# ABSTRACT

The rate at which data is being generated has given rise to multiple NoSQL databases which are being actively used to handle Big-Data. These new databases has many advantages over the age old Relational Database Management Systems because of their improved performance they have shown running against numerous heavy different workloads. This paper, illustrates the comparison shown by the performances carried out by two popular NoSQL databases (MongoDB and HBase). This paper will be evaluating the performance of HBase and MongoDB using Yahoo Cloud Service Benchmark (YCSB) and record their respective results. On the basis of workloads, we will measure different parameters of both HBase and MongoDB where HBase stores data in key value pairs in columnar fashion whereas MongoDB is a doc-oriented NoSQL database.


# INTRODUCTION

Today in the era of Digital World where everyone is facilitated with internet , there is humongous amount of data that is being generated every second in the form of pictures , videos , audios , etc . This huge generation of volume of data that is being generated every second has led to performance degradation .It's recorded that about 2.5 quintillion byte of data is being produced every day. The fact Cloud computing and Mobile computing, are the new emerging technologies has led a huge transformation of generation in data volume from structured to unstructured .The proper storage and management of this Big-Data at such volume and velocity has become an issue for the traditional databases when the data being produced is both structured and unstructured as the traditional RDBMS ( Relational Database Management System ) do support only structured data files . This shortcoming of the traditional RDBMS was overcome with the arrival of new database model ,NoSQL .

NoSQL DBMS emerged as a solution to the data processing requirements. NoSQL DBMS provides an efficient means for storing and accessing Big-Data as its servers are more easily horizontally scalable and replicable as compared to relational DBMS. It has no fixed structured schema because of which users are able to dynamically change the applications for data model compared to the traditional relational database models. It comes up with better performance . Okman said Horizontal scalability is one of the prime features of NoSQL databases . According to Flair , NoSQL database is categorized under four key-value database, column-oriented database, graph database and document database . The NoSQL database is categorized based on their capability to perform task under different workloads. This paper will be focusing on the comparison of performance performed by HBase and MongoDB by undergoing the benchmarking tool YCSB which will be used to perform operation counts and scalability tests to evaluate the performance of the database for different record counts like insert , update and read operations.

# Key Characteristics

## MongoDB

MongoDB is an opensource DBMS which was developed by 10 gen in US. It is one among the databases that is used for Big Data analytics in companies like Ebay , MTV, Foursquare, Expedia, Forbes, McAfee. It is the fastest open source database with more than 5000 customers and about 30 million downloads with each document stored in a JSON format (Java Script Object Notation). In other words, it is document oriented which stores documents in key value pairs. It is built with a schema less structure with no need for defining the structure of the data. There are various characteristics of MongoDB from which some of them are described below [1], [16] :

I. Aggregation framework : For the purpose of batch processing and data aggregation , MongoDB provides the feature of aggregation .It uses Map Reduce for parallel processing on big datasets in clusters. Map Reduce works in two combinations , Map which is used for filter and sort the data and Reduce which is used for operational summarizing.
II. File Storage : It uses the feature of replication which replicates and stores the data in multiple clusters from which data can be retrieved .It also comes up with load balancing feature over multiple clusters for file storage. Data is filled in shards for auto load balancing .
III. Schema-less : It is flexible over traditional databases models as it is schema less database which is written in C++ with an advantage of reduced friction and lack of setup.
IV. Replication : MongoDB being a distributed NoSQL database provides replication of data over multiple clusters (nodes) in a master-slave fashion which produce replica data sets in different machines. Here the Master performs the read and write operation, copy files where as the Slave performs only the read operation. Automatic failover is also a feature which is responsible for high availability .
V. Management Services : It can be used to locate the data on the web and then check for the replicated data backup .
VI. Supports ad-hoc queries and multiple storage engine.
VII. Sharding : In the old databases, scaling the data was a measure challenge which was overcome by the feature named Sharding. It is this feature that enables the distribution of data across different nodes in MongoDB which makes is very important feature. It allows horizontal scaling which is achieved

by allocating different machine on shards. The above picture explains Sharding process in detail.

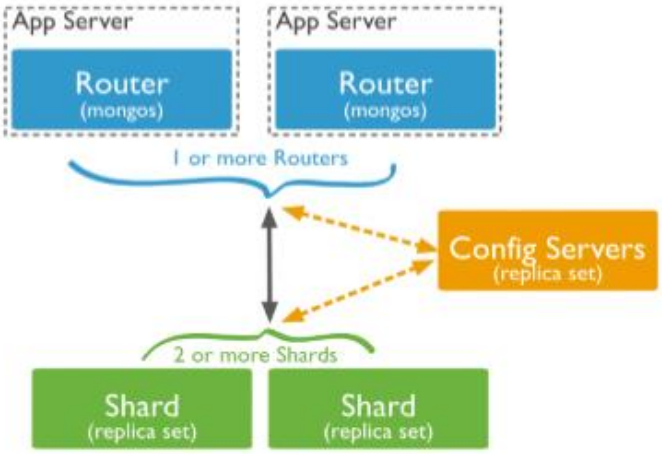



Fig 1 : Sharding process [1]

## HBase

Apache Hbase is a distributed database which is also open sourced and a non- relational database and has been written in Java. It is column oriented and also stores data as a key value pairs . It is built over the HDFS (Hadoop Distributed File System) as it's a part of the Hadoop system. It makes use of the Hadoop distributed file system for the feature of scalability . It has the potential to manage data sets consisting with billions of rows and millions of columns . Hbase has low input/output latency and high throughout [2]. Hbase is fast at providing look ups for huge datasets. Some of the key characteristics of HBase are mention above :

I. Hbase provides the data that is replicated from different nodes.
II. It facilitates failure support which is automatic
III. The programs used in HBase optimise Java API's for client access which makes it convenient for the user.
IV. HBase consists of with the feature of linear scalability and if there exists some failure than there is support option which is automatic.
V. Consistency in the read and write operations, integration with the HDFS and ease for access for the clients via JAVA API are some more features of HBase [3].
VI. It comes with flexible , multi-dimentional , column based map structure
VII. It has both linear as well as modular scalability.
VIII. Real-time querying is done using block cache and bloom filters [2]
IX. It consists of convenient base classes which is used to backup Hadoop Mapreduce jobs along with Apache HBase tables which can be integrated with Pig and Hive [2]

# Architecture

## Architecture of MongoDB

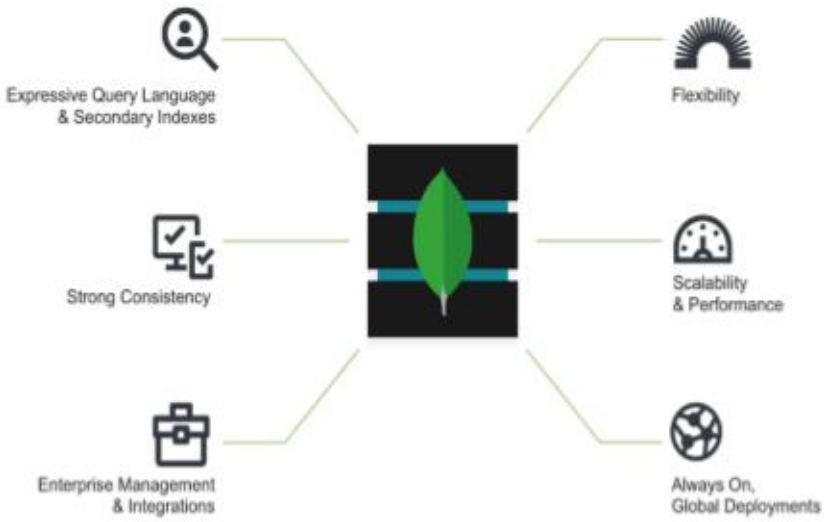



Figure2 : Nexus architecture of MongoDB [7]

The Architecture of MongoDB is completely based on the mixture of relational databases and non-relational databases as it incorporates the features of both of them . MongoDB has the ability to perform strong query processing language which provides both the analytical as well as the operational processes. The insertion of the new data which might be changing and adding should be read quickly [4].

