National College of Ireland

Data Storage and Management Project

on

Performance Analysis of HBASE and MONGODB

Kaushik Rajan
17165849

MSc Data Analytics – 2018/9

Submitted to: Vikas Tomer

Abstract

The objective of this project is to compare the different NoSQL databases, namely, HBase and MongoDB. The performance comparison is made between the two for the opcounts —12500, 25000, 50000, 75000 and 100000, and for 3 different workloads —Workload A, Workload B and Workload D, using Yahoo Cloud Serving Benchmarking (YCSB) test. The tests were run thrice and the average of the tests were considered to evaluate the performance of both the NoSQL databases at different workloads. The Databases were compared based on Average Latency of Read operation vs the total Throughput, Average Latency of Update operation vs the total Throughput and the throughput time when run with all the three workloads. Output logs taken were used to plot the graph using Tableau.

1 Introduction

Over 2.5 quintillion bytes of data are generated every single day from different platforms. In the last 2 years alone more than 90 % of the worlds data have been generated and it is said to increase multi-fold within the next few years. To store the data that are generated, RDBMS were used. RDBMS has been the data processing dictionary over a period of time and is the basis of SQL. The Traditional databases cannot handle the vast amount and the variety of data that are being generated in the current age. The major factors that are to be considered while picking a database are the 3Vs namely Volume, Velocity and Variety. The traditional transactional database cannot handle the unstructured data and also cannot be used for analytics purpose. Therefore, nowadays, NoSQL databases such as the Hadoop, HBase, MongoDB, Apache Spark, Cassandra are preferred over the traditional databases. Advantages of the NoSQL databases are they are faster, more efficient, better performance and can handle unstructured data. All the NoSQL databases have certain advantages and disadvantages over the other. Choosing the perfect database for the particular work is essential. For this project, the NoSQL databases HBase and MongoDB are compared. Different aspects such as the time taken for the READ operation, time taken for the UPDATE operation, time taken for the INSERT operation and throughputs are compared by subjecting the databases to different workloads and at different Opcounts.

2 Key Characteristics of Chosen Data Storage Management Systems

2.1 HBase

HBase is a open-source, distributed column-oriented database which is created on the top of HDFS, which is written in java, and which is designed after Googles Big Table. Hadoop can only perform batch processing whereas, HBase can access the data randomly and provides fast lookups for larger tables. HBase also has low latency while accessing a single row from billions of records. Since HBase is a column-oriented database, the records are sorted by rows and hence is suitable for Online Analytical Processing (OLAP). HBase also has recovery option as it provides duplication.

Other key features are:

- HBase is linearly scalable.
- Failure support is automatic.
- Provides consistent read and write operations.
- Integrates with Hadoop, both as a source and destination.
- Provides data replication within clusters.

2.2 MongoDB

MongoDB is a cross-platform document-oriented NoSQL database program and is written in C++. It is free and opensource. MongoDB uses JSON like documents. With MongoDB, the user can precisely control where they want to place the data. MongoDB has complete flexibility in deployment and can be easily migrated.

Other key features are:

- Supports Ad hoc queries.
- Can be indexed with primary and secondary indices.
- Easy recovery of data as MongoDB stores multiple replica of the data
- Scales horizontally using sharding.
- The system converts the file into multiple parts and stores each parts into different document.
- Supports map-reduce and aggregation tools.
- High Performance and efficiency.

3 Database Architectures

3.1 HBase

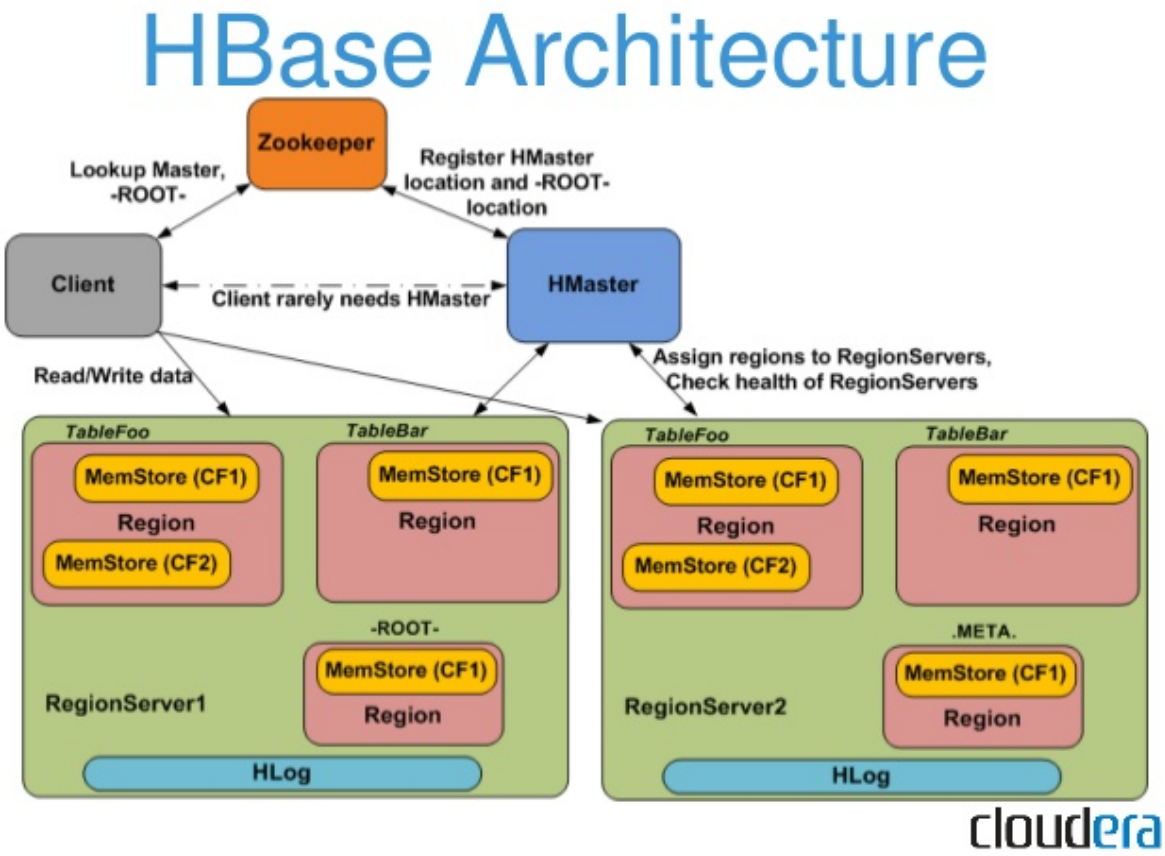


Figure 1: Architecture of HBase

The HBase Architecture mainly consists of 4 components

- HMaster
- HRegionserver
- HRegions
- ZooKeeper

3.1.1 HMaster

HBase Architecture consists of a single master node which is known as the HMaster and several slaves called the region server. A Region server can serve multiple regions but a region can only be served by a single region server. When a write request is sent by the user, the request is sent to the HMaster which in turn forwards it to the corresponding region server. HMaster is the master server in the HBase Architecture which acts as a main monitoring agent which watches over all the Region server instances. HMaster maintains the nodes present in the cluster and is the vital part due to which HBase

performs better. It takes care of all the region server and assigns and reassigns them with regions. It controls the load balancing. It provides the interface where the use can create, delete or update a table. [https://www.edureka.co/blog/hbase architecture/ (2018)]

3.1.2 HRegionserver

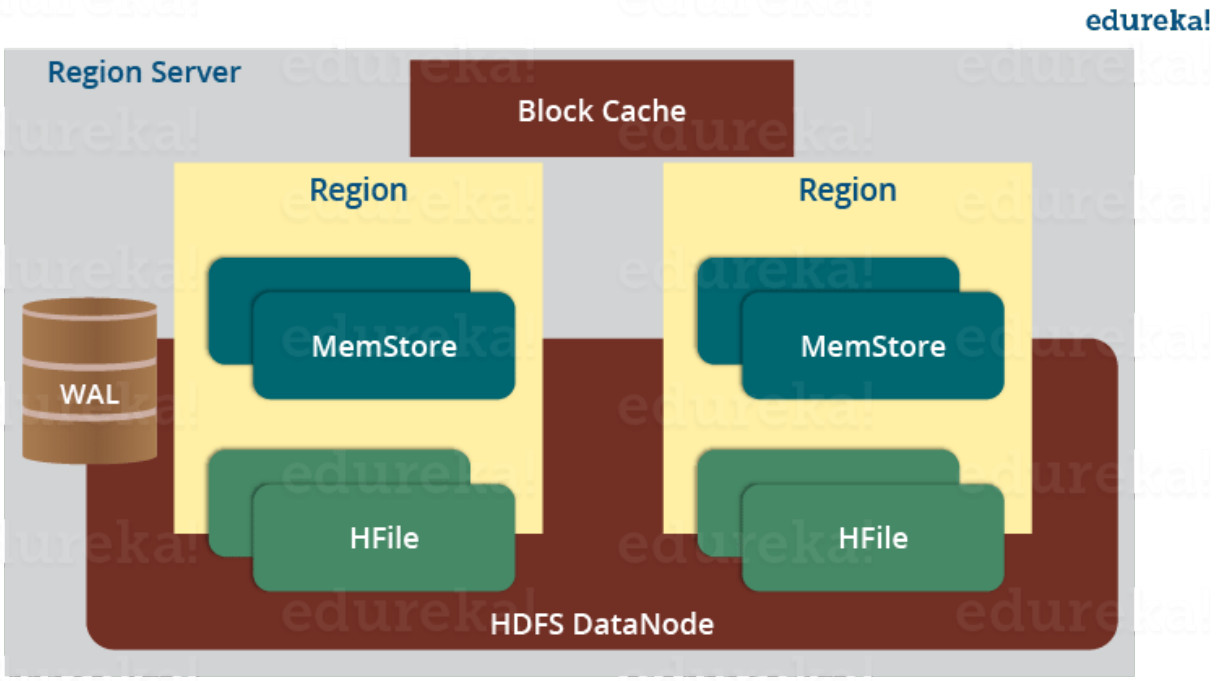


Figure: Region Server Components

These are the nodes which handle the requests that are received from the clients. The HRegionserver assigns the read write requests that it receives from the clients to the respective regions where the actual column exists. The Regionserver consists for multiple components, namely —

Block Cache —Frequently read data is stored in this block for faster access

MemStore —This is the write cache and it stores the data temporarily that are yet to be moved to the disk. Memstory is present in all the column family.

WAL —Stores the data that are not up for permanent storage.

HFile —Stores the rows that are sorted.

