

# Data Accessibility based Performance Assessment of MySQL Vs. MongoDB Data Storage Systems

Rahul Dhande  
18182852  
MSc in Cloud Computing

23rd April 2020

## Abstract

This research theme presents a detailed analysis and evaluation of the MySQL (Relational database) and MongoDB(Document based) by major focus on High availability and performance. Nowadays, Considering the increasing demand and amount of data has to manage, and the store is a significant concern for Information and telecommunication Industries. Subsequently, This massive amount of unstructured data affects the performance and efficiency of the database system. This complex kind of data requires proper mechanisms for storage and standards for access.

This report theme introduced MySQL and MongoDB are robust and competitive database platforms help to tackle this problem. The functionally includes storage size and the access time of both databases are different and unique in their own way. This report deals with as well as provides database storage architectures, standards, unique features and limitations of NoSQL databases over relational databases that will give the best fit and highly accessible for Industries for data store and management. Moreover, This report reviews a proper performance test plan and benchmark analysis of both databases using YCSB tool and Testharness.

**Keywords:** *MySQL, MongoDB, Performance, YCSB, Testharness and Workload*

## 1 Introduction

Nowadays, Day by day growing massive amount of Data are the most significant resource for Information and telecommunication industries. Since data generate from big companies such as Google, Twitter and Facebook and others as well as from IoT devices makes the complicated situation for management. This unstructured type of data requires proper standards for storage and management.(Fraczek[1] 2019,Gizem[2]2017) The aim to tackle this problem, Relational database has been used widely over the last 30 years. The industries and business sectors popularly use Relational Database Management System (RDBMS) like MySQL, Oracle, Microsoft SQL Server, IBM DB2 and PostgreSQL. These databases are well recognized, but as compared to these, MySQL is known for its faster execution of queries and most commonly used data storage facility for web applications due to its quick read data speed.(Ongo [3],2018).

According to (Shetty [4],2019) Due to the rise in undeniable unstructured data, Relational databases may be less effective when processing vast volumes of data at a greater speed. As compared to these, the NoSQL (Not-Only SQL) databases such as MongoDB are a Better and simpler way to manage heavy data with higher scalability, a stronger and more versatile data model and superior efficiency. It is a classical way for data retrieval and storage and offers a more robust data model, thus integrating many key features of relational databases.(Patil[5],2019) Replication of the relational database can be limited and often based on accuracy and not availability. On the other hand, NoSQL has a different reputation as well as being known to be

good at storing high amounts of data, and NoSQL's schema-free feature improved the database scalability.(Ongo [3],2018)

The MySQL and MongoDB have a different set of features and limitations which helps business sectors to select the best fit as per their requirements. (Fraczek[1])

Property	Sql	NoSql
The way of storing the data	Tables	Documents, key value,
Organization of data	A predefined scheme	A dynamic scheme
Scalability( increase in performance)	Vertical(larger ram, stronger processor)	Horizontal(more servers, instances)
Query language	Standardized Sql	Own query language
Data intercourse	Foreign keys	Nested documents
Security	Transactions, consistency, isolation	Does not exist

Figure 1: Comparison of MongoDB(Non-Relational) Over MySQL(Relational) [1]

There are various types of benchmarking tools that are available in the market to test the databases. The open-source Yahoo! Cloud Server Benchmark (YCSB) is one of the commonly used and efficient ways to provide comparative comparisons for relational and NoSQL databases, datasets, supports a wide range of database bindings and offers a number of required workloads.(Mahajan[6],2019). In this experiment we are present analysis of two databases using the same tool the main reason by using this tool is to its provides accurate information of all operations and gives a proper performance analysis.

Today's applications and IoT devices require proper data management and standards, rapid creation of features and flexible deployment. For this purpose in this report, We aim to focus on presenting a performance evaluation analysis of MySQL and MongoDB on the basis of higher data accessibility. For evaluation, we are going to use Yahoo! Cloud Server Benchmark (YCSB) benchmarking tool show proper results by taking considerations of different workloads A,D and E.