MongoDB is designed in a Multi-Model architecture which means there exists a multi storage system contained in a single environment . There is automatic management of data storage in MongoDB because of the factor there exists data replication between storage engines. For the representation of the data inside MongoDB, key-value pair is being utilised.[5]

The MongoDB data model is schema less with each document stored in binary JSON document format . JSON format is an open standard for a lightweight data interchange based on java script language . In JSON format there is name value pairs which is unordered for every object stored inside .Every object begins with left brace and ends with right brace . Value can contain an object and array. [6]

For the distribution of large datasets , MongoDB uses the property of sharding by which datasets are to be distrusted across various machines. And thus , it is successful in scaling data horizontally unlike in traditional databases.


Figure 6: MongoDB Data Model  storage architecture [7]

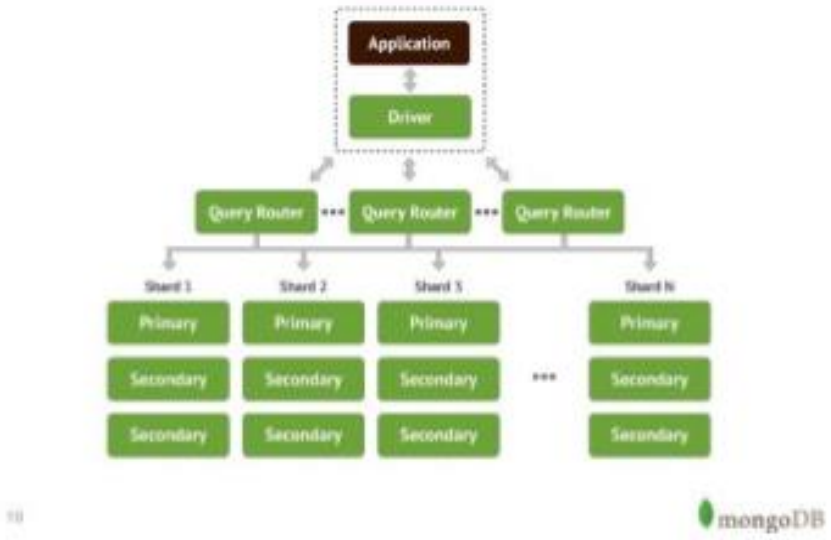



Figure 7 : MongoDB Architecture- MongoDB as a service [8]

## Architecture of Hbase.

Apache Hbase is a column oriented database model which has the capability to store data inside tables. The tables have a row-id which has the collection of several column families which are present inside the table. The column families present are stored inside in a key-value pair [9].It provides low latency read and write operations on top of HDFS. It stores the data in an extremely fault tolerant fashion which is good for storing sparse data. Just like

MongoDB , Hbase also comes up with a schema less structure where there are no predefined columns. It has an architecture of a Master - Slave , where there is one master node and several slave nodes or region servers . When the user sends a write / read request , it is received by the master node (HMaster) and then forwarded to the corresponding slave node [10].

The Architecture of Apache Hbase constitutes 3 important components :

I. H-master
II. Region server
III. Zookeeper

H- Master : It serves like a master node i.e it acts like a master server in the architecture of HBase. It monitors all the region server (slave node ) instances present inside different clusters . In an architectural model which is distributed and has several nodes residing in a cluster it runs on the Name node machine . The master node is responsible for maintaining various nodes in a cluster and plays a vital role in performance. It assigns services to different region nodes. If a user wants to modify the schema or change some meta data operations, the master node takes care of such operations . [9]

Region server : They operate as slave nodes . All the read and write operation of the user is directly taken over by the region server . The client server or the client does not require any permission from the master node for their communication . They are mainly responsible for managing and operating data present inside the distributed cluster. The slave node operate on the data node present inside the Hadoop cluster. It is also responsible for deciding the size of the region when data is spitted. [9]

Zookeeper : Inside Hbase , Zookeeper is a monitoring server that plays a vital role in maintaining the information configuration and offers distributed synchronization which gives access to the distributed process running across different nodes. The client needs to approach the zookeeper first in order to communicate with the regions. Zookeeper is also used to track and server failure and network partitions. [9]

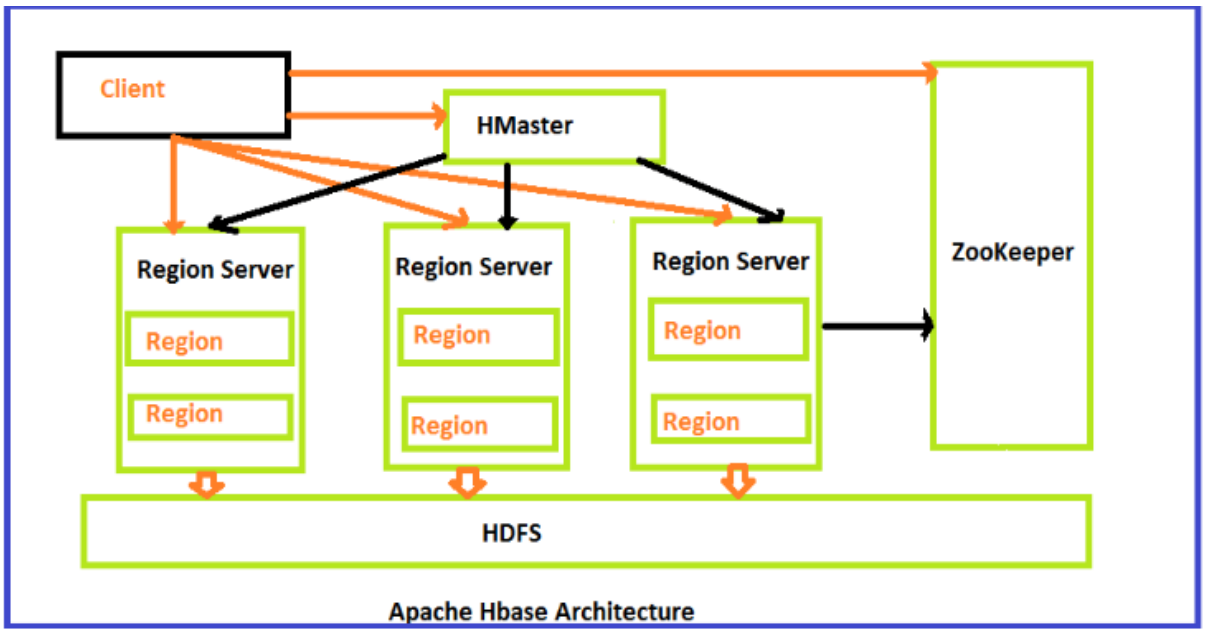



Figure 8 : Apache Hbase Architecture [9]

# Security

Today in the world of technology , where everything is data based , this Big Data is stored in huge databases . The data from these databases are being breached rapidly . According to the predictions , cybercrime will be costing about 6 trillion dollars every year by 2021. The Organizations will have to tackle new threat issues related to phishing , ransomware, etc [5]. It has become very important to maintain the security of these databases in order to maintain the privacy of a user / organisation . Security of NoSQL DBMS is of utmost importance . According to recent researches and evidences the attacks on these databases are increasing every day . These databases are quite susceptible to such attacks and requires a strong security mechanism to maintain the data integrity and avoid data theft . Some of the security measures taken by NoSQL databases like MongoDB and Hbase provide features such as authentication , access control and encryption of data

## *Security in MongoDB*

For the purpose of security MongoDB provides various features some of which are Authentication , access control , encryption of database in order to secure the MongoDB deployments . This NoSQL database supports plain MongoDB CR which is used for user authentication which is available in its open source version . For the secure connection between user and MongoDB clusters it uses SSL along with X.509 . It also uses this feature available for intra-cluster authentication . The feature of access control is enabled by making use of a system defined and custom defined roles with respect to a sharded cluster . In the case of data encryption , it is provided at a transport level with the help of SSL and basic

version of data operation auditing is available . The database security configurations is supported at a low level which completely relies on the database administrator .[11]

