaat, n.d.).

### HBase

- Apache HBase is distributed column oriented based database management system which can handle huge multi-structured data with thousands of numbers of columns and rows and it is built on the top of Hadoop file system i.e. HDFS. HBase provides real time read-write access to HDFS. HBase gives minimum latency to fetch single row from huge data. HBase gives high throughput with low input-output latency
- Characteristics of Apache HBase (Dezyre, 2016)
  1. HBase provides the access to java API.
  2. It integrates with Hadoop, both as a source and a destination.
  3. HBase have Schema less approach, no need to define data type for column and defines only column families.
  4. It provides Automatic failure support with the help of zookeeper
  5. It provides fast lookups for huge data.
  6. It provides low latency.

## **Database Architecture**

### **HBase**

HBase is NoSQL horizontally scalable database management system. A row key is present which indicates unique row like in relational database there is primary key which indicates unique records. The row is form from one or more columns. The columns which are in organised from in HBase is known as column family. Rows stored in separate partitions called as Regions. Row is instance of the data in table. There are three main components of Apache HBase architecture

- Master Server
- Regional Server
- Zookeeper

When needed regional server can be added or removed.

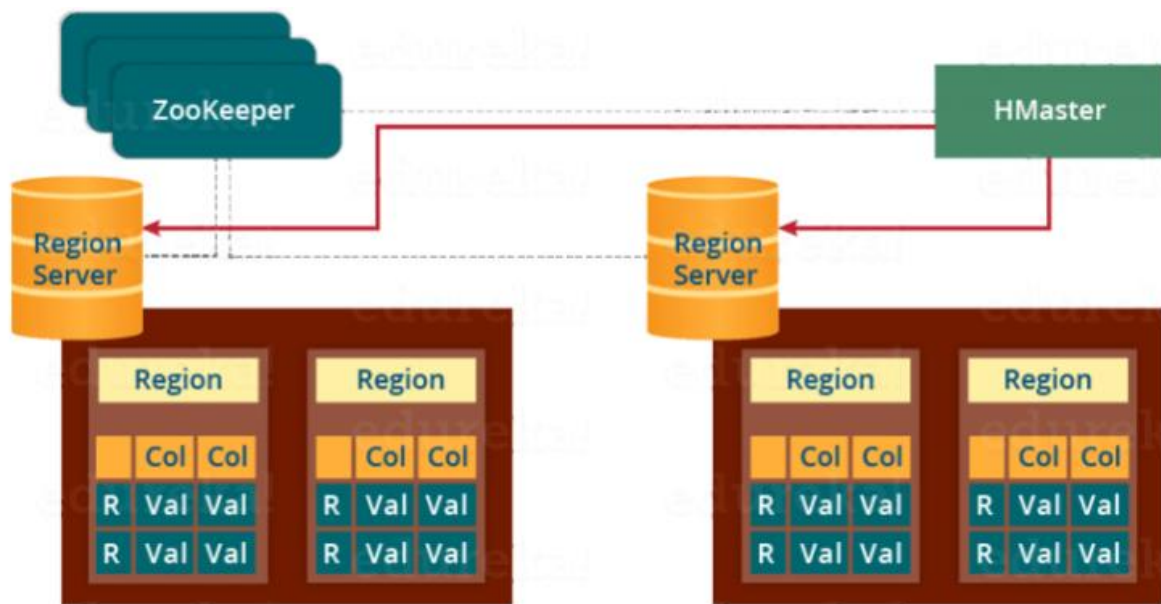


Fig 1 Components of HBase (tutorialspoint, nd)

#### Master server

Master server manages DDL operations by providing the interface to create, update and delete tables. Another functionality is master server allocate the regions to regional server. Like name node manages the data node in HDFS in the same way master server manages the regional server. It uses Zookeeper for managing the regional server instances.

#### Regional Server

Main responsibility of regional server is handle the data related operation by communicating with client. Regional server handles all read write request which comes under it. Another functionality of regional server is to decide the size of region by region size threshold (edureka, nd)

#### Zookeeper

Zookeeper provides services like maintaining configuration information, providing distributed synchronization etc. Master server uses zookeeper nodes to find the available servers. Communication between client with region taken place through the zookeeper. Failure tracking is another responsibility of zookeeper (3pillarglobal., nd)

#### MongoDB

MongoDB is document oriented NoSQL database in which data is in key value format. Basically, the data stored in to binary JSON (BJSON) object which is light weight. It has Multi-Model architecture, there are multiple storage set up in the single environment.