## 2 Key Characteristics of Chosen Data Storage Management Systems

### 2.1 Document Based:MongoDB

The following characteristics are making MongoDB popular and easily available ([7],2019)

1. Dynamic scheme based Ad-Hoc Queries:-  
MongoDB supports dynamic scheme type organized data which show the feature of ad-hoc queries for organizing data. Ad-hoc queries are updated in real-time, which results in better performance
2. Without Schema Database:-  
It has no schema, so it may have several different fields, content, and size than another document in the same set as because of this MongoDB shows versatility in managing the databases.
3. Flexible Indexing  
In MongoDB, any field indexed with primary and secondary indices can be indexed. Making query searches quicker, increasing the efficiency of MongoDB indexing.
4. Multi-Replication(Master Slave):-  
Replication is the method which MongoDB uses when it comes to redundancy. This function distributes data to several machines as well as providing primary nodes and

replica sets of one or more. In replication, The secondary node, for instance, is primary when the main node is down for certain reasons. Overall, this saves our maintenance time and guarantees smooth service.

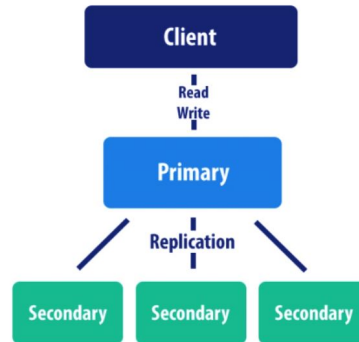


Figure 2: Multi-Replication [7]

5. Aggregation based system:-

MongoDB has an aggregation mechanism that includes batch process data for efficient usability and gets a single result even after performing different operations on group data. The three ways of creating an aggregation system are the aggregation pipeline, map-reduce mechanism and single purpose aggregation methods.

6. Enhanced data storage:-

MongoDB supports GridFS for files storage and retrieval. This splits a document into sections called chunks and stores them in a separate folder, and all chunks except the last chunk have a default size of 255kB.

7. Multiway Sharding:-

This function for larger datasets and assists in spreading this troublesome data to multiple instances of MongoDB. MongoDB collections that have a bigger size are distributed in different sets. The sets are called shards.

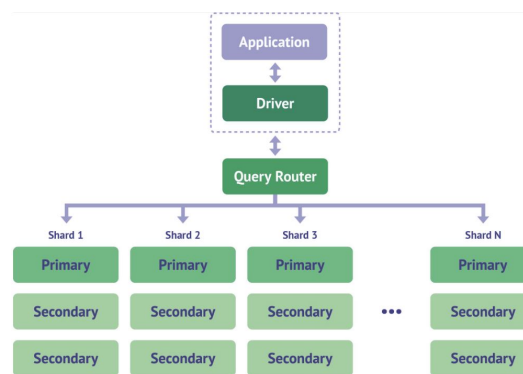


Figure 3: Multiway MongoDB Sharding [7]

### 2.1.1 Characteristics of MongoDB over MySQL

MongoDB can scale-up inside and through several distributed data centres, offering previously unachievable new levels of availability and scalability with relational databases such as MySQL. The versatile data model of MongoDB ensures that the database schema will change

with business needs while the static relational structure of MySQL slows down developers as they have to adapt objects in code to a relational structure.([8],2019)

1. Supports Auto Quick Failover

MongoDB can detect failures natively, automatically select a new primary node in less than five seconds in most cases and Failover in MySQL is a manual process that taxes the operating team at the most crucial moment.

2. Documents render demands quickly

MongoDB document model maps objects in application code automatically and making it simple for developers to know and use while MySQL imposes constraints on the database schema, and data modelling can slow down the development.

## 2.2 Relational:MySQL

MySQL 8.0 adds new enhancements as compared to Non-Relational Databases in following([9],2019)

1. Multi-valued Data Dictionary

MySQL now provides a transactional data dictionary which stores database object information

2. Improved Resource Management

MySQL facilitates the creation and management of resource groups and enables the assignment of threads running within the server to different groups to permit the execution of threads based on the resources available to the group as well as group attributes to monitor its resources, activate or limit the resource use by group threads.