MongoDB has various extensive features that are able to detect, defend and control access .Some of them are discussed above :

*Authentication :* This database supports number of authentication mechanisms that is used by the user to verify their identity which allows MongoDB to integrate into an already existing authentication system of the user . Mechanisms like SCRAM, x 509 certificate authentication is used for multiple authentication . It also provides integration with Windows active directory , Keberos , LDAP and thus simplify the access control to database. [1]

*Authorization :* To control the access of MongoDB system it employs Role Based Access Control (RBAC) . According to this a specific user is granted one or more roles which determines the access given to that user for database operations and resources . Outside from this role assigned to that user the user has no access . MongoDB gives the option to the administrator to define views which displays only a subset of data which is a view used to mask specific fields.[1]

*Auditing* : Mongog and Mongos comes up with feature of auditing . This auditing facility gives the administrator and user the advantage to trace system activity for deployments with many users. The native MongoDB audit logs can be used to track access and operations performed on database which can be utilized for security administration .

*Encryption* : The data inside MongoDB database can be encrypted with ease over the network , on disks or backups. The encryption engine provides encryption of data at-rest .Encryption at-rest when used in combination with transport encryption , helps ensure security and privacy standard . Also by encrypting the database the staff who has the authorization credentials can only access the encrypted data providing security and privacy of data.[1]

## ***Security in Hbase***

For the purpose of security Hbase also provides various features some of which are Authentication , auditing logs ,access control of database in order to secure the Hbase deployments. Hbase supports token based authentication for the purpose of map- reduce operations . It not only uses token based authentication but also provides user authentication which makes it secure and user privacy oriented . For the purpose of user authentication it uses the feature of SASL (Simple Authentication and Security Layer) with

the combination of Kerberos which is based on every connection . In addition to this authorization in Habse are controlled by ACL (Access Control Lists) or Coprocessors along with column family level of granularity as per user bases. Hbase also comes up with the feature of logging support which is upto the level of data node. But however, like in Mongo other monitoring and high level auditing features are absent in Hbase . It also lacks configuration security and encryption of data at rest as available in MongoDB [11]. Since Hbase relies on Zookeeper and Hbase therefore it relies on a secur HDFS and secure Zookeeper .

Some of the mechanisms provided by Hbase to secure various components and aspects and the way it relates it to the rest of Hadoop infrastructure as well as client are as above :

- Hbase provides Secure HTTPs due to which the Web user interface becomes secure . Which is done manually by configuring the conf files . It also uses SPNEGO for kerberos authentication along with Web UI's.
- Hbase Native Security : One of the most important mechanisism of Hbase security Is the "wire encryption". It is set between different nodes as well as between external client nodes and Hbase nodes . This wire encryption is set on the top of Kerberos protocol and architecture for authentication purpose , key distribution and ticket issuing to make more deployments by a dedicated Kerberos . Thus , Encryption can thus be activated and configured separately for different layers.[13]
- VPN protected virtual private cloud : One approach to secure Hbase deployments in cloud is by operating the entire cluster within a VPC , that is logically separated from the cloud provider infrastructure and accessed by different network connection. In this way all the traffic from the cluster and the external clients use this connection and are easily secured from eavesdropping and manipulation from VPN solutions [13]
- User Authentication : Hbase can be configured to provide user authentication which makes sure that only user can communicate with Hbase . This authorization system is implemented at the RPC level which is based on simple authentication and security layer (SASL) which also supports Kerberos. SASL provides authentication , encryption negotiation , message integrity verification on a per connection bases [14] .
- User Authorization : The Authorization system which is also known as Access controller coprocessor is available from onwards Hbase 0.92 and offers the ability to define Read/write/admin/create authorization policy for a specific user . [14]

# Learning From Literature Survey

The main purpose of the paper "*Experimental Comparative study of NoSQL database",* is to observe the performance of the two main widely employed solutions, viz, HBase & MongoDB and this has been done using YCSB tool. The aim was to assist & support those actors who are interested in cloud computing & Big data and make it easier for them to decide between the two solutions. These contemporary systems are being differentiated on certain criterions, such as, model used, storage methods, performance, consistency, availability, durability, & various other factors.

The configurations that underwent this study were ubuntu 14.10, pc 64 Bit, Core i5 & 6GB Ram, in a physical machine. These experiments were performed as tests and were then compared with the results which were obtained with the following configurations, ubuntu 32bit, OS windows 7 32 bit, 2GB of Ram & 4 GB RAM but in a virtual machine.

A comparative study has been run between the two systems, namely, HBase & MongoDB, This was done to give an insight on the set of indicators to the interested users to be able to decide on adopting of the specific solutions. A set of configurable workloads that consisted 1000 differentiated operations both of writing and reading were executed many times in all different days on databases that contained 600000 records which were loaded initially. It was observed that different mechanisms & optimizations which are employed by the NoSQL solutions to improve the performance actually have a direct impact on the time taken in the execution of various workloads.

It was found that MongoDB is more efficient in comparison to HBase in the read operations, i.e., read mostly, read only, read modify write & heavy update whereas Hbase was observed to be better in write operations, i.e., updating only, data loading, & update mostly. Hbase traces the changes that database undergoes, and data replication takes place automatically as well, this in turn increases the updating process. Whereas, the updating in MongoDB is slower in comparison to the number of updates achieved. It utilizes the locking mechanisms that enhance the time taken to execute the update.

In order to conclude, NoSQL systems react favourably to the changes of scale and are also considered to be the ideal solution to manage the big volumes of data commonly known as big data [15] .

# Performance And Test Plan

In this paper I am going to compare and analyse the performances of NoSQL database , MongoDB and Apache Hbase by using the YCSB tool. Yahoo! Cloud Service Benchmark (YCSB) is a widely used tool to compare and analyse the NoSQL databases based on generated outputs and workloads created by it. YCSB has an inbuilt workload executor that takes care of different client threads and executes the CRUD operations. It generates set of scenarios like read/write , insert , update operations on the basis of which test analysis and comparison will be done for different workloads .

## Methodology :

For the analysis and comparison the platform used was an Open Stack Cloud which was setup by National College of Ireland (NCI) . On this open stack an instance was created on which all the required installations were made. All the programs were run on a 4 GB RAM, 40 GB of Disk of open instance. The Programs that needed to be installed in the virtual machine are as above :

- Hadoop 2.8.2
- Hbase 1.2.6
- MongoDB 3.2.10
- YCSB 0.11.0

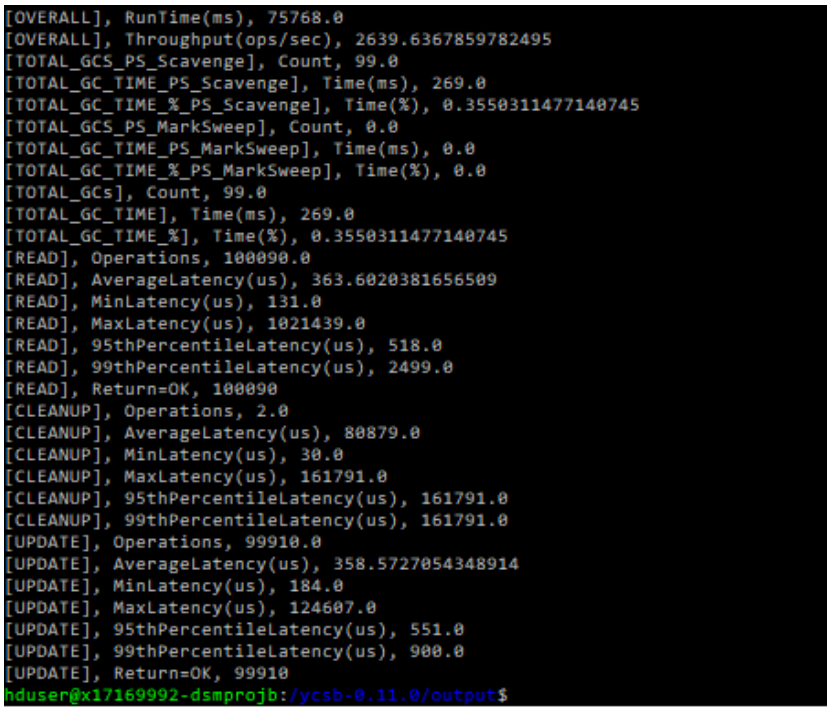
Also for the working and part of installation of these programs additional supporting software's were installed such as Python , Open JDK Java Runtime Environment , Java Development kit Open JDK , RSYNCH and a Secure shell . YCSB was used to generate read write , update and insert operations for different workloads which is used for benchmarking . YCSB provides set of predefined workloads used for benchmarking such as Workload A,B,C and D . For our analysis we have chosen Workload A and C since they come up with different read , insert and update operations. The workload used are as above :