3.1.3 HRegions

These are the basic elements of HBase Cluster and consists of 2 components namely —Memstore and HFile

3.1.4 Zookeeper

Zookeeper is also called as the coordinator as it coordinates all the processes happening inside the Hbase distributed system. It maintains the server state inside the cluster. There is a continuous communication between the region servers and the HMaster with the Zookeeper. The region servers and the HMaster continuously sends signal to the zookeeper so that the zookeeper can check if a server is alive or not. When a HMaster fails to send a notification to the zookeeper, the session is deleted. It also takes care of the data recovery process and provides server failure notification. The zookeeper provides backup server if a particular server fails. [https://www.tutorialspoint.com/hbase/hbase$_a$*rchitecture.htm* (2018)]

Meta Table

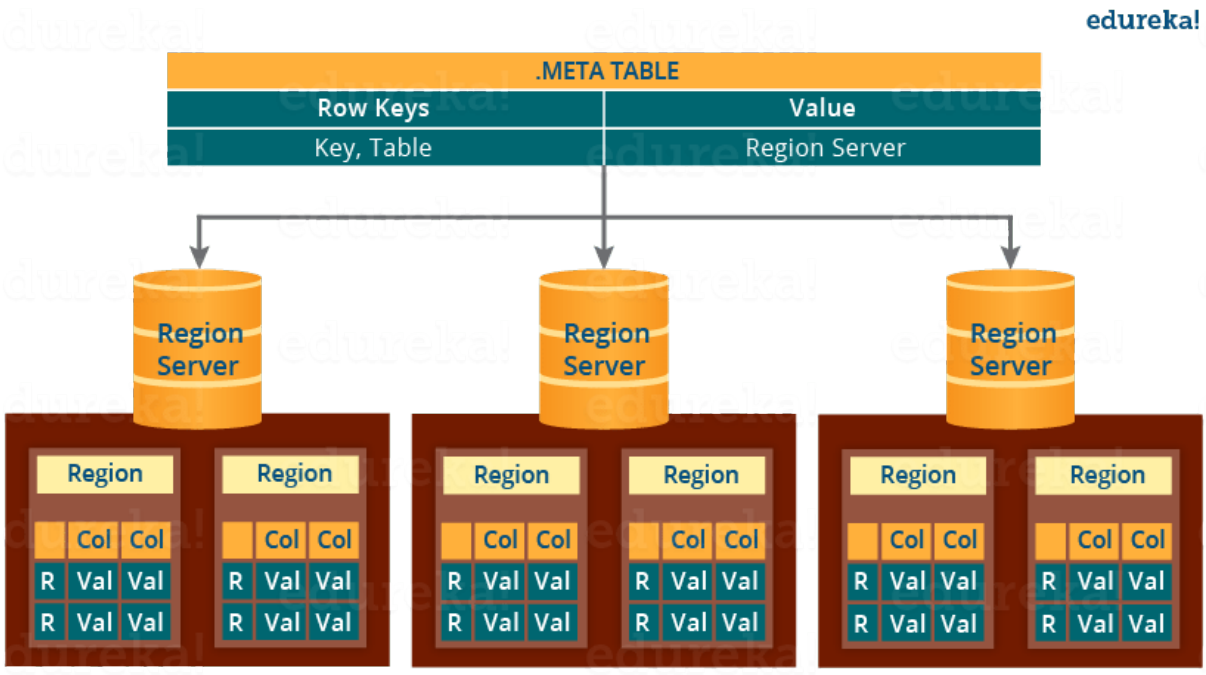


Figure: META Table

The Meta table is a catalog table for HBase which maintains the records of all the available Region servers in the system. As observed from the figure, the table is maintained in the form of keys and values.

HBase Write Mechanism

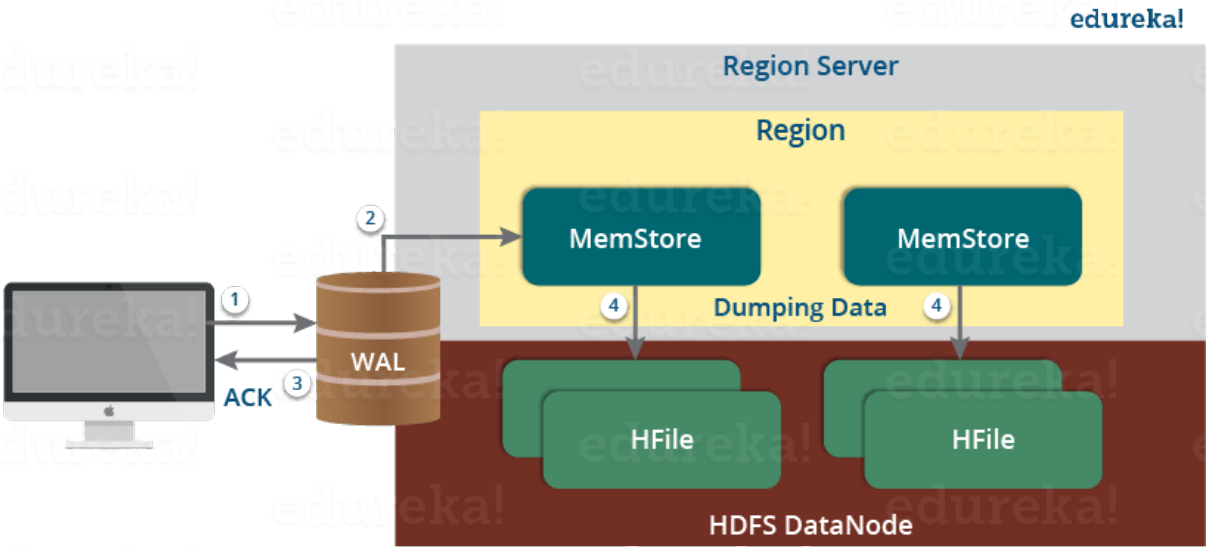


Figure: Write Mechanism in HBase

Whenever there is a write request, it is written on the WAL. The request is then copied to the Memstore and the acknowledgement is sent to the client. When MemStore attains threshold, the data is moved into a Hfile.

3.2 MongoDB

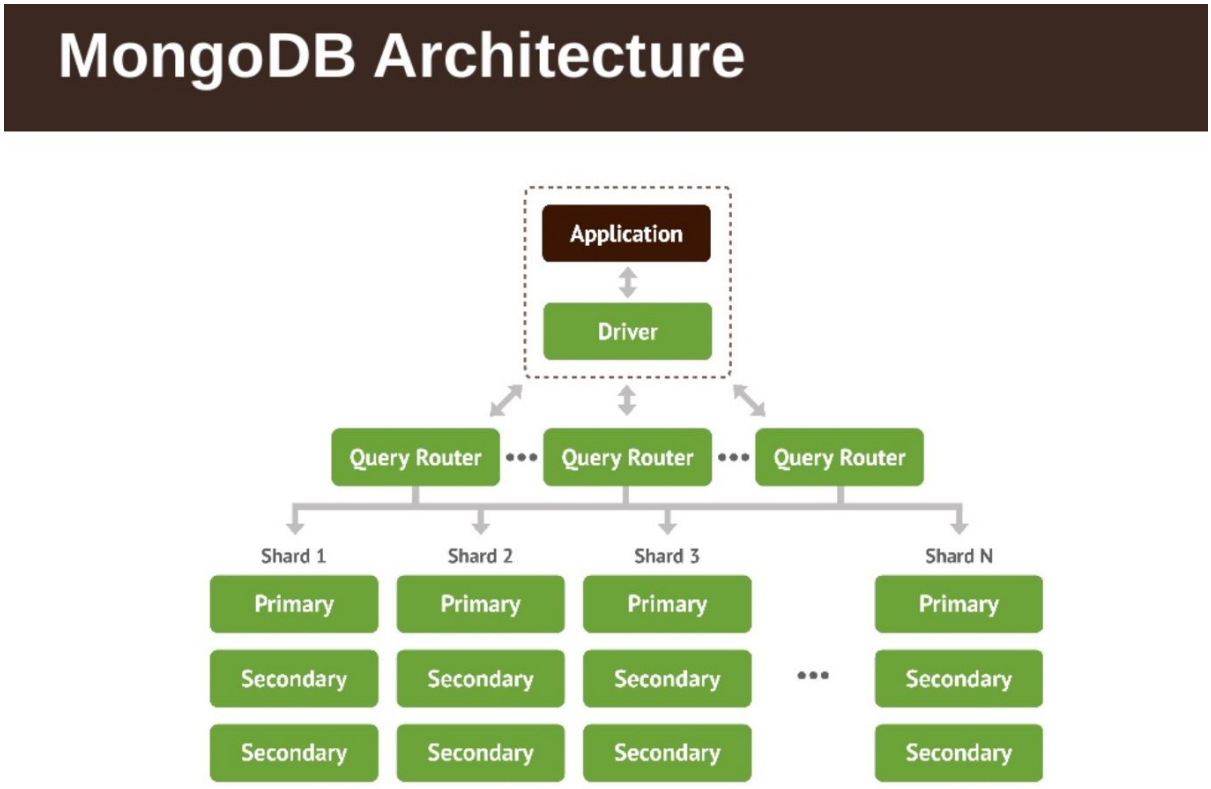


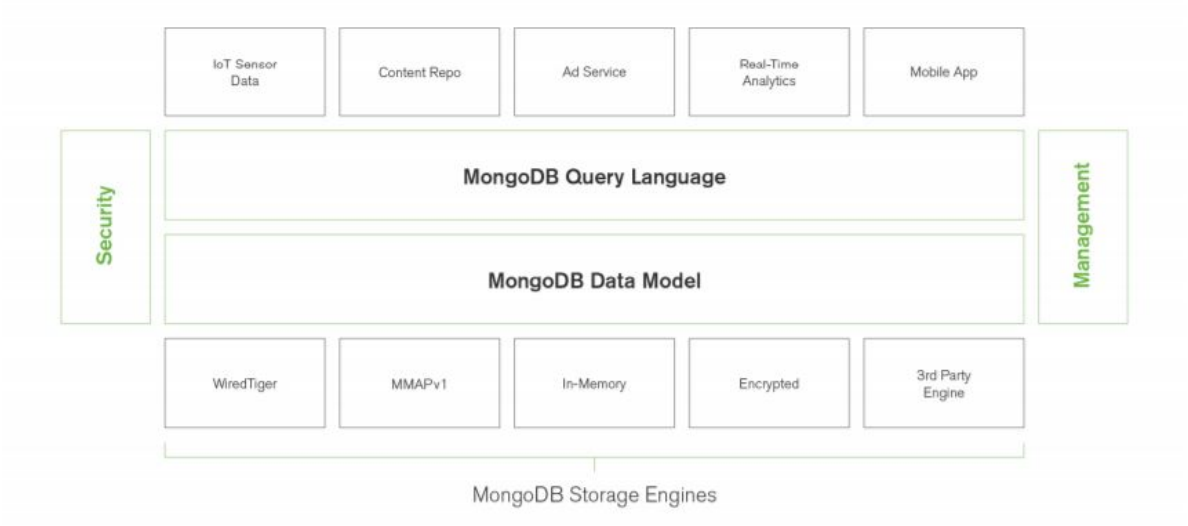
Figure 2: Architecture of MongoDB

The Nexus architecture: The main idea behind the creation of MongoDB is to combine the best of Relational databases and NoSQL.

Expressive query language: Clients are allowed to access and manipulate data as per their wish and MongoDB supports analytical application also.