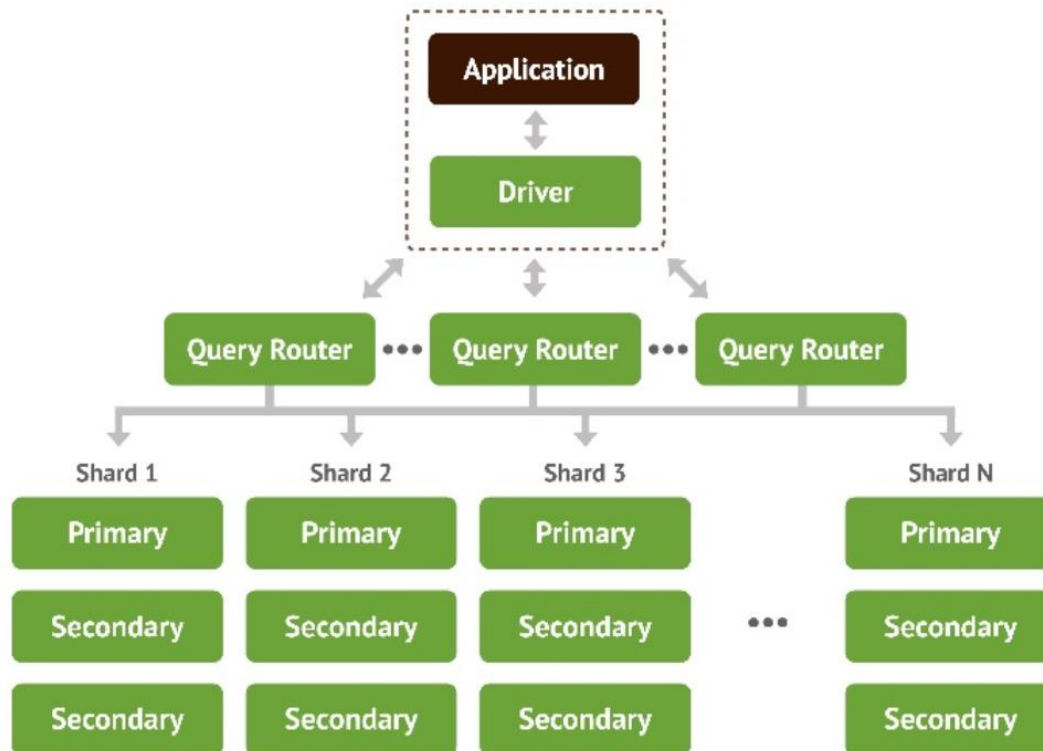


Figure 2. MongoDB Architecture (hortonworks, nd)

There is a replica sets which contains one is primary which is for write and two secondaries for read. The OpLog equalises the data between primary and secondary It ensures that data is same on primary and secondary. Mongod is the primary daemon process for the MongoDB which manages data access and other background management operation. Mongos acts as routing server of shard of MongoDB. Mongos processes the queries and detect the location of data in shard location. There is a interactive JavaScript shell in MongoDB called Mongo. It provides JavaScript environment to use with MongoDB. (dzone, nd)

## I. Scalability

Scalability is ability of the database system to handle the increasing amount of data, Its ability of the database to scale according to workload. The system should be capable of maintaining the performance though the workload varies. There are two types of scalability 1) Vertical scalability 2) Horizontal scalability

**Vertical scalability:** Vertical scalability means adding more CPU power, GPU or ram to the machine to increase the capacity and make it faster. Stalling improves system to handle more number of requests by maintaining the performance. Vertical scaling is costly as compare to horizontal scaling.

**Horizontal scalability:** In this multiple entities are connect to form single unit system. It is economic as compare to vertical scalability. It is partitioning of data. Both MongoDB and HBase are horizontally scalable databases.

## MongoDB

Mongo dB is based on horizontal scaling due to sharding . In MongoDB data is distributed across the machine so it gives high throughput for large data sets also. (slideshare, nd)

## **HBASE**

HBase is also horizontally scalable. Data is distributed region wise in the form of wide columns. Region are made up with the sorted columns data. Data exceeds then it divide in to two parts and for each key is created which is unique value.

## **II. Availability**

### **HBase**

HBase maintains multiple copies of the data in primary as well as secondary regions and which is spread out through the HBase cluster. In worst case if region servers get down or crashed the data can be continuously read from secondary regional server (hortonworks, nd)

### **MongoDB**

MongoDB maintains the availability by using Replicasets replica set use the election to elect the primary member. Initially there is one master node and if system demands new master node which will be selected. We can read the data via secondary during voting which makes MongoDB more available. (MongoDB, nd)

## **III. Reliability**

### **HBase**

HDFS replicates the data across the server and it stores 3 replication for a block on same rack by same node and different rack by different node. This makes HBase highly reliable. So if the the hardware get crashed still the data will be safe due to clustering. (George, 2011)

### **MongoDB**

Sharding provides the fail over safety and also allow data to scale horizontally. Instead of one big server it uses number of small servers. MongoDB has very strong query language which allows us to querying calculations and functions because MongoDB is queried using java script. As the data is stored in document format MongoDB does not required to joins the table which makes it faster. (MongoDB, nd)

## **Performance Test Plan**

For this execution, Operations are performed on Yahoo Cloud Serving Benchmark (YCSB). Inputs are given like type of work load to perform and operation counts to be executed.