| WORKLOAD | Operation | Type | Application |
|---|---|---|---|
| Workload A | 50 read and 50 write | Updates heavy workload | Used in session store recording |
| Workload C | 100 % read | Read Only | User profile cache (Eg : Hadoop) |

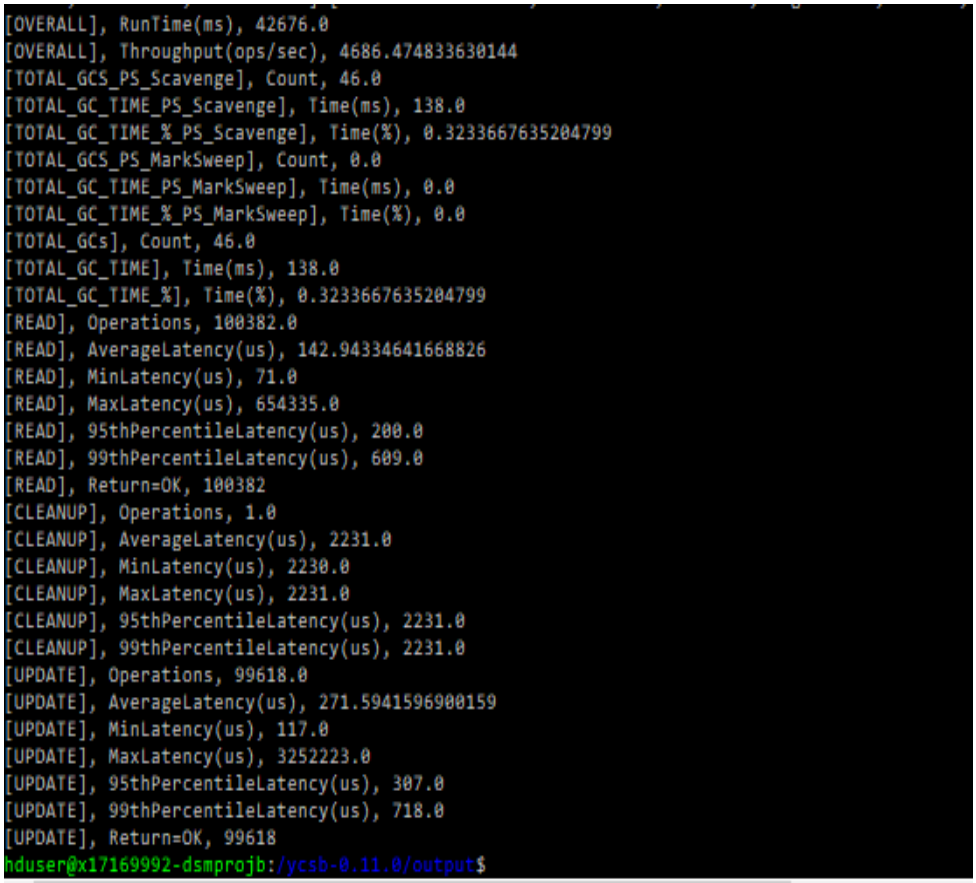
Both the Databases were tested and analysed on the same operational count which has 1 KB rec size . The given op-counts for both workloads are as follows :

| Operational Count | 50,000 | 100,000 | 150,000 | 200,000 |
|---|---|---|---|---|

For Workload A at the record count of 200,000 running on Hbase the result obtained are :




For Workload A at the record count of 200,000 running on MongoDB the result obtained are



# AVERAGE WORKLOAD CALCULATIONS

*RESULTS FOR MongoDB at Workload A*

| TEST 1 | MongoDB | | Workload A | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 4283.755 | 5522.42 | 5068.93 | 4686.47 |
| Read Operation | 25049 | 49949 | 75200 | 100382 |
| Avg Read Latency | 151.122 | 139.15 | 135.44 | 142.94 |
| Update Operation | 24951 | 50051 | 74800 | 99618 |
| Avg Update latency | 277.31 | 204.11 | 238.78 | 271.59 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 390.39 | 264.78 | 165.74 | 244.09 |

| TEST 2 | MongoDB | | Workload A | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 5378 | 4733.50 | 5320 | 5880.62 |
| Read Operation | 24904 | 49925 | 75075 | 100163 |
| Avg Read Latency | 135.78 | 136.11 | 125.01 | 129.24 |
| Update Operation | 25096 | 50075 | 74925 | 99837 |
| Avg Update latency | 209.53 | 267.22 | 238.24 | 199.49 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 167.90 | 264.78 | 176.16 | 219.70 |

| **Average Results** | MongoDB | | Workload A | |
|---|---|---|---|---|
| | | | | |
| Throughput | 4830.878 | 5127.96 | 5194.465 | 5283.545 |
| Read Operation | 24976.5 | 49937 | 75137.5 | 100272.5 |
| Avg Read Latency | 143.451 | 137.63 | 130.225 | 136.09 |
| Update Operation | 25023.5 | 50063 | 74862.5 | 99727.5 |
| Avg Update latency | 243.42 | 235.665 | 238.51 | 235.54 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 279.145 | 264.78 | 170.95 | 231.895 |

## ***RESULTS FOR MongoDB at Workload C***

| Test 2 | MongoDB | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 7151.11 | 6955.01 | 8946.67 | 9696.96 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 124.1 | 135.21 | 104.67 | 98.68 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 177.50 | 178.79 | 158.80 | 162.20 |

| Test 1 | MongoDB | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 7697.044 | 6955.07 | 9106.91 | 8909.87 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 116.85 | 135.21 | 103.09 | 106.61 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 192.35 | 149.91 | 138.334 | 178.603 |

| **AVERAGE RESULT** | MongoDB | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 7424.077 | 6955.04 | 9026.79 | 9303.415 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 120.475 | 135.21 | 103.88 | 102.645 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 184.925 | 164.35 | 148.567 | 170.4015 |

## ***RESULTS FOR Hbase AT Workload A***

| Test 1 | Hbase | | Workload A | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 2333.7222 | 2968.152 | 2837.255 | 2639 |
| Read Operation | 24951 | 49952 | 74930 | 100090 |
| Avg Read Latency | 323.64 | 275.25 | 301.19 | 363.6 |
| Update Operation | 25049 | 50048 | 75070 | 99910 |
| Avg Update latency | 408.05 | 337.17 | 363.51 | 358.57 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 390.62 | 370.69 | 377.2 | 388.12 |

| Test 2 | Hbase | | Workload A | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 2855.51 | 3056.98 | 2732 | 3248.1 |
| Read Operation | 25184 | 49993 | 74996 | 99805 |
| Avg Read Latency | 241.24 | 270.63 | 346.13 | 266.88 |
| Update Operation | 24816 | 50007 | 75004 | 100195 |
| Avg Update latency | 384 | 342.60 | 345.12 | 326 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 394.69 | 384.24 | 378.24 | 394.60 |

| **AVERAGE RESULT** | Hbase | | Workload A | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 2594.616 | 3012.566 | 2784.628 | 2943.5 |
| Read Operation | 25067.5 | 49972.5 | 74963 | 99947.5 |
| Avg Read Latency | 282.44 | 272.94 | 323.66 | 315.24 |
| Update Operation | 24932.5 | 50027.5 | 75037 | 100052.5 |
| Avg Update latency | 396.025 | 339.885 | 354.315 | 342.285 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 392.655 | 377.465 | 377.72 | 391.36 |