Secondary indexes: MongoDB provides good indexes for read-write operation which improves its efficiency.

3.3 MongoDB Flexible Storage Architecture



The Flexible Storage Architecture automatically controls the flow of data which decreases the complexity. MongoDB ships with 4 storage engines namely —

WiredTiger Storage engine: This provides the best all round performance and storage improvement.

Encrypted Storage engine: This ensures that all the highly confidential data are well protected by encrypting them.

In-Memory storage engine: This is helpful for Realtime analytics.

MMAPv1 engine: This is the improved version of previous version of the storage engine.

4 SECURITY

4.1 HBase

HBase provides secure client access which runs on the top of HDFS. To protect itself from unauthenticated/unauthorized users and network based breach, it has network authentication protocol called the Kerberos. Kerberos are basically a secret key and the client has to prove their identity. Authentication of each service is done using a keytab file and the procedure to create a keytab file is similar to the one in Hadoop.[https://data-flair.training/blogs/hbase security/ (2018)]

4.1.1 Kerberos Authentication

The HBase client authenticates itself to the Kerberos server after which it receives a TGT. The job of the Kerberos authentication is to check if the received request is valid request or not and to check if it comes from the respective HBase Region Server.

4.1.2 Kerberos Authorisation

A limit can be set in HBase as to how many clients can access it at any given time. Also, the access level can be set for each client. The different levels of authorisation are:
Superuser: A client who has unlimited access

Read(R): Read access only.

Write(W): Write access only.

Execute(X): Execute access only.

Create(C): Create access only.

Admin(A): Right to perform cluster admin operations.

A Superuser is free to perform any of the available operations in HBase which is configured in HMaster level in hbase-site.xml. A Global user is granted a global access and has administration rights to access all the tables. A Namespace authorised user has the rights to access all table inside the namespace allocated to them. A Table access user can access a particular table for which the user has the right. Similarly, Authorisation can be combined and different users can have multiple access. For example, A global admin can access all the tables and can also perform clusterwise operations. A table-admin can create or drop a table.

4.2 MongoDB

MongoDB has 5 core security areas:

Authentication: LDAP centralizes the available data into the directory

Authorization: Defines role-based access control for each of the users.

Encryption: More important for the protection of all the data stored in MongoDB

Auditing: Ability to find which user requires which kind of access to the database.

Governance: Refers to the validation of the documents and preventing the sensitive data from entering into the main system.

LDAP Authentication is the place where users information without their roles are stored. MongoDB has built-in user roles which is turned off by default. However, it misses items like password complexity, age based rotation and identification. LDAP fills in these gaps.

5 major built in roles :
Read: Users are given read only access

ReadWrite: Allows users to edit data and read them

dbOwner: Has the right to grant access to the users and add or remove users.

dbAdminAnyDatabase: Has rights to create dbAdmin.

Root: Is the superuser which is similar to the one on HBase.

5 Literature Survey

In the paper presented by the author Ganesh Chandra (2015), the author has tested the performance of the NoSQL tools such as MongoDB, HBase and Cassandra. The Author has stated that the NoSQL data bases are slowly replacing the traditions data bases and the author has highlighted the performance of the 3 databases used for the test and have concluded that Cassandra is better than the other 2 databases. The Databases were tested using different workloads and were compared based on number of read operations, write operations, update operations, insert operations and throughput.

The author Pallas et al. (2016) has presented the paper with the information about the security aspect of HBase and how sensitive data will be analyzed and how analyzing the sensitive data will decrease the performance of the HBase system. The author says that security of HBase comes for a price, That is, either the performance of the Database or addition of more resources.

The author Waage & Wiese (2015) has made a paper explaining that the NoSQL databases are not so secured and has compared the performance of the database using YCSB using the encrypted data which is highly secured for benchmarking the database.

Paper presented by the author Yoon & Lee (2018) containts the information on recovering the deleted data in MongoDB database. MongoDB database usually creates multiple replica of the data as backup and incase if the original data gets deleted or crupted, it can recover the data from the created replicas. In the paper the internal structures of the wiredTiger and MMAPV1, which are the storage engines of the database are analyzed in depth.

The author Colombo & Ferrari (2017) talks about how privacy has become the major requirement for the DBMS and has proposed an effective solution to integrate access control into MongoDB and providing granular access to each user as per their requirement.

The author FOTACHE & COGEAN (2013) talks about the difference between MySQL and NoSQL databases and which is better for a mobile application. The author covers in depth about the performance of both MySQL and an NoSQL database by using them for a mobile application, The author also talks about the migration of database from MySQL to a NoSQL database. The author concludes that a NoSQL database is better than a MySQL database for a mobile application as it performs better and is more efficient.

6 Performance Test Plan

Yahoo! Cloud Servicing Benchmark (YCSB) is used for Benchmarking the selected databases HBase and MongoDB. YCSB is the most common open source tool which is used to carry out the benchmarking tests. Initially, after configuring, both the NoSQL databases are run using the commands start-dfs.sh, start-yarn.sh, start-hbase.sh and then the testharness script is used to process the operations in different test scenarios. The databases were subjected to different workload and at different opcounts. Their performance were then compared based on the average latency, throughputs, average time taken for read operation, average time taken for write operation and average time taken for insert operation.

Different operations done by the workloads are:

- Read
- Update
- Insert
- Scan

The performance of the databases also depends on the configuration of the machine used.

Device Specification:
Machine name : Dell Predator
Processor used: Intel core-I7
Machine RAM : 12gb

Virtual machine(Openstack)
O.S: Ubuntu 16.04 LTS (64-bit)
RAM: 4.0 Gb
HDD Space: 64 Gb

Details of the different types of available workloads
Workload A: Read/Update —50/50
Workload B: Read/Update —95/5
Workload C: Read only
Workload D: Read latest workload
Workload E: Short ranges
Workload F: Read-modify-write

Workloads used for this project:

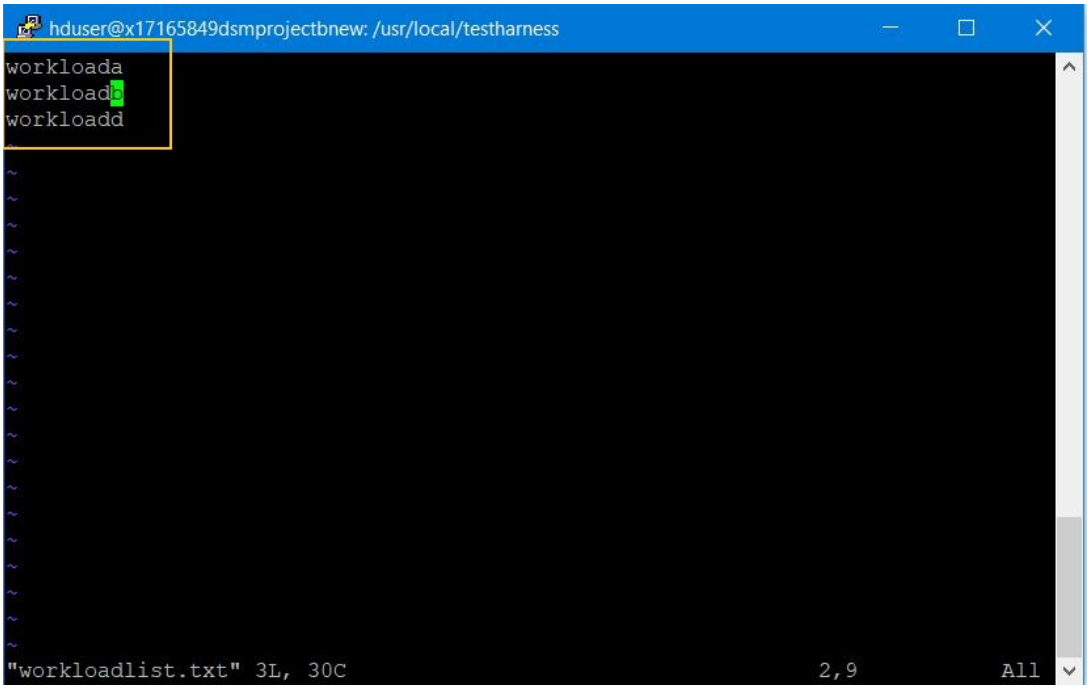


Figure 3: workloadlist.txt

Operational counts used:

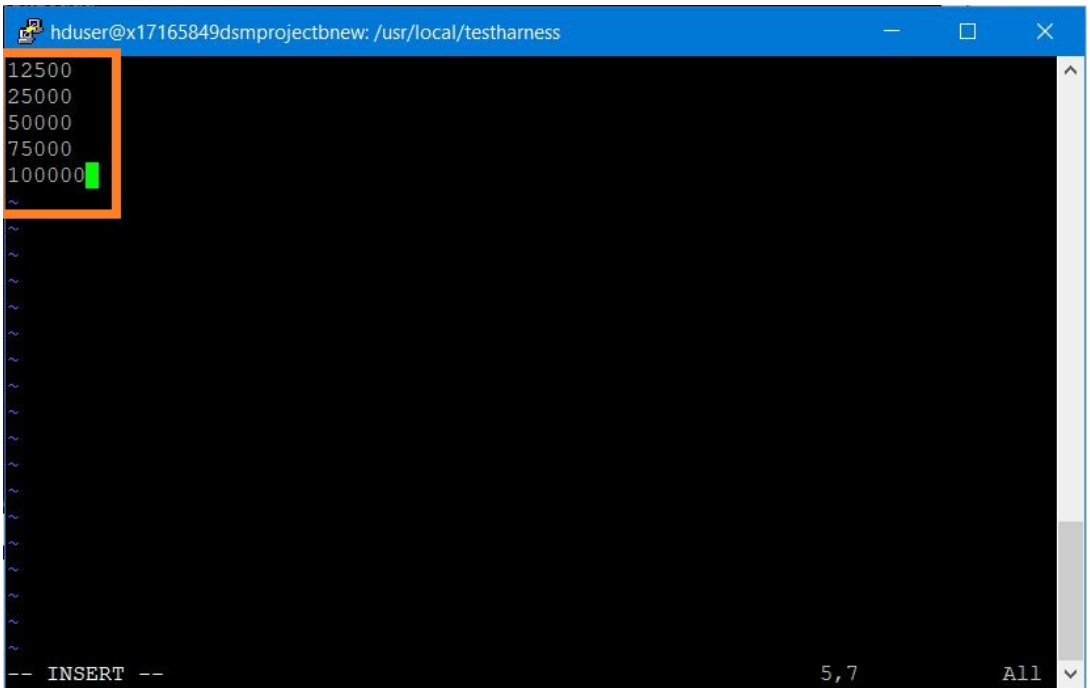


Figure 4: opcounts.txt

The opcounts used are 12500,25000,50000,75000,100000.

The opcounts and the workloads are set inside the testharness file and the test is run using the command - ./runtest.sh and the output is saved inside the YCSB file as output.