In test Harness application, set of operations loaded for both NoSQL database systems.

Open stack instance is used with system specifications 40 GB disk and 4 GB RAM.

Above programs have been added

- Hadoop 2.2.2
- YCSB 0.11.0
- MongoDB 3.2.10
- HBase 1.4.9
- Java Runtime Environment 8
- Java development tool kit 8
- Ssh
- Rsync

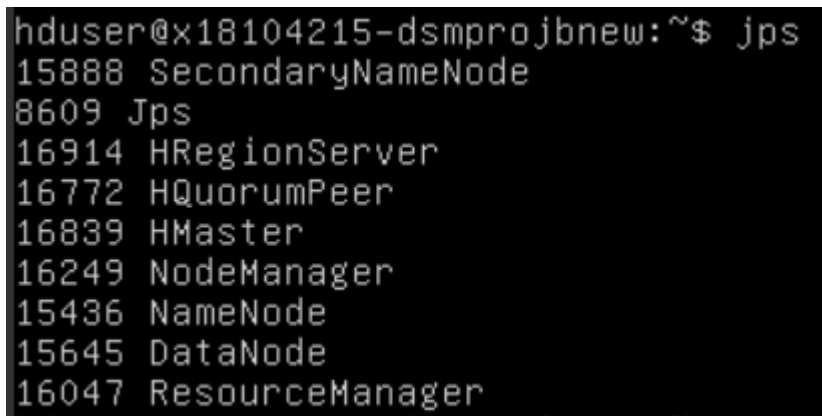
Here HBase is installed on the top of HDFS.

In addition to the mentioned software, the other applications installed were:

- Java Runtime Environment OpenJDK 8
- Java SE Development Kit OpenJDK 8
- Python - 2.7.15-1
- Rsync
- ssh

Once the software is installed, the DFS, Yarn, HBase and Mongo processes are run on the server.

After completing the installation of above software processes for Hadoop, HBase and Mongo run on the server

A terminal window screenshot showing the output of the 'jps' command. The prompt is 'hduser@x18104215-dsmprojbnew:~\$'. The output lists several Java processes: '15888 SecondaryNameNode', '8609 Jps', '16914 HRegionServer', '16772 HQuorumPeer', '16839 HMaster', '16249 NodeManager', '15436 NameNode', '15645 DataNode', and '16047 ResourceManager'.

```
hduser@x18104215-dsmprojbnew:~$ jps
15888 SecondaryNameNode
8609 Jps
16914 HRegionServer
16772 HQuorumPeer
16839 HMaster
16249 NodeManager
15436 NameNode
15645 DataNode
16047 ResourceManager
```

Fig 3: Live process for HBase and Hadoop

In test harness tool the operation count has been put in opcounts.txt the operation counts are as above

- 1) 50000
- 2) 100000
- 3) 150000

In Test harness tool there is a file named workloadlist.txt , the workload which used for the test is put in the file. The selection of workload is as follows:

#### Workload A

- This workload is for Update heavy Workload
- Operation ratio is 50% read and 50% update

#### Workload D

- This workload is read workload
- Operation ratio is 95% read and 5% insert

After running the workload by calling runtest.sh the output is generated in ycsb/Output/test3 path

#### Results

For understanding purpose of the result below mentioned some basic terms

- 1) Operation Count: The number of operations to be performed
- 2) Read count: It is the number of read operations with respect to total operations count
- 3) Insert count: It is the number of insert operations with respect to total operations count
- 4) Average Latency: It is average time taken to send the response from one point to another
- 5) Throughput: Is how much data has been read or inserted.

#### Throughput Comparison

##### Throughput vs Record count for Workload A

	HBase	MongoDB
Workload A	Throughput (ops/sec)	Throughput (ops/sec)
50000	2254	5688
100000	2170	5119
150000	1899	4681
Throughput vs Record count		