## ***RESULTS FOR Hbase AT Workload C***

| Test 1 | Hbase | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 3342.91 | 3858.91 | 3906 | 3015.36 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 262.277 | 238.38 | 243.89 | 293.54 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 545.09 | 582.09 | 525.09 | 519.27 |

| Test 2 | Hbase | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 3399.21 | 3800.16 | 3889 | 3216.12 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 258.12 | 230.12 | 239.11 | 285.56 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 550.31 | 577.65 | 500.13 | 522.24 |

| AVERAGE RESULT | Hbase | | Workload C | |
|---|---|---|---|---|
| | 50000 | 100000 | 150000 | 200000 |
| Throughput | 3371.06 | 3829.535 | 3897.5 | 3115.74 |
| Read Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Read Latency | 260.1985 | 234.25 | 241.5 | 289.55 |
| Insert Operation | 50000 | 100000 | 150000 | 200000 |
| Avg Insert Latency | 547.7 | 579.87 | 512.61 | 520.755 |

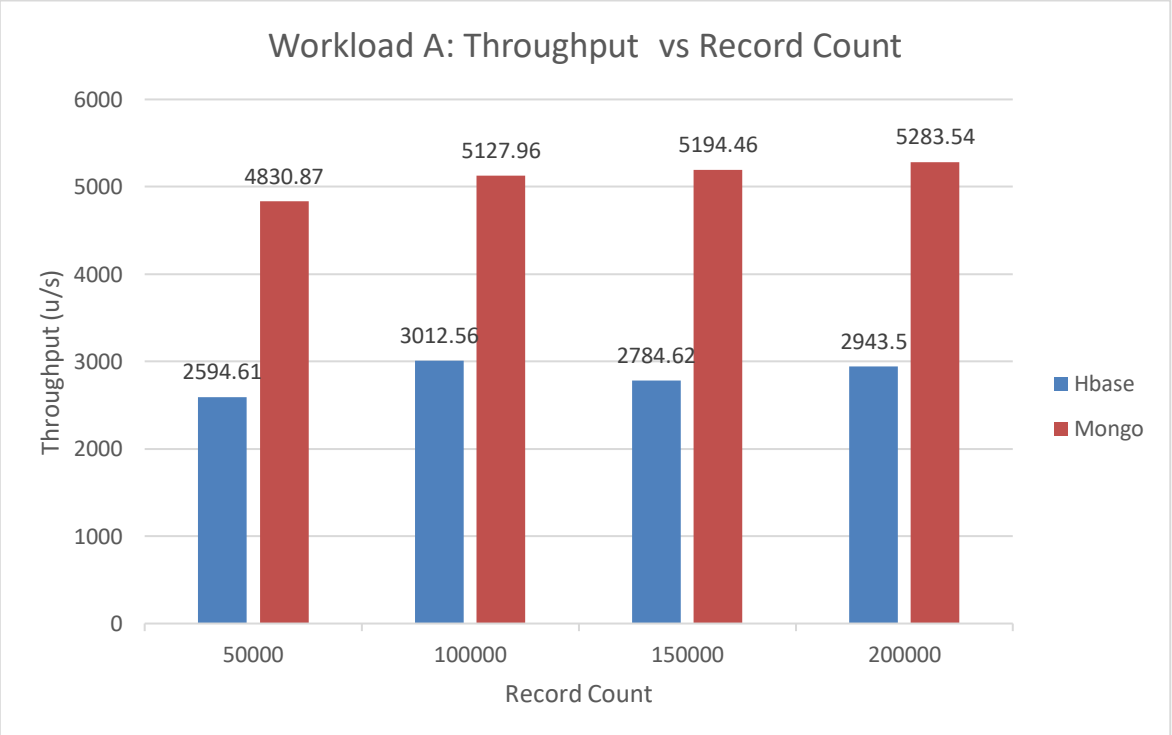
# Result And Analysis

The Graphs below illustrates the results of the performances between MongoDB and Hbase based on Workload A and C .

### *Results and Analysis of Workload A*

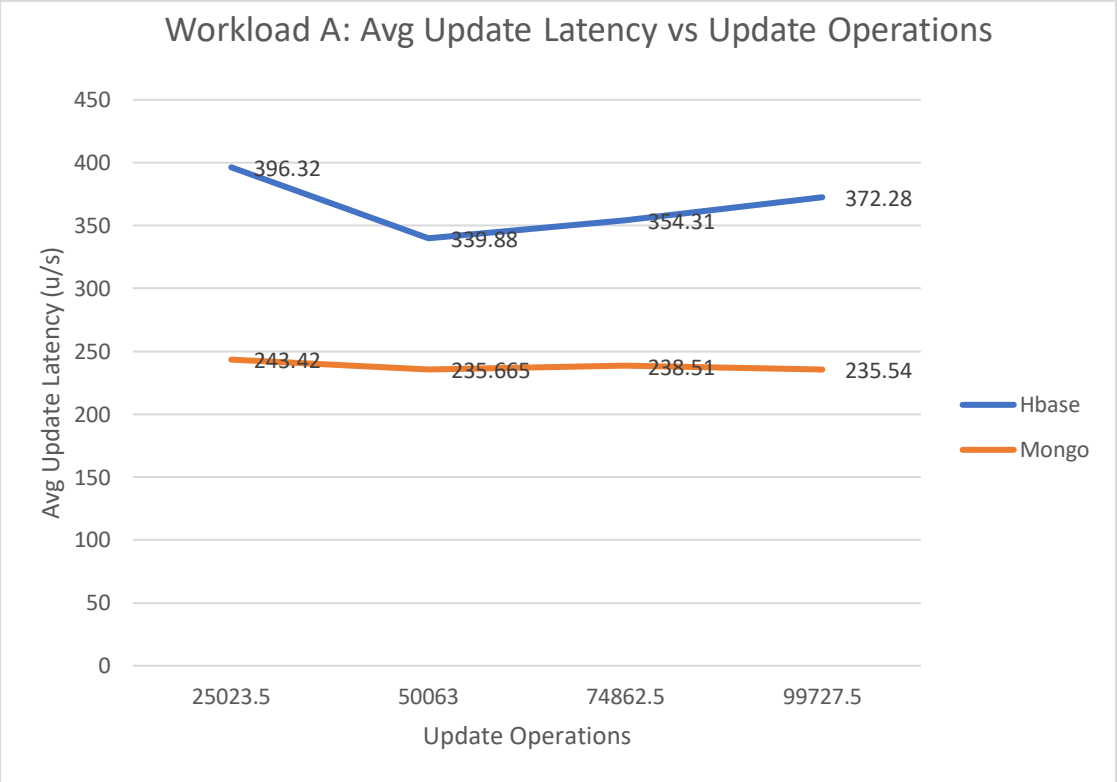
*Throughput and Record Count Analysis :*

| | MongoDB | HBase |
|---|---|---|
| Workload C | Throughput(ops/sec) | Throughput(ops/sec) |
| 50,000 | 4830.87 | 2594.61 |
| 100,000 | 5127.96 | 3012.56 |
| 150,000 | 5194.46 | 2784.62 |
| 200,000 | 5283.54 | 2943.5 |