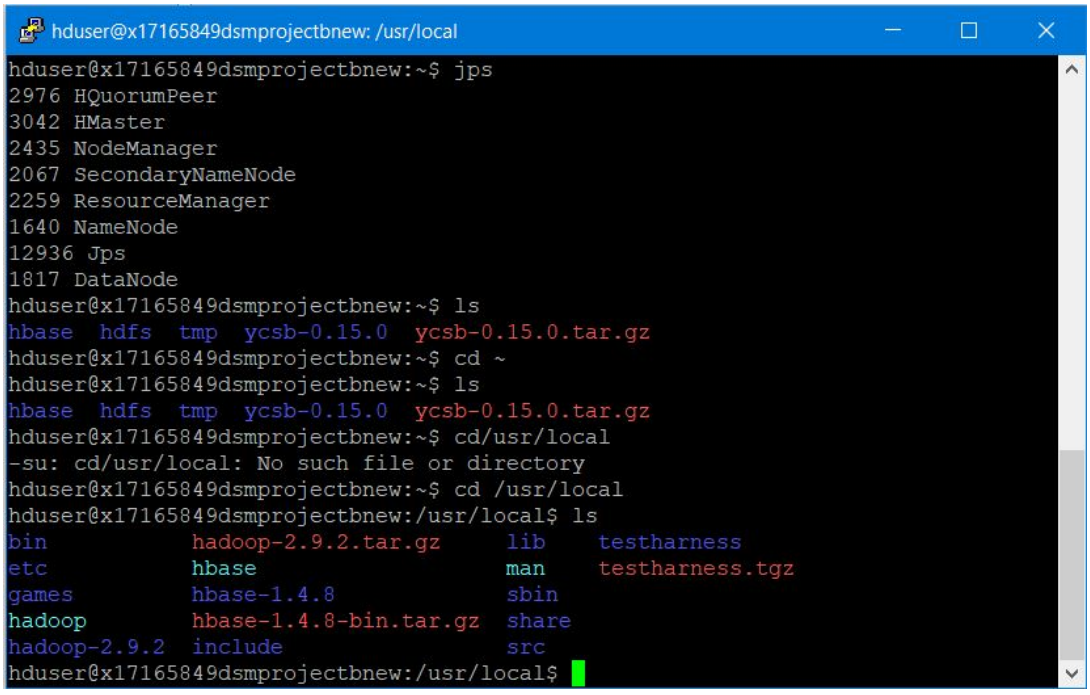


Figure 5: Testharness

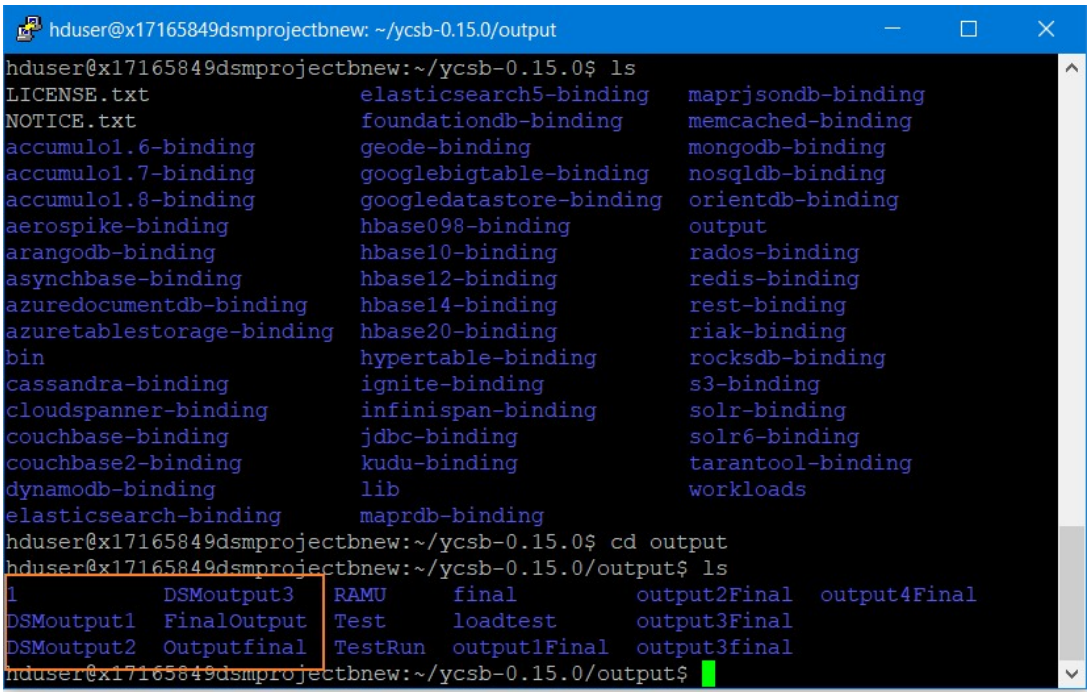


Figure 6: Output File

The outputs where then used to compare the databases.

7 Evaluation and Results

7.1 Workload A

In Workload A, the databases HBase and MongoDB are compared on the basis of average latency of of read and update operations and total throughput.

7.1.1 Average Latency for Read Operation Vs Total Throughput

On comparing the Average latency for read operation against the Total throughput, it can be inferred that MongoDB performs better as it has on an average more throughput at a lower latency than HBase

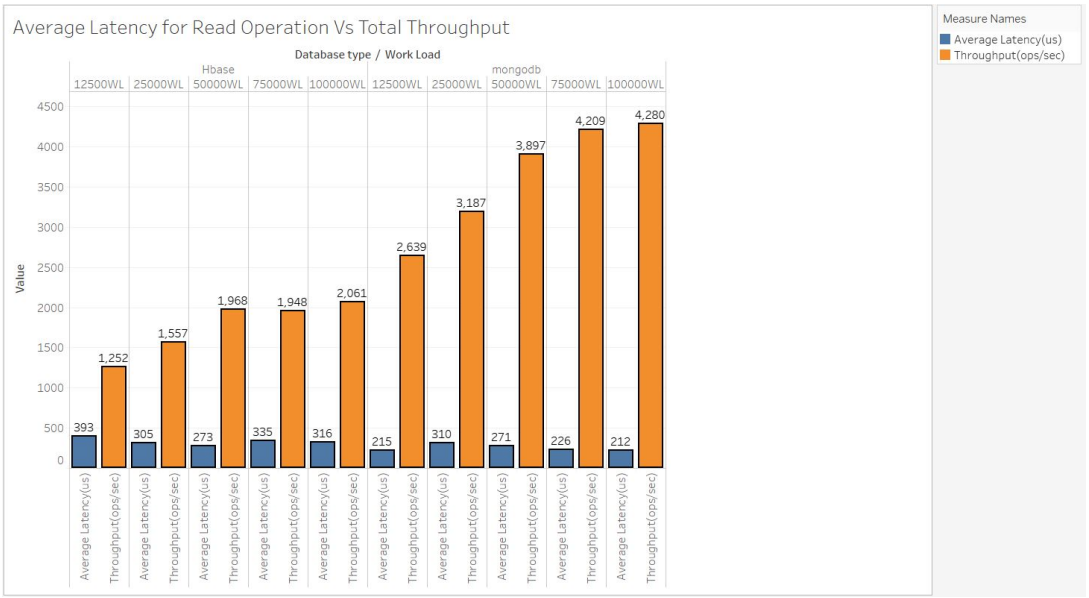


Figure 7: Average Latency for Read Operation Vs Total Throughput

7.1.2 Average Latency for Update Operation Vs Total Throughput

HBase has the highest average latency and lowest total throughput at the lowest Opcounts while MongoDB has the highest total throughput with lowest average latency at the highest opcount and therefore MongoDB is better than HBase.

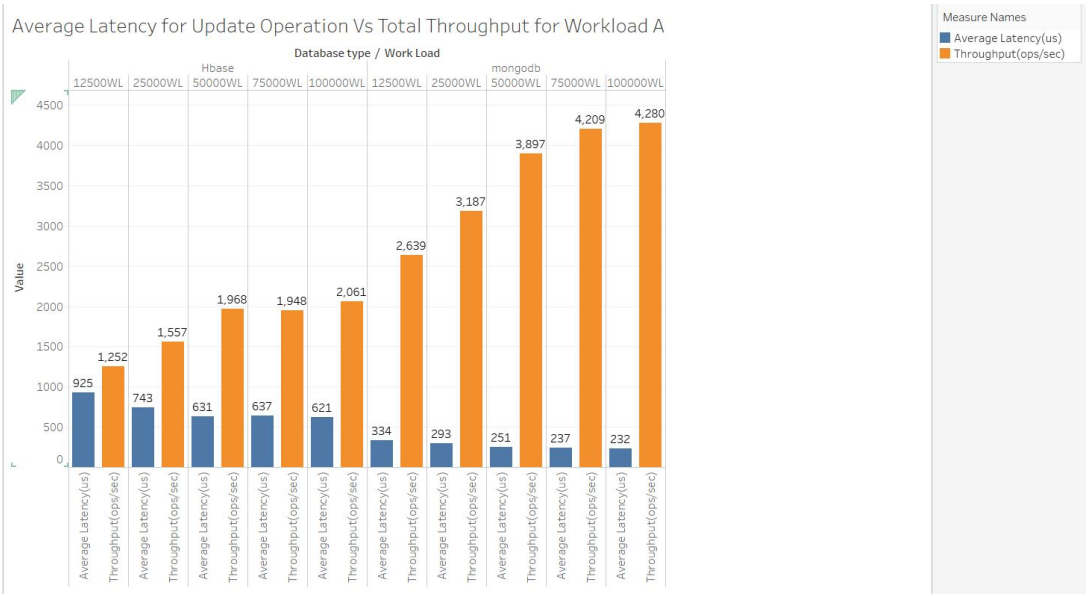


Figure 8: Average Latency for Update Operation Vs Total Throughput

7.1.3 Read Operation and Average Latency

This is a comparison between the number of Read operations and the average latency for both the databases while using workload A with Opcounts 12500,25000,50000,75000,100000.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	6255	392.866238	Hbase	READ
2	25000WL	12565.33333	304.8529525	Hbase	READ
3	50000WL	24875.33333	273.3053738	Hbase	READ
4	75000WL	37607.66667	334.8997391	Hbase	READ
5	100000WL	50126.33333	315.7491446	Hbase	READ
6	12500WL	6215.333333	215.1135411	mongodb	READ
7	25000WL	12543.66667	310.1713351	mongodb	READ
8	50000WL	24979.66667	271.1313663	mongodb	READ
9	75000WL	37665.33333	225.5821995	mongodb	READ
10	100000WL	50186.66667	212.0737895	mongodb	READ

Figure 9: Output Table

Figure 10: Read Operation and Average Latency for Workload A

From the graph it can be inferred that the number of read operations increase as the operational count increases for both the databases. The number of Read operation count is similar in both the databases over different Opcounts. The Average Latency in general is lesser in MongoDB than HBase and hence it can be inferred that MongoDB is more suited for Workload A