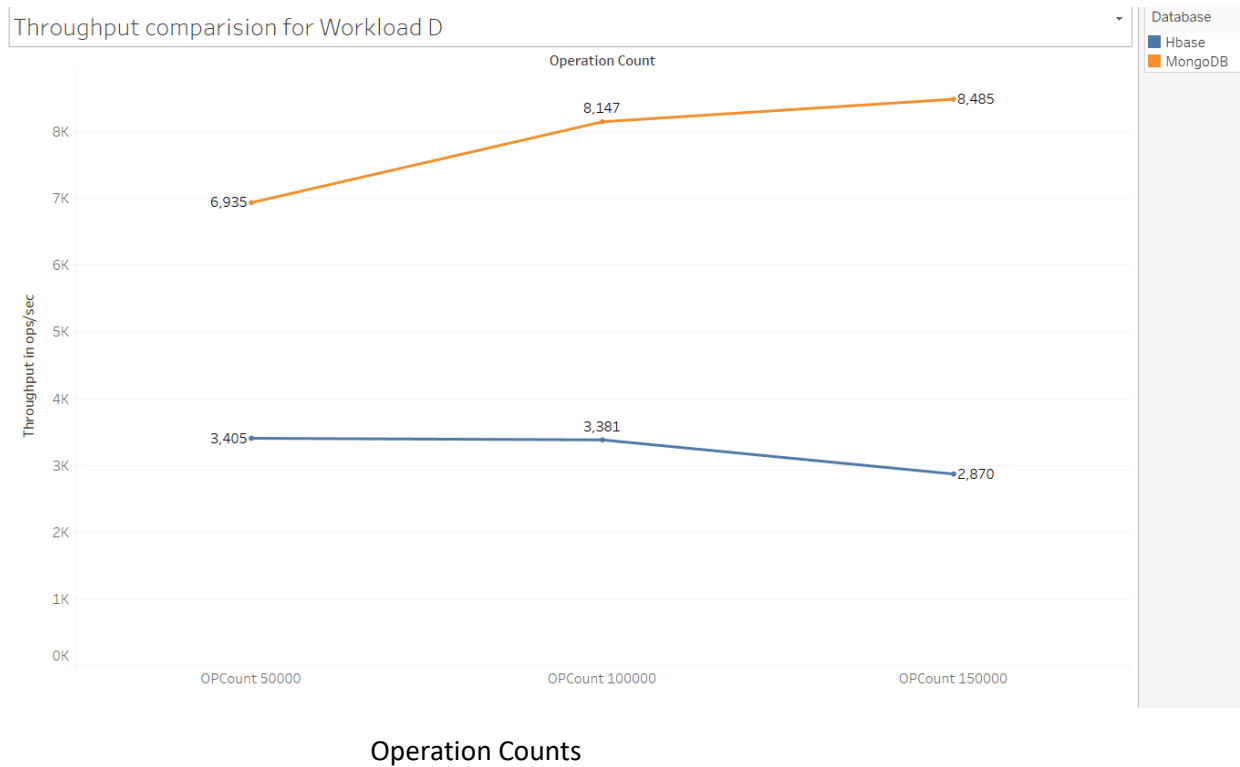


For Workload A it can be seen that as operation count is increasing the throughput for both MongoDB and HBase getting decreasing but for MongoDB its dropping more as compare to HBase , here in this case its difficult to predict the performance of the databases but comparatively HBase performance is better than MongoDB

#### Throughput vs Record count Workload D

	HBase	MongoDB
Workload D	Throughput (ops/sec)	Throughput (ops/sec)
50000	3405	6935
100000	3381	8147
150000	2870	8485
Throughput vs Record count		