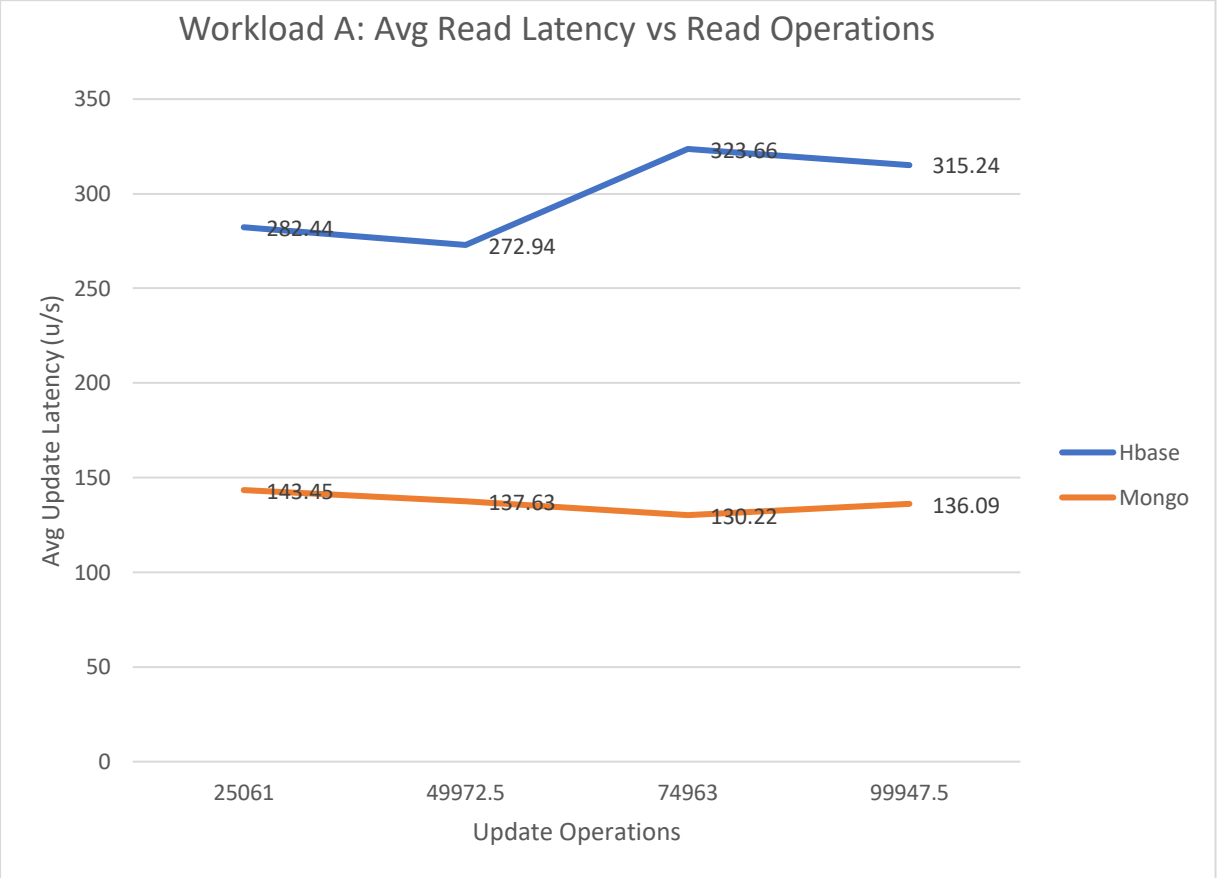
The throughput comparison of both the databases gives us information about which of the database performance is better in terms of operations per second. From the above bar graph it is  illustrated that the performance shown by MongoDB has clearly outperformed the performance shown by Hbase at every record count . At record count of 200,000 MongoDb has shown a peak performance where at 100,000 record count Hbase reaches its peak.

***Average Update latency and Update Operation Analysis  :***



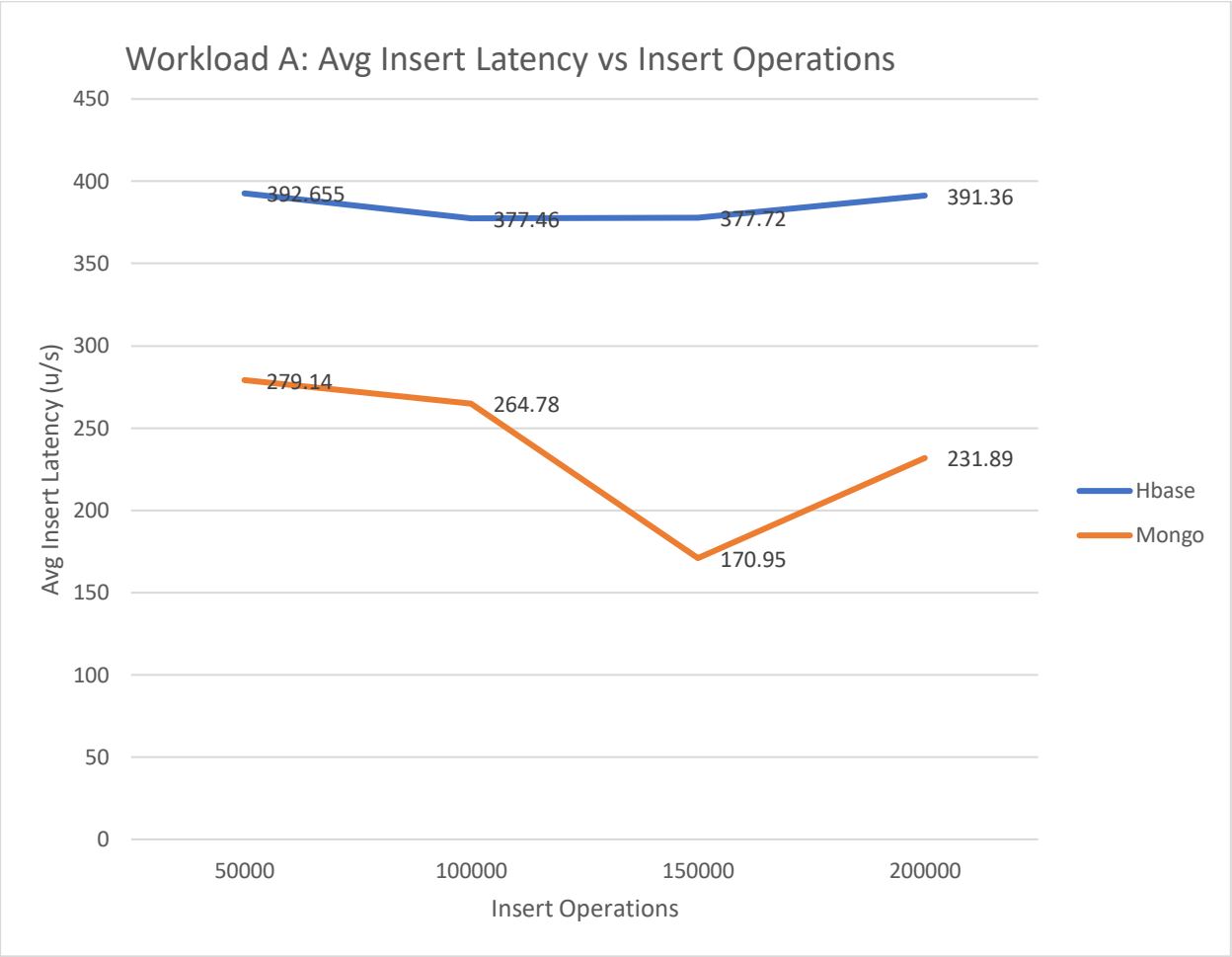

The above line graph depicts the recorded average latency recorded in milliseconds for Update Operations of MongoDB and HBase. The graph illustrates that the average latency of Hbase is greater than the average latency shown by  MongoDB for all the four operations . The average latency was least for both the databases at 50063 update operation .It can be concluded that the performance of MongoDB has shown a better performance when the records are being updated in the databases .