7.1.4 Update Operation and Average Latency

The number of Update operation and the average latency between the databases are compared while using workload A with Opcounts 12500,25000,50000,75000,100000.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	6245	924.9427436	Hbase	Update
2	25000WL	12434.66667	743.2577404	Hbase	Update
3	50000WL	25124.66667	630.8292755	Hbase	Update
4	75000WL	37392.33333	637.0591282	Hbase	Update
5	100000WL	49873.66667	620.7003145	Hbase	Update
6	12500WL	6284.666667	334.2472459	mongodb	Update
7	25000WL	12456.33333	292.9389626	mongodb	Update
8	50000WL	25020.33333	250.9822171	mongodb	Update
9	75000WL	37334.66667	237.0938511	mongodb	Update
10	100000WL	49813.33333	231.5821972	mongodb	Update

Figure 11: Output Table

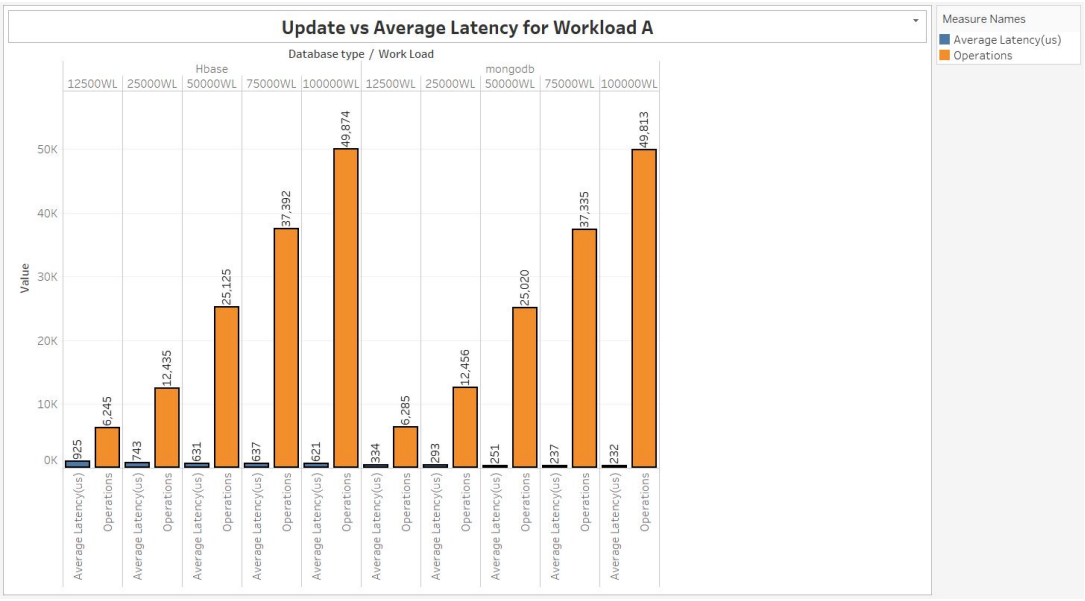


Figure 12: Update Operation and Average Latency for Workload A

Similar to Read Operation, MongoDB has very low latency while performing Update operation also and hence MongoDB would be the preferred choice over HBase for Workload A

7.1.5 Throughput

Overall Throughput is the number of total operations performed each second and it is one of the major factors in determining the better database. More the throughtput, better is the database.

ID	WorkLoad	Overall Throughput	Database type	Operation Type
1	12500WL	1252.254558	Hbase	Update
2	25000WL	1556.748398	Hbase	Update
3	50000WL	1967.764982	Hbase	Update
4	75000WL	1948.12627	Hbase	Update
5	100000WL	2061.289985	Hbase	Update
6	12500WL	2639.216504	mongodb	Update
7	25000WL	3187.123014	mongodb	Update
8	50000WL	3896.555779	mongodb	Update
9	75000WL	4209.357882	mongodb	Update
10	100000WL	4279.637899	mongodb	Update

Figure 13: Output Table

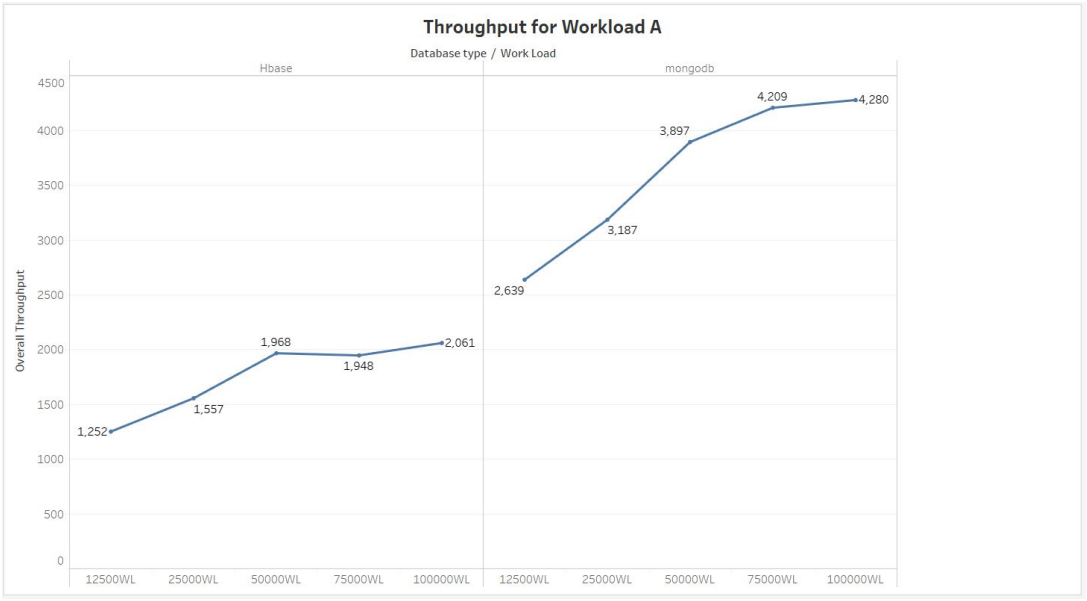


Figure 14: Throughput for Workload A

From the graph, it can be inferred that MongoDB is the clear winner when it comes to choose a database for workload A. Mongo DB has higher throughput for all the opcounts when compared to HBase.

7.2 Workload B

The databases HBase and MongoDB are compared on the basis of average latency for read and update operations and Total Throughput using Workload B

7.2.1 Average Latency for Read Operation Vs Total Throughput

The MongoDB outperforms HBase in Workload B also as it has higher throughput at a lower average latency than HBase

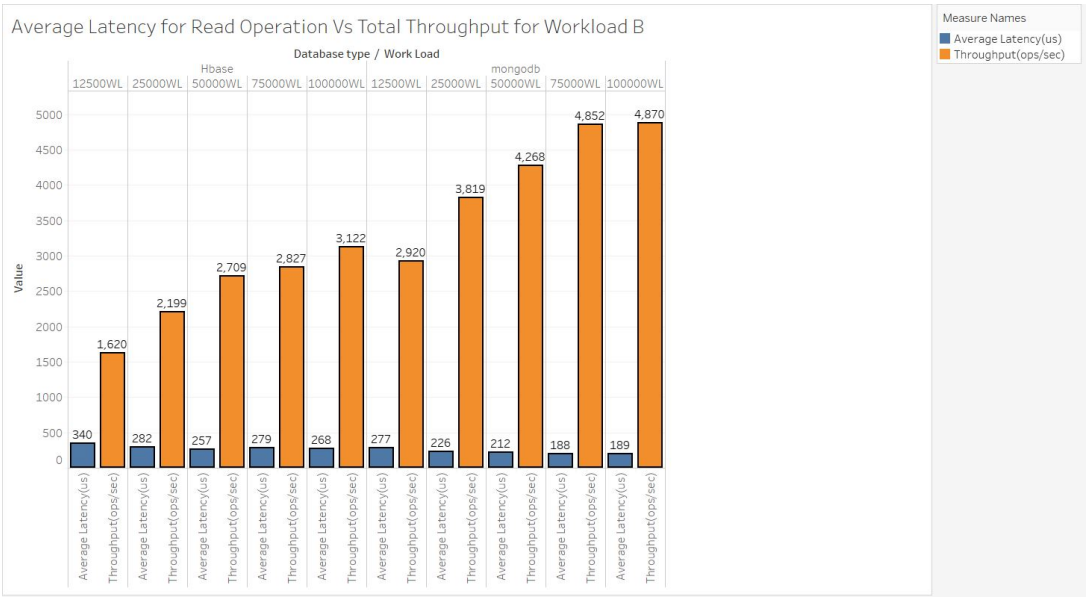


Figure 15: Average Latency for Read Operation Vs Total Throughput

7.2.2 Average Latency for Update Operation Vs Total Throughput

Similar pattern is observed when comparing Average Latency for update opertion against the total throughput also. MongoDB is the better one amoung the two.

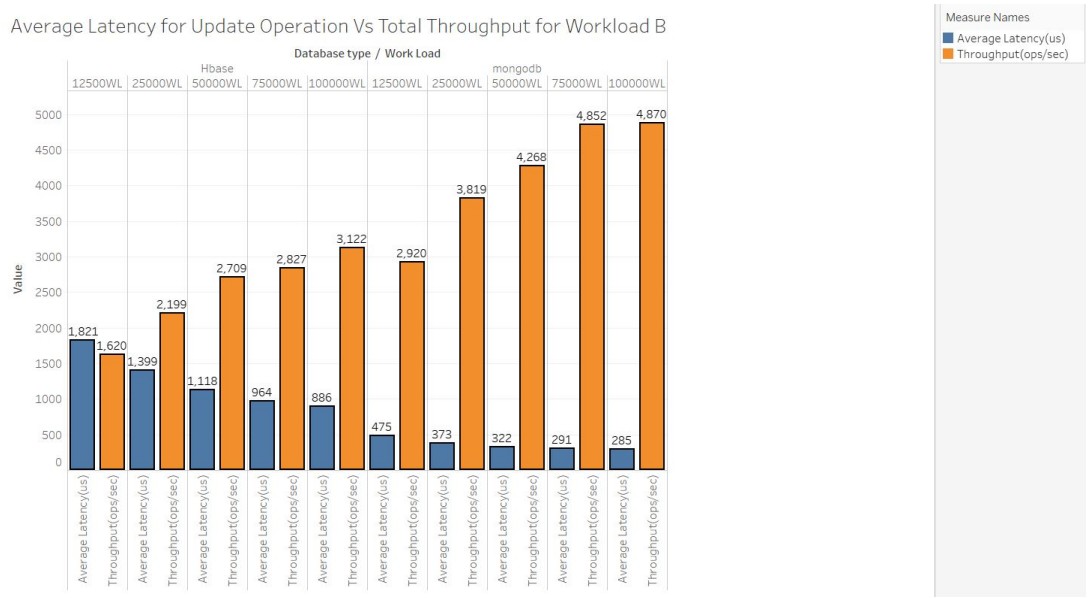


Figure 16: Average Latency for Update Operation Vs Total Throughput

7.2.3 Read Operation and Average Latency

This is a comparison between the number of Read operation and the average latency for both the databases while using workload B with Opcounts 12500,25000,50000,75000,100000.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	11885	340.0349934	Hbase	READ
2	25000WL	23738.33333	282.3103125	Hbase	READ
3	50000WL	47501	256.9796753	Hbase	READ
4	75000WL	71292	278.5497473	Hbase	READ
5	100000WL	95017	268.3867509	Hbase	READ
6	12500WL	11898	277.3931959	mongodb	READ
7	25000WL	23761.33333	225.8029687	mongodb	READ
8	50000WL	47488.33333	212.033129	mongodb	READ
9	75000WL	71254.66667	188.4058226	mongodb	READ
10	100000WL	94954.33333	189.3313946	mongodb	READ