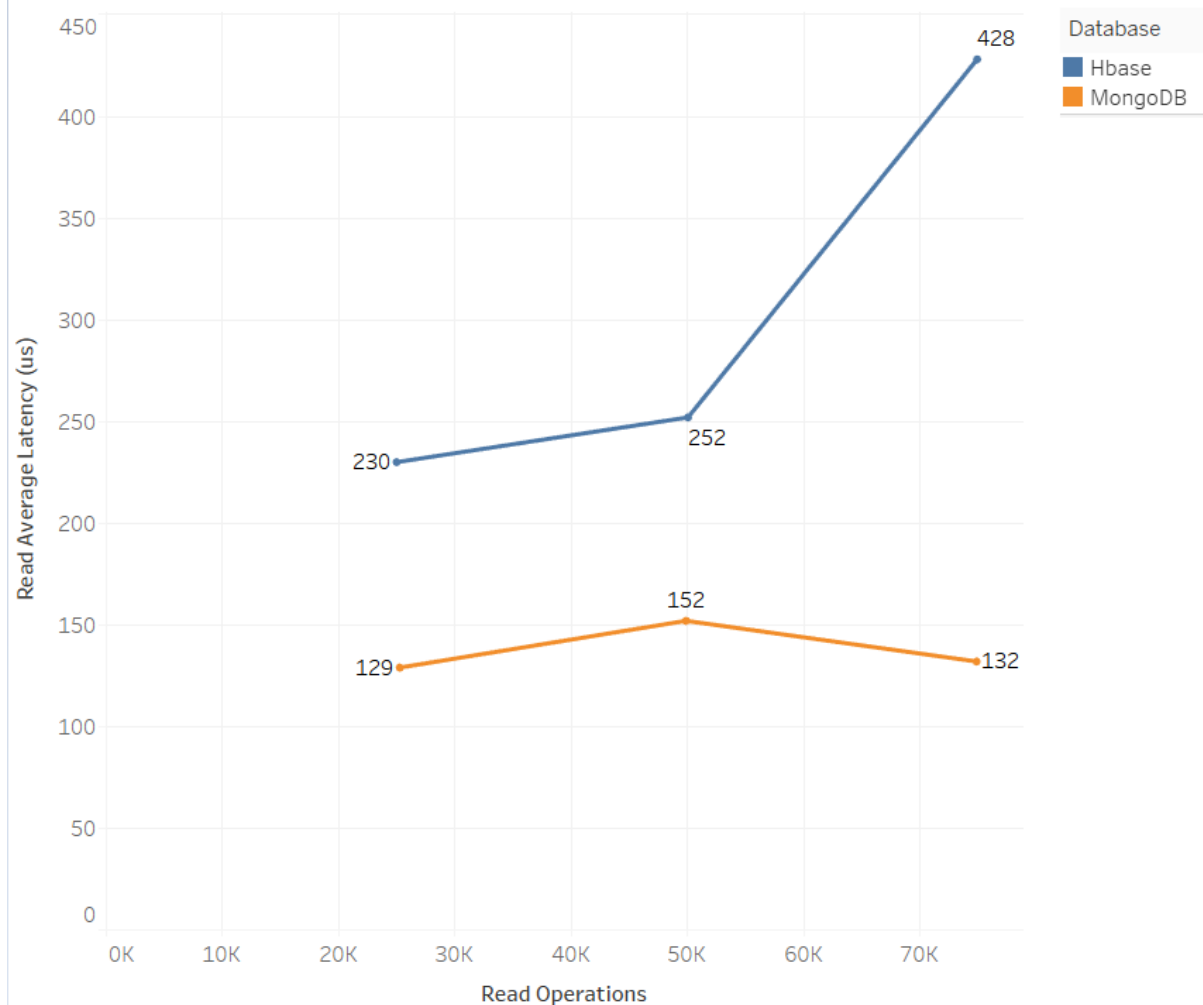
This graph compares the Operations per seconds with Operation counts for MongoDB and HBase for Workload D. X axis represents Operation counts and Y axis represents throughput. As we can see from the graph that As operation counts increases Throughput is increasing for MongoDB but for HBase its getting decrease for same operation count. We can conclude that Performance of MongoDB is better than HBase.

### Read operations vs Read Average Latency (us) For Workload A

Workload A	HBase		MongoDB	
	Read Count	Average Read latency (in us)	Read Count	Average Read latency (in us)
50000	24969	230	25245	129
100000	50042	252	49873	152
150000	74916	428	74869	132