***Average Read latency and Read Operation Analysis :***




In the above given graph the first line depicts the read operation that has been run on Hbase and MogoDB which has been compared with the average read latency for Workload A which by default is for 50 % read and 50 % update. The line graph clearly depicts that the average latency of MongoDB is less as compared to the average latency of Hbase database. Also MongoDB remains to be more constant. When workload was increased Hbase showed increase in average latency . Hence it can be concluded that in MongoDB performs better in terms of reading the record as compared to Hbase even when increasing the workload.

***Average Insert latency and Insert Operation Analysis :***

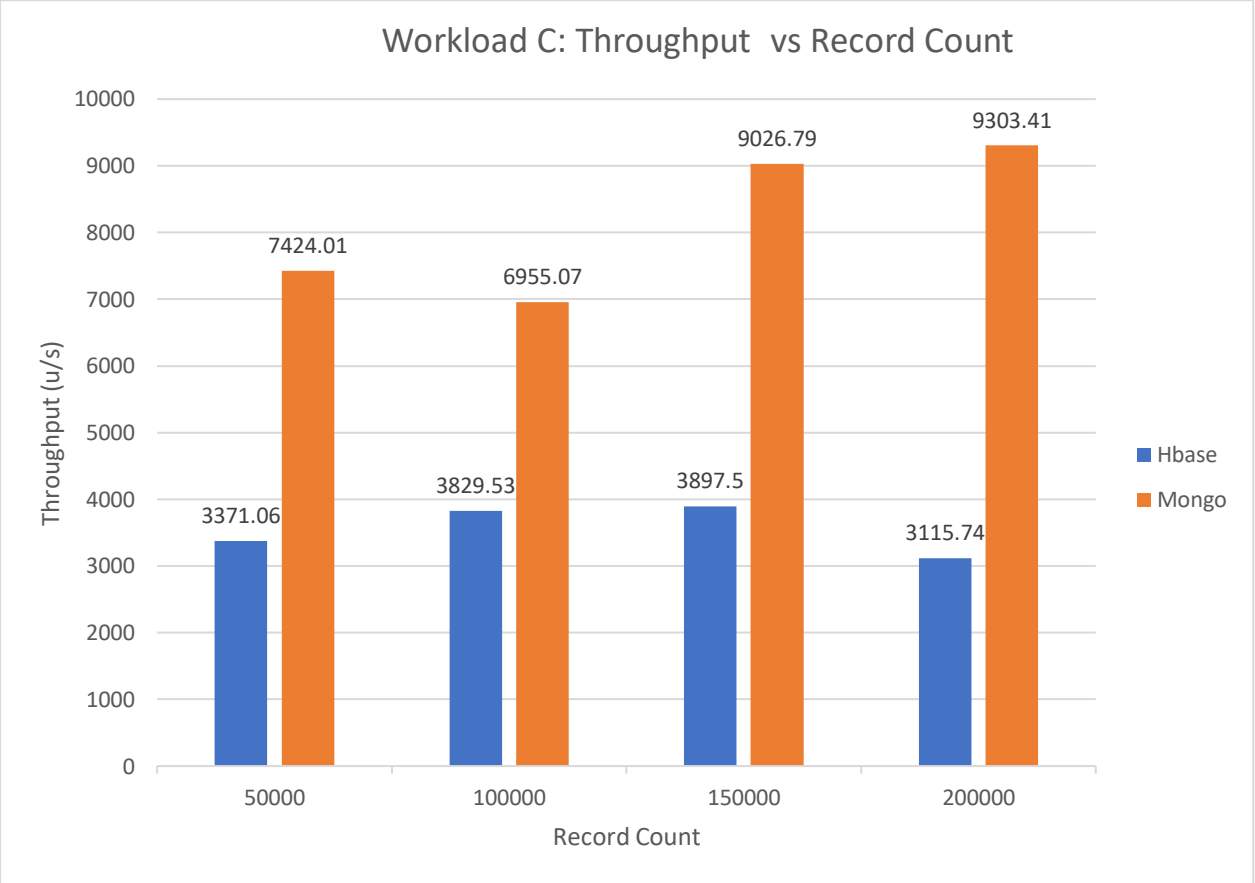



The above line graph depicts the insert average latency recorded in milliseconds for Insert Operations of MongoDB and HBase for Workload A. The graph illustrates that the average latency of Hbase is greater than the average latency shown by MongoDB for all the four operations .It can be concluded that the performance of MongoDB has shown a better performance when the records are being inserted in the databases .

## *Results and Analysis of Workload C*

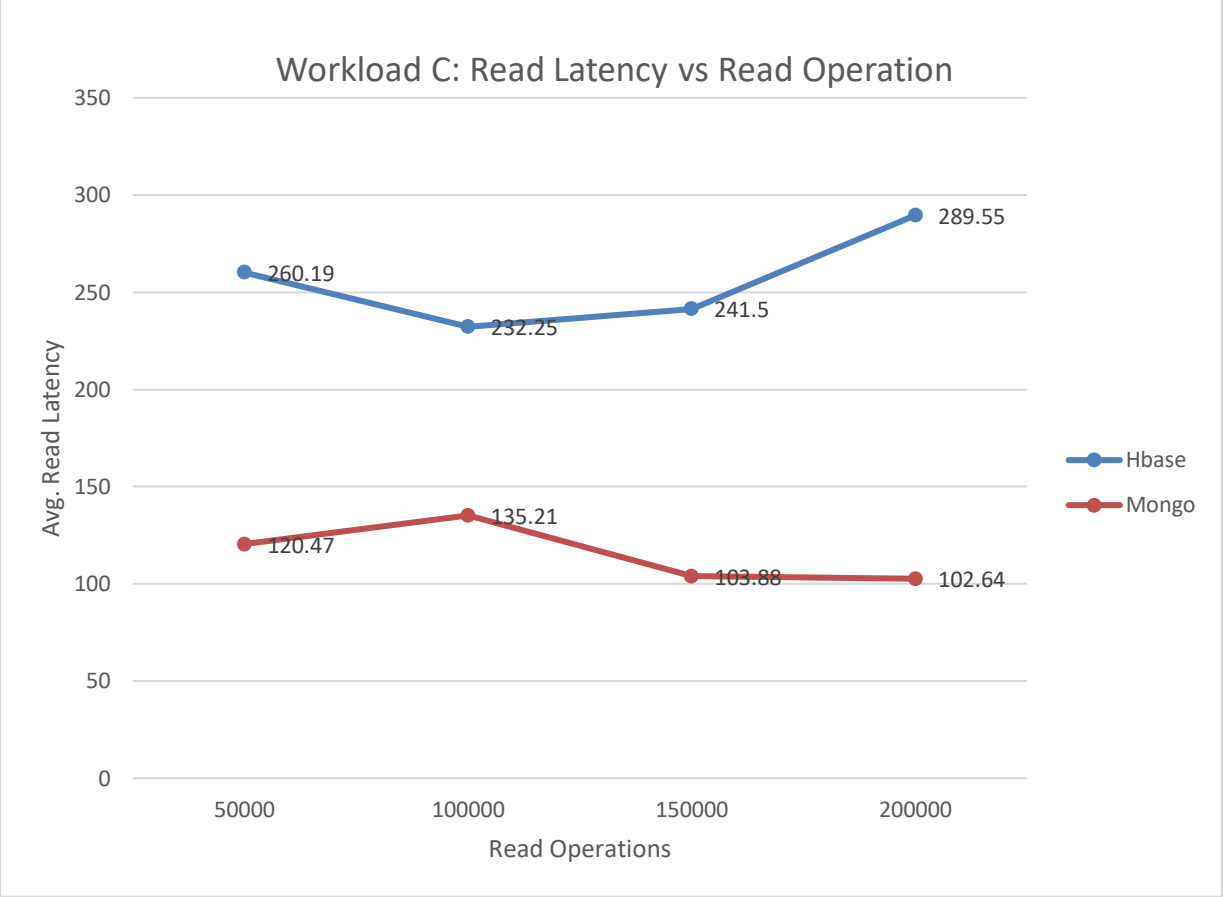
*Throughput and Record Count Analysis*

| | MongoDB | HBase |
|---|---|---|
| Workload C | Throughput(ops/sec) | Throughput(ops/sec) |
| 50,000 | 7424.01 | 3371.06 |
| 100,000 | 6955.07 | 3829.53 |
| 150,000 | 9026.79 | 3897.5 |
| 200,000 | 9026.79 | 3115.74 |