Figure 17: Output Table

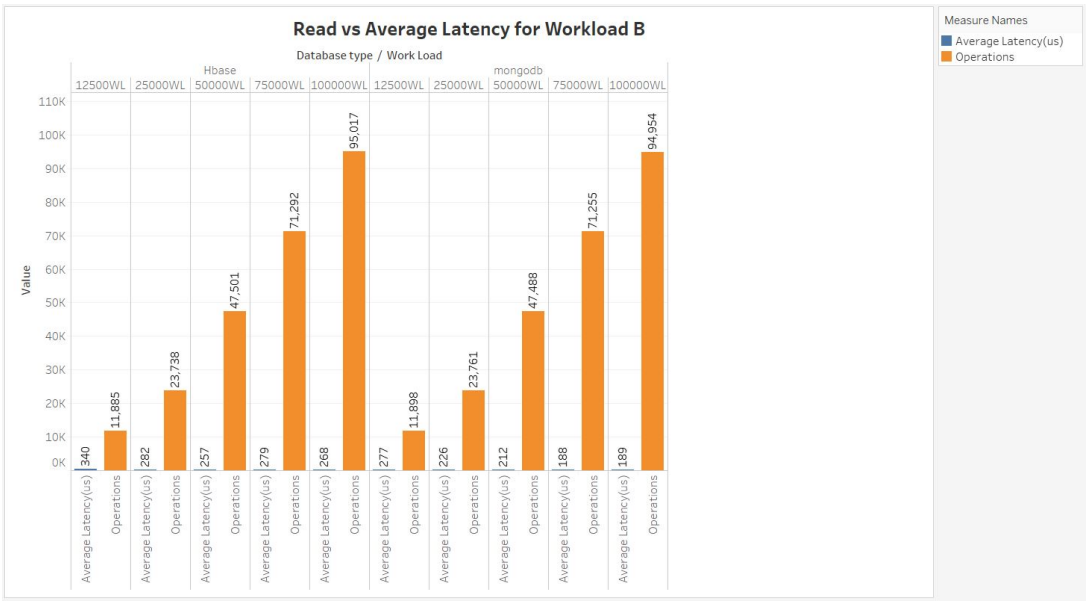


Figure 18: Read Operation and Average Latency for Workload B

Once again MongoDB does better performance than HBase. The Graph shows that the latency is low for MongoDB than HBase over different opcounts.

7.2.4 Update Operation and Average Latency

The Update operation and the average latency are compared accross the different opcounts for both the data bases while testing with workload B.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	615	1821.114655	Hbase	UPDATE
2	25000WL	1261.666667	1398.895999	Hbase	UPDATE
3	50000WL	2499	1117.749676	Hbase	UPDATE
4	75000WL	3708	964.4242398	Hbase	UPDATE
5	100000WL	4983	885.7473943	Hbase	UPDATE
6	12500WL	602	475.3603172	mongodb	UPDATE
7	25000WL	1238.666667	372.8627393	mongodb	UPDATE
8	50000WL	2511.666667	321.5863451	mongodb	UPDATE
9	75000WL	3745.333333	291.0227511	mongodb	UPDATE
10	100000WL	5045.666667	285.4782741	mongodb	UPDATE

Figure 19: Output Table

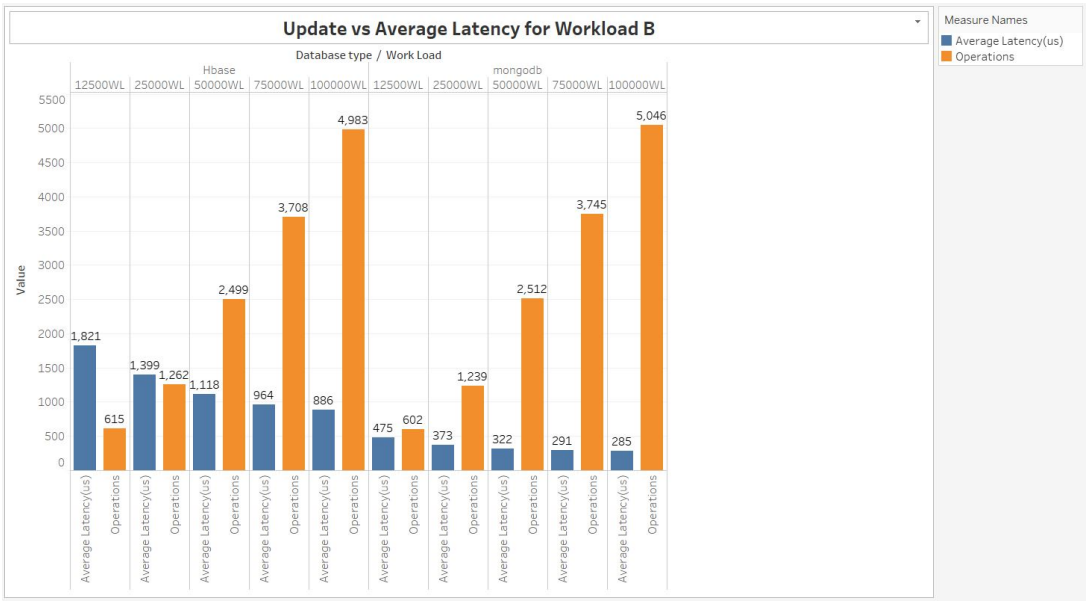


Figure 20: Update Operation and Average Latency for Workload B

HBase produces lesser update counts at higher average latency than MongoDB, which produces more update counts at lower average latency and hence MongoDB is preferred over HBase.

7.2.5 Throughput

Over all throughput is the most important factor in determining which database to use at different conditions. More the throughput, better the database is.

ID	WorkLoad	Overall Throughput	Database type	Operation Type
1	12500WL	1619.937988	hbase10	OVERALL
2	25000WL	2198.604268	hbase10	OVERALL
3	50000WL	2709.271412	hbase10	OVERALL
4	75000WL	2827.239889	hbase10	OVERALL
5	100000WL	3121.874415	hbase10	OVERALL
6	12500WL	2920.114508	mongodb	OVERALL
7	25000WL	3819.260249	mongodb	OVERALL
8	50000WL	4267.646067	mongodb	OVERALL
9	75000WL	4852.31114	mongodb	OVERALL
10	100000WL	4870.341135	mongodb	OVERALL

Figure 21: Output Table

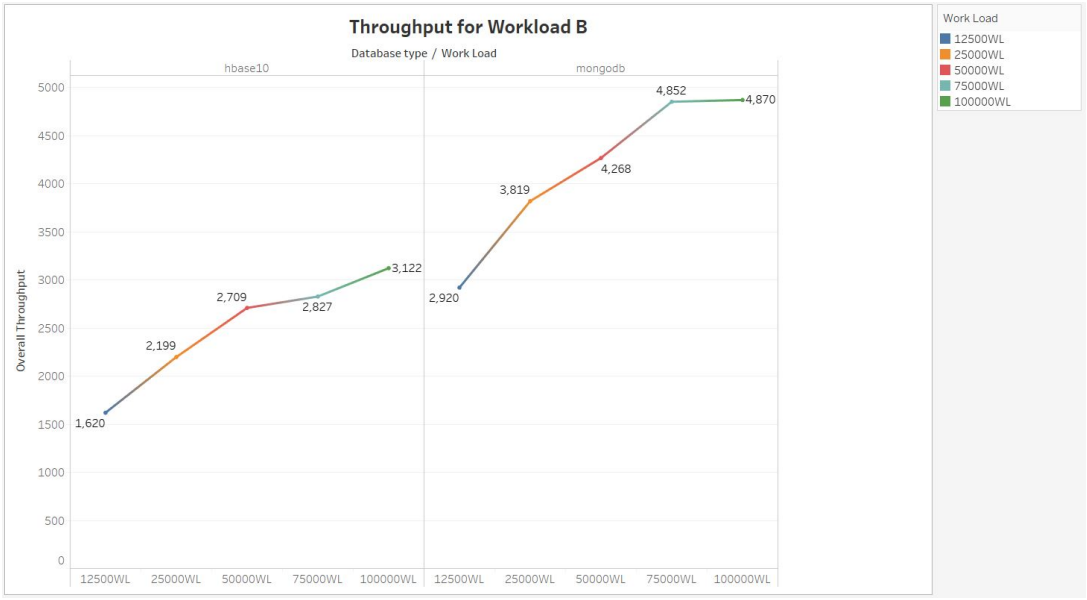


Figure 22: Throughput for Workload B

7.3 Workload C

The databases HBase and MongoDB are compared on the basis of average latency of read and Insert operations and Total Throughput using Workload C

7.3.1 Average Latency for Insert Operation Vs Total Throughput

Average Latency for Insert operation is compared with the Total throughput for workload C for both the databases and MongoDB outperforms HBase and hence it is preferred over HBase.