Read operations vs Read Average Latency (us) For Workload A

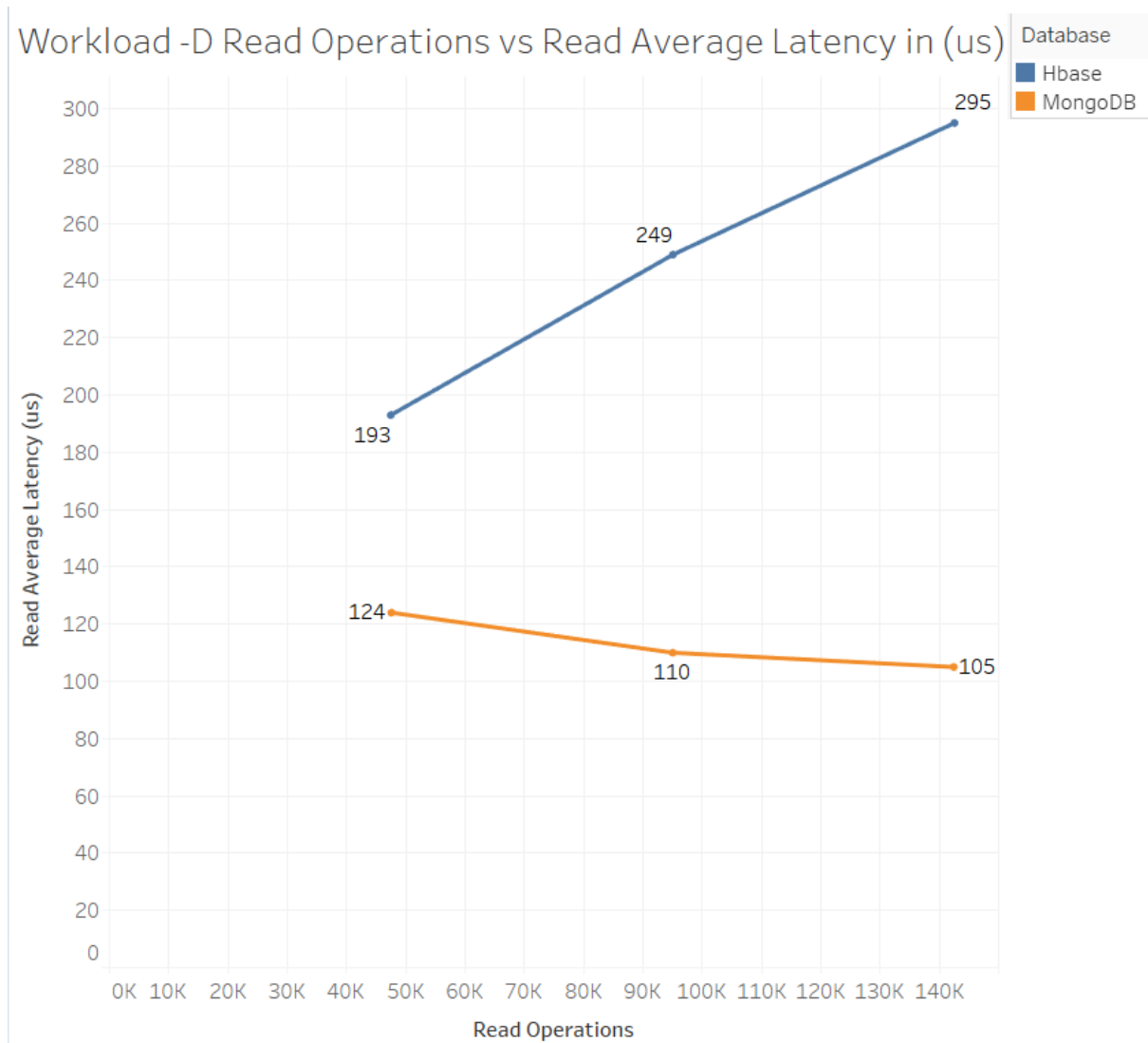
Workload -A Read Operations vs Read Average Latency in (us)



Graph shows read Operations on X axis and Read average latency on Y axis. Here we can see that upto read operation count 50042, the read latency is increasing for both MongoDB and HBase but after that point the latency get decrease for MongoDB which shows that performance of MongoDB is higher than HBase and In case of HBase the average latency is continuously increasing. This may be test were running on different network connection with varying speed.

### Read operations vs Read Average Latency (us) For Workload D

	HBase		MongoDB	
Workload D	Read Count	Average Read latency (in us)	Read Count	Average Read latency (in us)
50000	47454	193	47546	124
100000	95035	249	95016	110
150000	142533	295	142434	105
Read operations vs Read Average Latency (us) For Workload D				