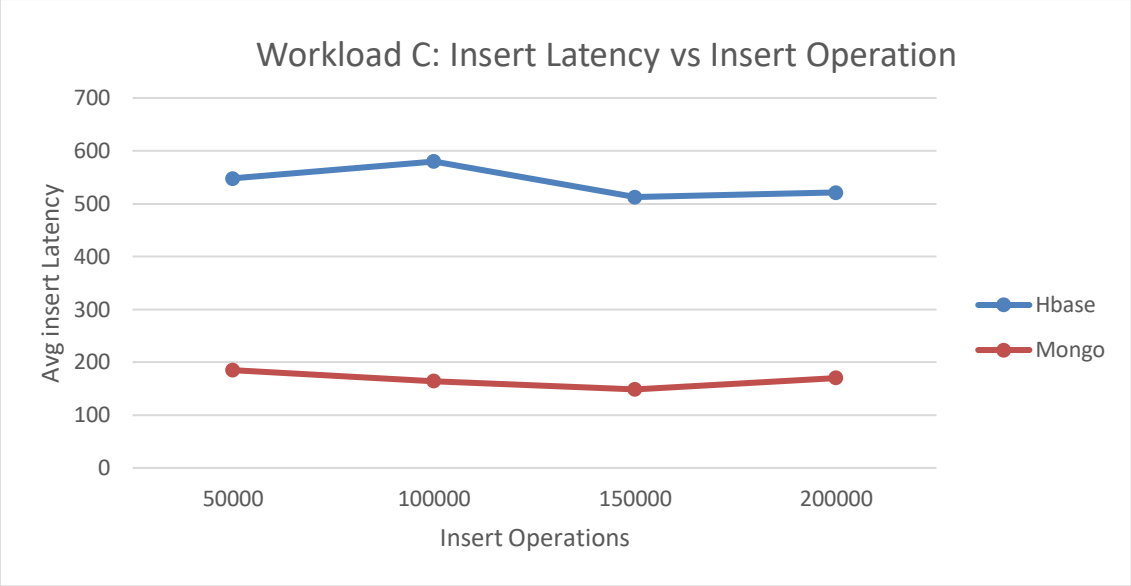

The Graph shows the throughput for Workload C (100% read operation ).The throughput comparison of both the databases gives us information about which of the database performance is better in terms of operations per second. From the above bar graph it is illustrated that the performance shown by MongoDB has clearly outperformed the performance shown by Hbase at every record count . At record count of 200,000 MongoDb has shown a peak performance.

***Average Read latency and Read Operation Analysis :***




In the above given graph the first line depicts the read operation that has been run on Hbase and MogoDB which has been compared with the average read latency for Workload C which by default is for 100% read operation. The line graph clearly depicts that the average latency of MongoDB is less as compared to the average latency of Hbase database. Also MongoDB remains to be more constant. Hence it can be concluded that in MongoDB performs better in terms of reading the record as compared to Hbase even when increasing the workload.

***Average Insert latency and Insert Operation Analysis :***




The above line graph depicts the insert average latency recorded in milliseconds for Insert Operations of MongoDB and HBase. The graph illustrates that the average latency of Hbase is greater than the average latency shown by MongoDB for all the four operations ..It can be concluded that the performance of MongoDB has shown a better performance when the records are being inserted in the databases .

## CONCLUSION

On performing the comparative test analysis of both the databases , MogoDB and Hbase for Workload A and Workload B at different Record Counts for operations like Read , Insert And Update which was analysed using the YCSB benchmarking tool we come to know that :

1. For Workload A : When compared the throughput with the record operations it was concluded that wholly at the workload till 200,000 the performance of MongoDB was outstanding comparatively. When both the databases were compared on the basis of Read count and Average read latency as well as when compared Insert operation with average insert operation , it was clear that MongoDB has performed better in both the cases. It showed a constant performance for Read operations where as for Insert operations it fluctuated .
2. For Workload C : Also in the case of Workload C which is 100% read by default for every operation MongoDB has shown a better performance than Hbase . When compared for Insert operations MongoDB not only performs better but maintains a constant pase . These test results have been analysed till record count of 200,000 for which Read , Update and Insert operation comparisons MongoDB has performed better overall

# REFERENCES


[1] "Open Source Document Database", *MongoDB*, 2018. [Online]. Available: https://www.mongodb.com/. [Accessed: 1- Dec- 2018].

[2] "Apache HBase – Apache HBase™ Home", *Hbase.apache.org*, 2018. [Online]. Available: https://hbase.apache.org/. [Accessed: 7- Dec- 2018].

[3] "Apache HBase – Apache HBase™ Home", *Hbase.apache.org*, 2018. [Online]. Available: https://hbase.apache.org/book.html . [Accessed: 7- Dec- 2018].

[4] "MongoDB Architecture Explained", *Ibmbpnetwork.com*, 2018. [Online]. Available: https://www.ibmbpnetwork.com/linux-blog/mongodb-architecture. [Accessed: 9- Dec- 2018].

[5]"MongoDB Architecture | MongoDB", *MongoDB*, 2018. [Online]. Available: https://www.mongodb.com/mongodb-architecture. [Accessed: 8- Dec- 2018].

[6] Jongseong yoong, Doown Jeong. Forensic investigation framework for the document store NOSQL DBMS: MongoDB as a case study. June 2016, Page 53-65

[7] "MongoDB Arquitectura y modelo de datos - Sitio Big Data", *Sitio Big Data*, 2018. [Online]. Available: http://sitiobigdata.com/index.php/2017/12/27/mongodb-architecture-data-model/. [Accessed: 9 - Dec- 2018].

[8] E.Laxmi Lydia, K. Shankar ,"Correlating NoSQL Databases With a Relational Database: Performance and Space ,"*Research Gate. ,*Vol .118, no.7,235-244,2018.

[9] *Guru99*, 2018. [Online]. Available: https://www.guru99.com/hbase-architecture-data-flow-usecases.html. [Accessed: 8- Dec- 2018].

[10] "Overview of HBase Architecture and its Components", *DeZyre*, 2018. [Online]. Available: https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295. [Accessed: 11- Dec- 2018].

[11] Anm Zahid, Rahat Masood and Muhhamad Awais Shibli, "Security of Sharded NoSQL databases" , 4799-5852, 2014

[12] Ben Spivey, Joey Echeverria , Hadoop : Security Protecting your big data Platform. 2015

[13] Frank Pallas, Johannes Gunther , David Bermbach," Pick your choice in Hbase : security or performance,"*IEEE* .DOI : 10.1109 , Dec.2017.

[14] M. Bertozzi , "How-to: Enable User Authentication and Authorization in Apache HBase - Cloudera Engineering Blog", *Cloudera Engineering Blog*, 2018. [Online]. Available:

https://blog.cloudera.com/blog/2012/09/understanding-user-authentication-and-authorization-in-apache-hbase/. [Accessed: 13- Dec- 2018].

[15] Houcine Matallah, Ghalem Belalem and Karim Bouamrane ,"Expirimental comparative study of NoSQL databases : HBASE versus MongoDB by YCSB ,"*Research Gate.,* July 2017.

[16] "MongoDB Features - javatpoint", *www.javatpoint.com*, 2018. [Online]. Available: https://www.javatpoint.com/mongodb-features. [Accessed: 7- Dec- 2018].