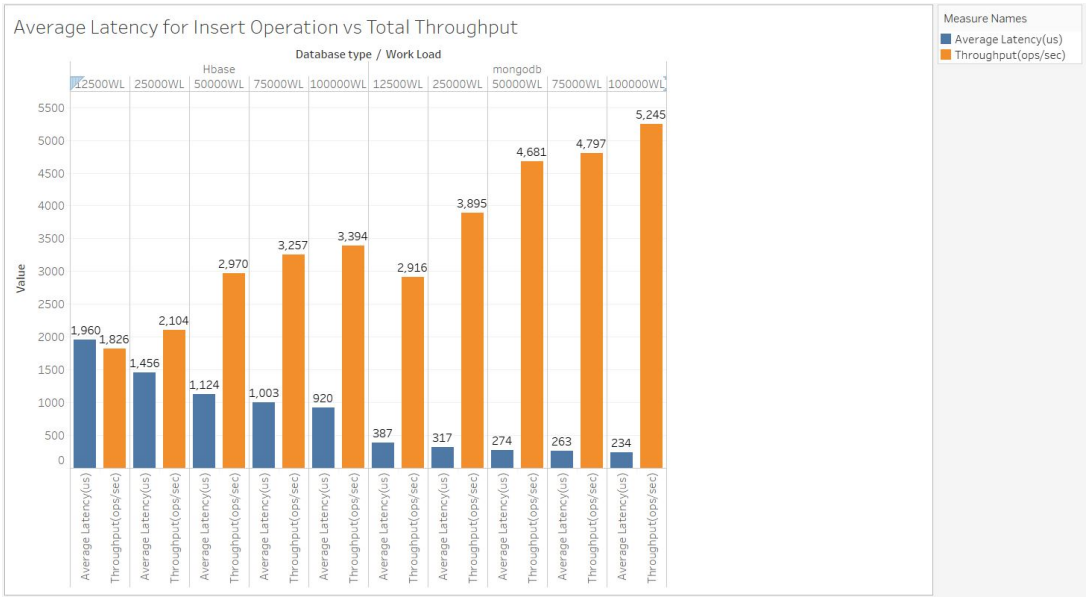


Figure 23: Average Latency for Insert Operation Vs Total Throughput

7.3.2 Average Latency for Read Operation Vs Total Throughput

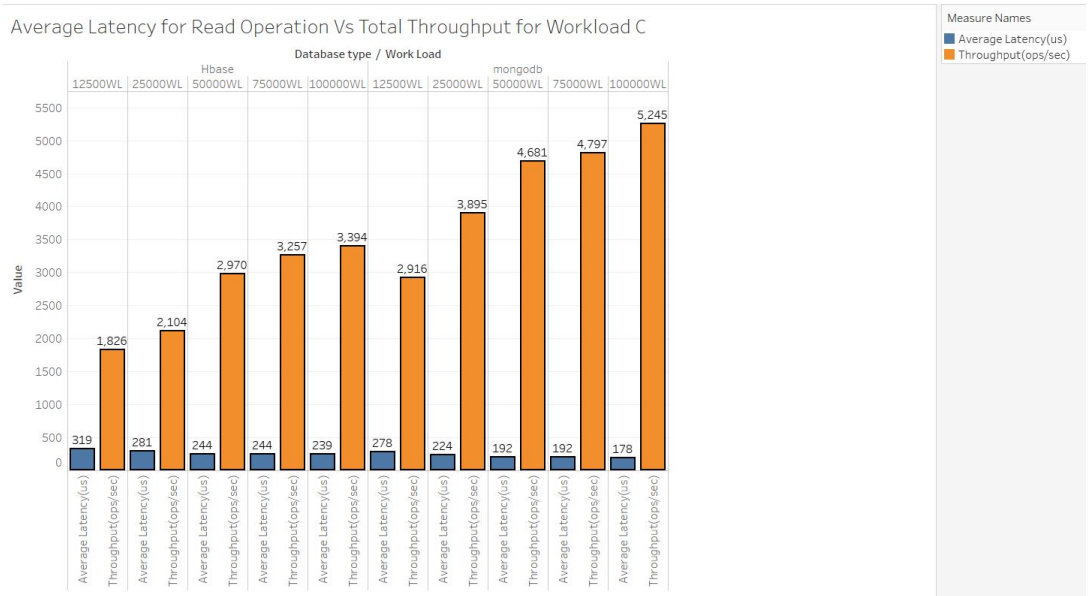


Figure 24: Average Latency for Read Operation Vs Total Throughput

7.3.3 Read Operation and Average Latency

This is a comparison between the number of Read operation an the average latency for both the databases while using workload B with Opcounts 12500,25000,50000,75000,100000.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	11865.33333	318.5807201	Hbase	READ
2	25000WL	23724	280.7493234	Hbase	READ
3	50000WL	47430	244.0351486	Hbase	READ
4	75000WL	71213.33333	243.922759	Hbase	READ
5	100000WL	95051.33333	238.8169183	Hbase	READ
6	12500WL	11869	278.0646512	mongodb	READ
7	25000WL	23725.66667	224.0081319	mongodb	READ
8	50000WL	47447.66667	192.3703216	mongodb	READ
9	75000WL	71238.66667	191.927512	mongodb	READ
10	100000WL	94904	177.9945002	mongodb	READ

Figure 25: Output Table

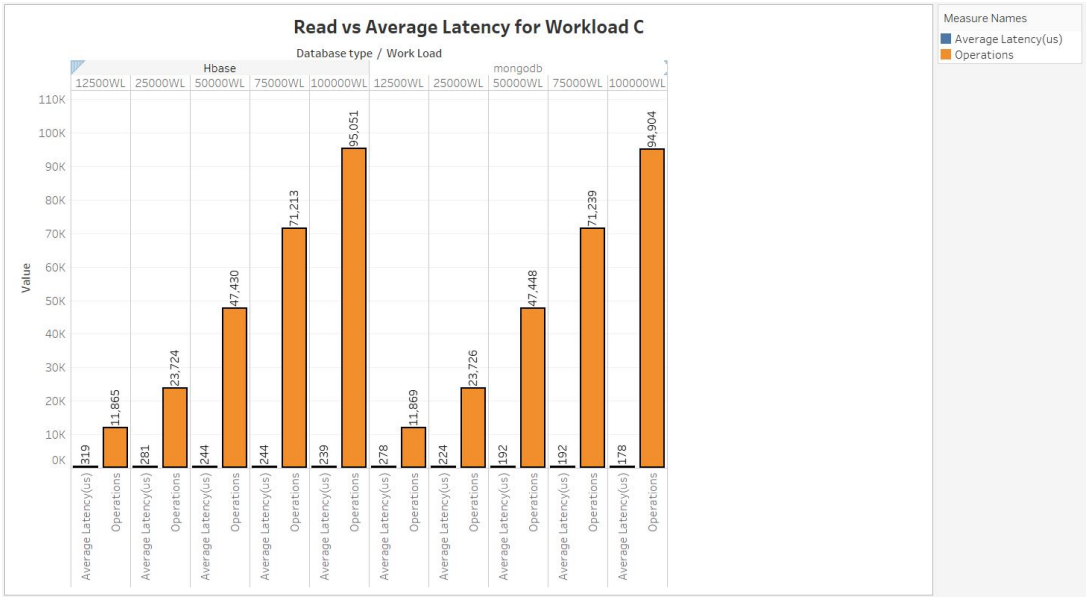


Figure 26: Read Operation and Average Latency for Workload B

As observed from the Figure 20, MongoDB does more read operations at low latency when compared to HBase and hence it is preferred over HBase.

7.3.4 Insert Operation and Average Latency

In this section, Number of Insert counts are compared with the Average latency for Workload C for the databases HBase and MongoDB over multiple opcounts.

ID	WorkLoad	Operations	Average Latency(us)	Database type	Operation Type
1	12500WL	634.6666667	1959.75248	Hbase	INSERT
2	25000WL	1276	1455.855479	Hbase	INSERT
3	50000WL	2570	1124.310346	Hbase	INSERT
4	75000WL	3786.666667	1003.348067	Hbase	INSERT
5	100000WL	4948.666667	920.068847	Hbase	INSERT
6	12500WL	631	386.8694908	mongodb	INSERT
7	25000WL	1274.333333	316.9145548	mongodb	INSERT
8	50000WL	2552.333333	274.4066392	mongodb	INSERT
9	75000WL	3761.333333	262.8197871	mongodb	INSERT
10	100000WL	5096	234.4461939	mongodb	INSERT

Figure 27: Output Table

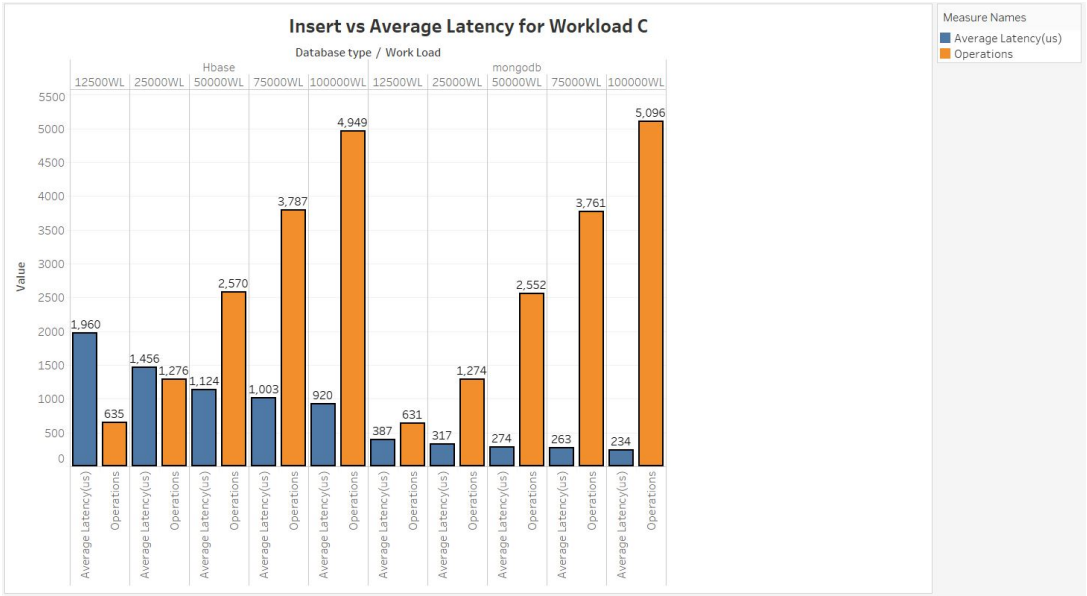


Figure 28: Read Operation and Average Latency for Workload B

The MongoDB performs better again and hence is preferred over HBase.

7.3.5 Throughput

ID	WorkLoad	Overall Throughput	Database type	Operation Type
1	12500WL	1825.880233	Hbase	OVERALL
2	25000WL	2103.783951	Hbase	OVERALL
3	50000WL	2969.8659	Hbase	OVERALL
4	75000WL	3256.649804	Hbase	OVERALL
5	100000WL	3393.829541	Hbase	OVERALL
6	12500WL	2916.11669	mongodb	OVERALL
7	25000WL	3894.708959	mongodb	OVERALL
8	50000WL	4680.557345	mongodb	OVERALL
9	75000WL	4797.244495	mongodb	OVERALL
10	100000WL	5244.910202	mongodb	OVERALL

Figure 29: Output Table

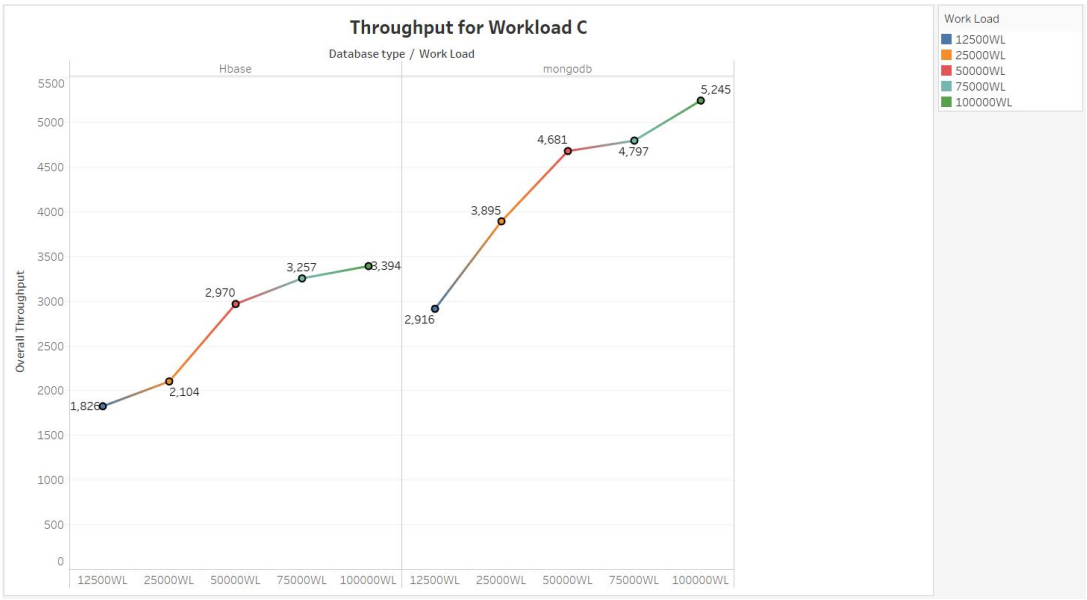


Figure 30: Throughput for Workload B

MongoDB has better throughput rate than HBase and hence it is the better choice than HBase for Workload C.