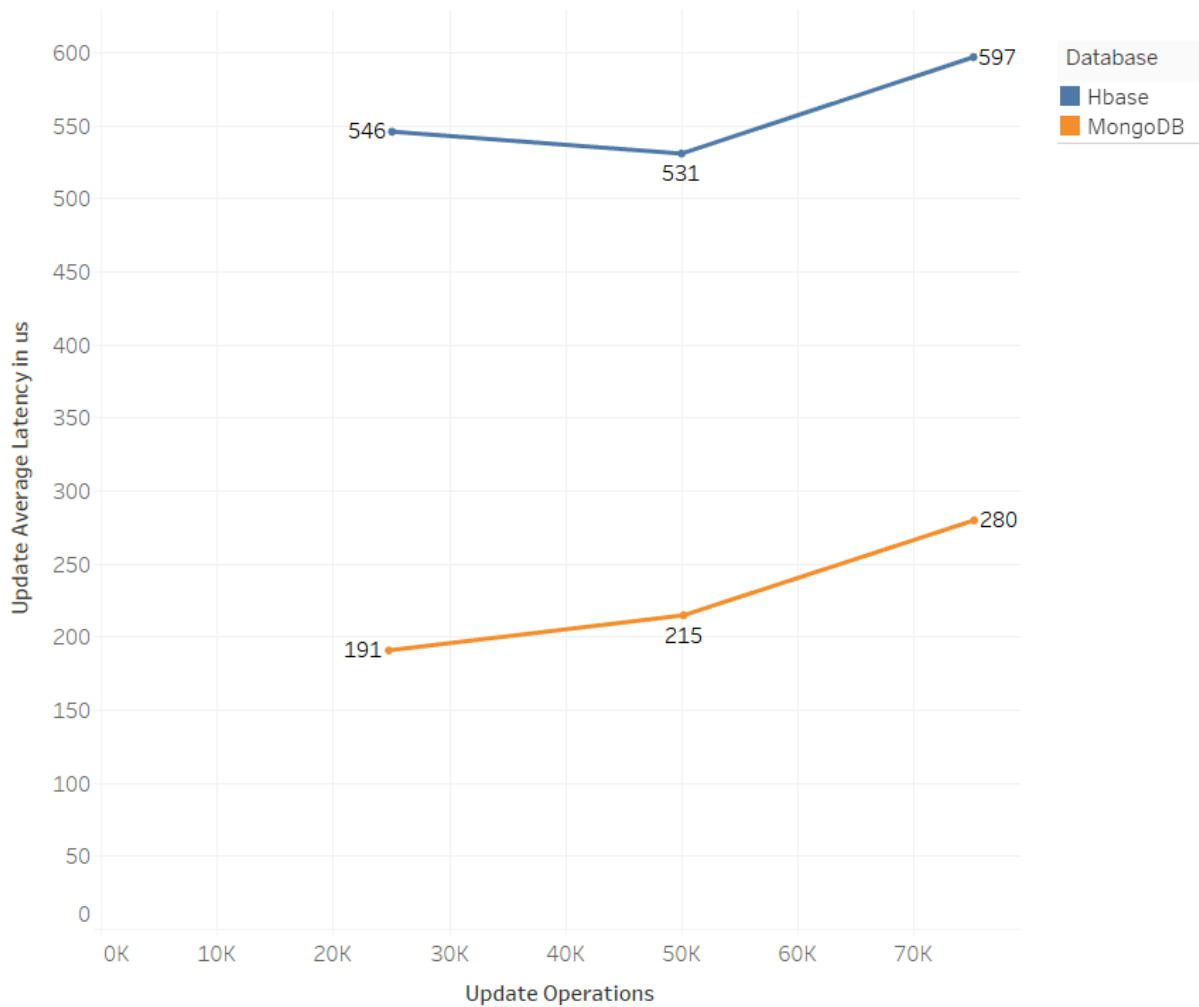
Graph shows read Operations on X axis and Read average latency on Y axis. From graph we can say that for read operation average latency for MongoDB is better than HBase. For MongoDB As Read operation increases the read average latency is decreases but HBase shows exactly opposite behaviour. Conclusion is MongoDB is better as compare to HBase database

### Average Insert Latency vs Insert Count for Workload A

	HBase		MongoDB	
Workload A	Update Count	Average Update latency (in us)	Update Count	Average Update latency (in us)
50000	25031	546	24755	191
100000	49958	531	50127	215
150000	75084	597	75131	280