7.4 Throughput vs Average Latency

The 2 databases are compared on the basis of throughput vs Average Latency and it can be inferred from the graph that MongoDB performs better at all the three workloads used for the test. MongoDB has more overall throughput than Hbase and that too at a lower latency than HBase, which therefore confirms that MongoDB should be the preferred database over HBase.

Workload A

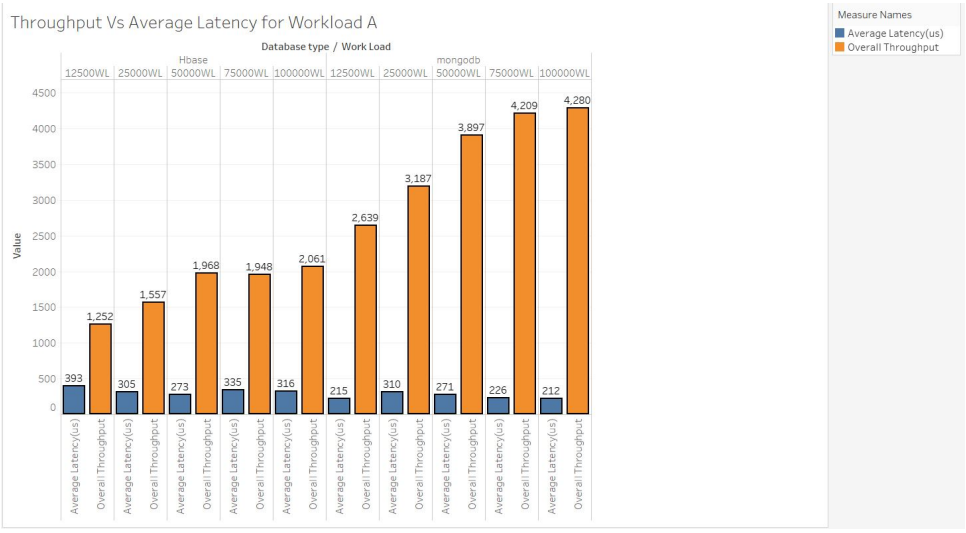


Figure 31: Throughput vs Average Latency for Workload A

Workload B

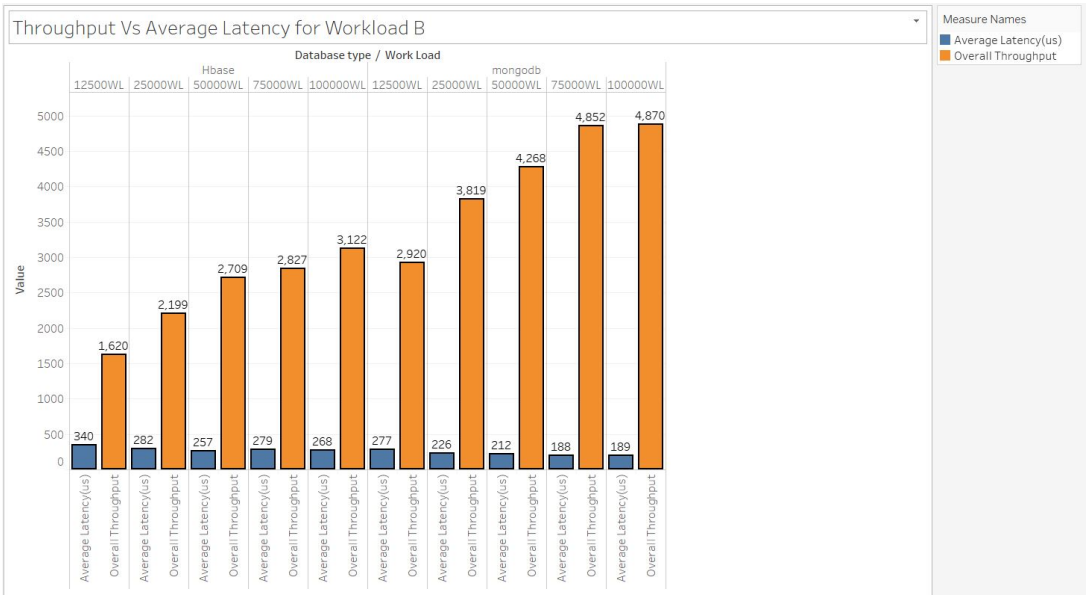


Figure 32: Throughput vs Average Latency for Workload B

Workload c

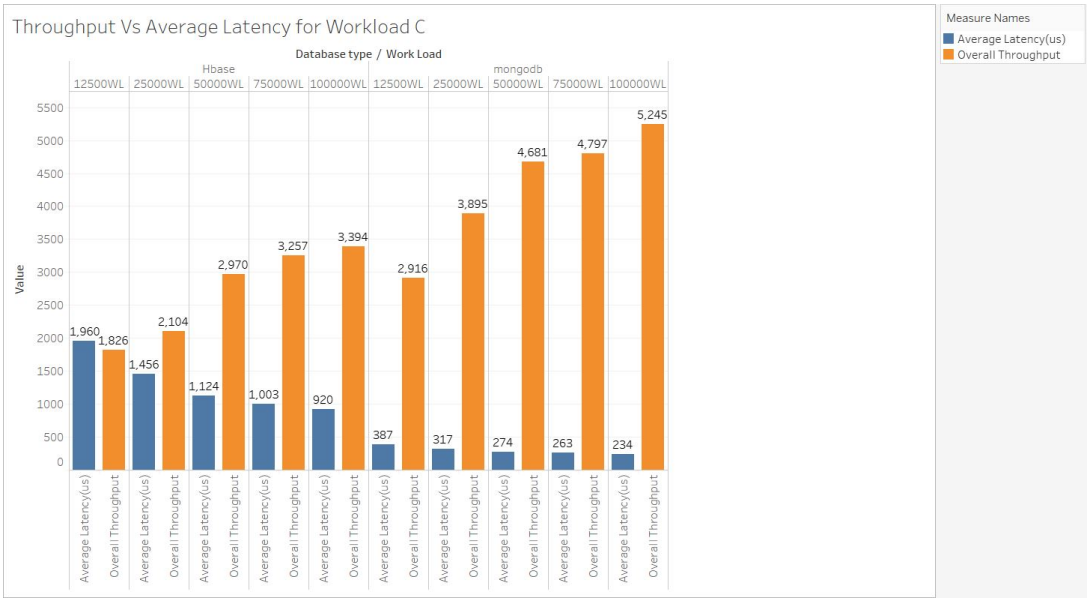


Figure 33: Throughput vs Average Latency for Workload C

Result: After running the test and analyzing the outputs for the 2 databases for different workloads at different opcounts, it can be concluded that MongoDB is the better one among the two for the tests undergone. MongoDB outperforms HBase in every aspect and for all the different workloads.

8 Conclusion and discussion

In real life, a database is picked based on the requirement and the type of data that is used. Before choosing a database to handle all the data, a company runs various tests and checks various parameters such as —performance, efficiency, speed, ability to handle unstructured data and security. The performance of a database also depends on the hardware used to run the particular database. Different NoSQL databases have different advantanges and disadvantanges and hence benchmarking tests are run to determine which database is best suited for a particular company or a project. The motive of the project was to install and compare the 2 NoSQL databases —HBase and MongoDB and to test the performance of both the databases by subjecting them to multiple workloads. The outputs of the database are then compared based on their Read-Update operation, Average Latency and Throughput. For all the workloads, the databases were compared using the average latency for read operation against the total throughout, average latency for update operation against the total throughput, average latency for insert operation against the total throughput, and finally comparing the total throughput against the average overall latency. MongoDB turned out to be the better performer for all the tests which were performed. MongoDB was well ahead of HBase in terms of performance and is the clear choice of database that will be picked when the test is performed for all the 3 workloads used in this project.

References

Colombo, P. & Ferrari, E. (2017), 'Enhancing mongodb with purpose-based access control', *IEEE Transactions on Dependable and Secure Computing* **14**(6), 591–604.

FOTACHE, M. & COGEAN, D. (2013), 'Nosql and sql databases for mobile applications. case study: Mongodb versus postgresql.', *Informatica Economica* **17**(2), 41 – 58.
URL: *http://search.ebscohost.com/login.aspx?direct=trueAuthType=ip,cookie,shibdb=bthAN=88803 livescope=sitecustid=ncirlib*

Ganesh Chandra, D. (2015), 'Base analysis of nosql database.', *Future Generation Computer Systems* **52**(Special Section: Cloud Computing: Security, Privacy and Practice), 13 – 21.
URL: *http://search.ebscohost.com/login.aspx?direct=trueAuthType=ip,cookie,shibdb=edselpAN=S01 livescope=sitecustid=ncirlib*

https://data-flair.training/blogs/hbase security/ (2018), 'Security'.
URL: *https://data-flair.training/blogs/hbase-security/*

https://www.edureka.co/blog/hbase architecture/ (2018), 'Hbase architecture: Hbase data model hbase read/write mechanism'.
URL: *https://www.edureka.co/blog/hbase-architecture/*

https://www.tutorialspoint.com/hbase/hbase$_a rchitecture.htm$(2018), '$Hbase'$.
URL:*https://www.tutorialspoint.com/hbase/hbase$_a$rchitecture.htmPallas, F., Gnther, J.& Bermbac*

Waage, T. & Wiese, L. (2015), Benchmarking encrypted data storage in hbase and cassandra with ycsb, *in* F. Cuppens, J. Garcia-Alfaro, N. Zincir Heywood & P. W. L. Fong, eds, 'Foundations and Practice of Security', Springer International Publishing, Cham, pp. 311–325.

Yoon, J. & Lee, S. (2018), 'A method and tool to recover data deleted from a mongodb', *Digital Investigation* **24**, 106 – 120.
URL: *http://www.sciencedirect.com/science/article/pii/S1742287617302347*