Average Insert Latency vs Insert Count for Workload A

### Update Operations vs Update Latency

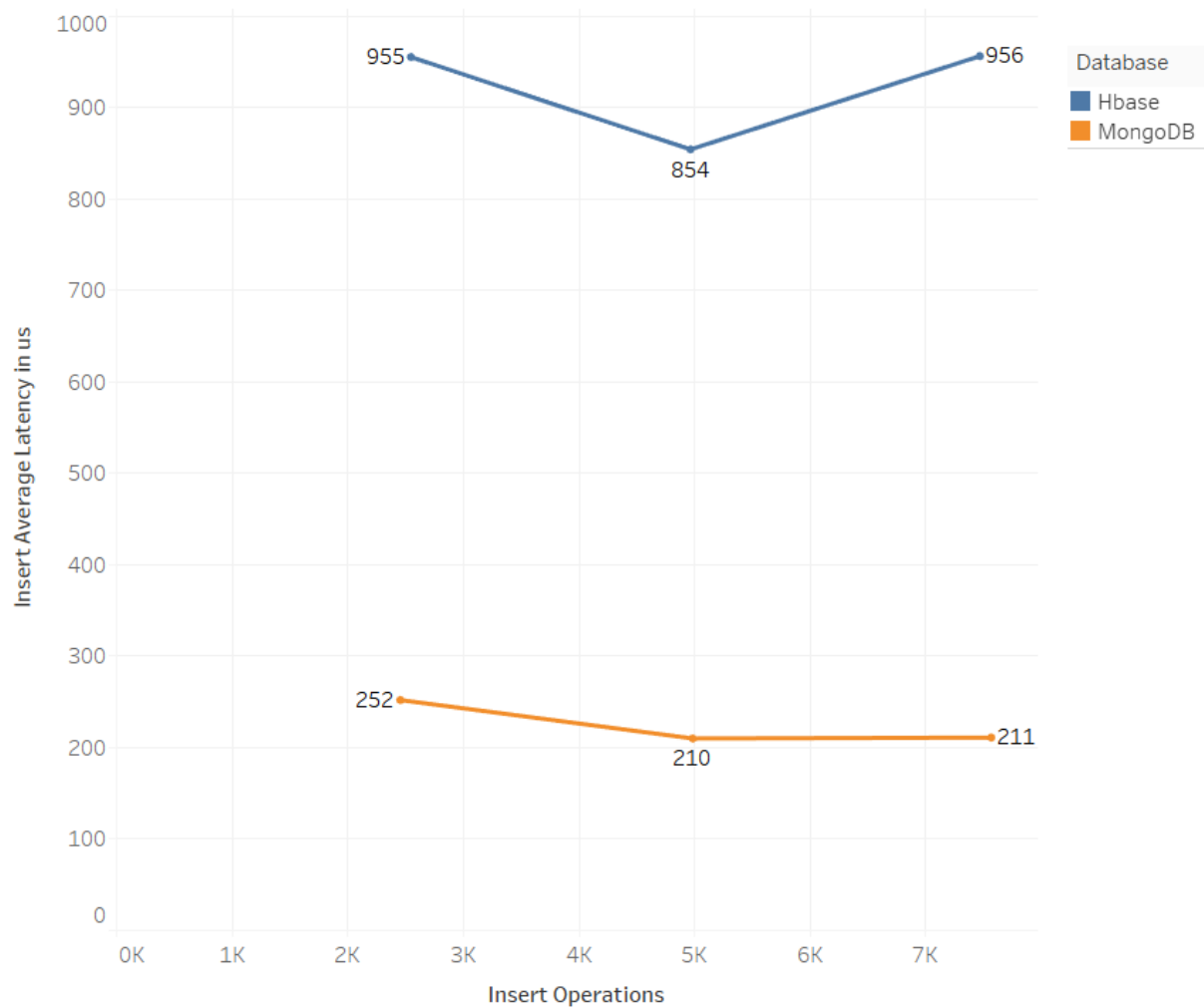


From graph it can be seen that as update operations increases the update average latency is also increases but ideally in NoSQL databases average latency should decrease as operations increases but in this case for HBase and MongoDB update average latency is increasing.

### Average Latency vs Update Count for Workload D

	HBase		MongoDB	
Workload D	Insert Count	Average Insert latency (in us)	Update Count	Average Insert latency (in us)
50000	2546	955	2454	252
100000	4965	854	4984	210
150000	7467	956	7566	211
Average Latency vs Update Count for Workload D				

### Insert Operations vs Insert Latency



The comparison of the graph shows that as Insert operation increases the insert latency is getting decrease but after certain point it again increases. But If we compare the data of MongoDB with respect to HBase. We can say that MongoDB performs better than HBase.

## Conclusion

As the data is increasing day by day, so data management will become the key factor in upcoming future. In this paper we have discuss the architecture and features of HBase and MongoDB databases. We compare HBase and MongoDB on different parameter like Scalability, availability and reliability. HBase is suitable for both structured and non-structured database, as per requirement we can choose the database by considering the factors like security like as compare to MongoDB HBase is more secured, but our experimental analysis shows that the performance of MongoDB is better as compare to HBase. If the data is very huge and it is in key value pair then it is advisable to use HBase instead of MongoDB. Whereas if data is document oriented then its better to use MongoDB. It's up to user for which kind of data he wants to use the database.

## References

3pillarglobal., nd. [Online]

Available at: <https://www.3pillarglobal.com/insights/what-is-MongoDB>

Cattell, R., 2010. Scalable SQL and NoSQL data stores. *SIGMOD Record*, Volume 39, pp. 12-27.

Dezyre, 2016. *Overview of Hbase architecture and its component*. [Online]

Available at: <https://www.dezyre.com/article/overview-of-HBase-architecture-and-its-components/295>

dzone, nd. [Online]

Available at: <https://dzone.com/articles/divide-and-conquer-high-scalability-with-MongoDB-t>

edureka, nd. [Online]

Available at: <https://www.edureka.co/blog/HBase>

George, L., 2011. *HBase: The Definitive Guide*,. 3 ed. nd: O'Reilly Media.

hortonworks, nd. [Online]

Available at: <https://hortonworks.com/blog/apache-HBase-high-availability-next-level/>

Intellipaat, n.d. *HBase Introduction*. [Online]

Available at: <https://intellipaat.com/tutorial/HBase-tutorial/introduction/>

MongoDB, n.d. *MongoDB sharding*. [Online]

Available at: <https://docs.mongodb.com/manual/sharding/>

MongoDB, nd. [Online]

Available at: <https://docs.MongoDB.com/manual/core/replica-set-high-availability/>

slideshare, nd. [Online]

Available at: <https://www.slideshare.net/MongoDB/MongoDB-administration-101>

tutorialspoint, nd. [Online]

Available at: [https://www.tutorialspoint.com/hbase/hbase\\_architecture.htm](https://www.tutorialspoint.com/hbase/hbase_architecture.htm)