3. Secured table encryption management

Modified table encryption can be handled globally by setting and applying default encryption. The privilege test takes place when creating or modifying a schema or general tablespace with an encryption setting that varies from the default table encryption setting as well as allowing overriding default encryption settings as allowed by table encryption privilege test.

4. Faster Replication with JSON Documents

MySQL improved Replication allows binary logging of partial JSON database changes using a lightweight binary format, saving space in logging over full JSON database logging.

5. Flexible Connection management

MySQL Server allows for the configuration of a TCP / IP port explicitly for administrative connections and offers more control over compression usage to reduce the number of bytes sent to the server by connections.

6. Enhancement in procedures

The server performs tasks previously performed by MySQL update as well as the server now performs all required update tasks automatically at the next startup and is not reliant on MySQL upgrade invoking a DBA. Furthermore, the server updates the contents of the support tables and also monitors how the system performs automated data dictionary and upgrade operations.

7. Upgrade security and account management

MySQL includes functions that are called privilege collections that can be created and removed, which have privileges that can be granted which revoked from them, as well as those that can be granted and revoked from user account. In addition, MySQL preserves password history records, allowing for restrictions on the reuse of past passwords.

Moreover, MySQL sets the access control granted on the named pipe to clients to the minimum required for efficient Windows communication. Without any additional setup, current MySQL client software could open named pipe connections.

### 3 Database Storage Architectures

#### 3.1 MongoDB (NoSQL)

The MongoDB database is a kind of NoSQL document storage database which can realize vast data storage and management. It can be characterized by the firstly horizontal size of the storage nodes is convenient. Secondly, to boost high availability, data can be automatically backed up in multi-nodes. Thirdly, a document object is used as a storage device, and finally, with fine granularity and easy access, the response to the request is fast, the speed is typically millisecond. For document databases such as MongoDB, the essential storage components are collections instead of tables for relational databases. Moreover, a database may be of any specific form of data, such as dates, lists, numbers, strings, or a sub-document. Simultaneously, within a single deployment, it has special different storage engines. In addition, this structure is useful for data transfer between storage engine technologies. Figure shows detailed overview of MongoDB storage architecture includes various functionality of security and management. (B.Jose [10],2017)



Figure 4: MongoDB Data Storage Architecture with WiredTiger Storage Engine [10]

- **WiredTiger Engine-**  
This has competitiveness control and native compression properties as well as the best efficiency in storage and performance. MongoDB enables both memory motor blends for ultra-low latency operations with a disk-based motor for a complete life and the assembly of large-scale, highly available, flexible frameworks, allowing various sensors and applications to store their data in a schematically adaptable manner. Additionally, It also supports pluggable storage engine APIs that allow an outsider to build MongoDB storage engine.
- **Auto multi-valued sharding-**  
Auto Sharding is an important enhancement and form of database partitioning that divides very large databases into smaller, faster, more easily controlled sections called shards

that have more server replica nodes added to the network. Moreover, This is a very fast database and provides indexes not only on the primary attributes but on the secondary attributes as well. Besides, This feature also available in sub-documents.

- Effective query language  
Supports Dynamic queries, sorting, secondary indexes, comprehensive updates, fast aggregation are just a few features of RDBMS in MongoDB. This also provides adaptability and scalability.

### 3.2 MySQL (RDBMS)

MySQL 8.0 is very versatile and provides various types of storage engines as a plugin for different kinds of needs as well as actions explains changes when using storage engines, such as transactional InnoDB and non-transactional MyISAM engines data storage and specific SQL execution methods. In addition, system-specific components such as memory will be used within the server and buffers will be used for SQL operation depending on the type of storage system. The storage architecture of MySQL makes it unique and preferred option for the majority of business sectors as well as industries for data storage and management. Also, it presents the Client-server system.(Lalit [11],2019)

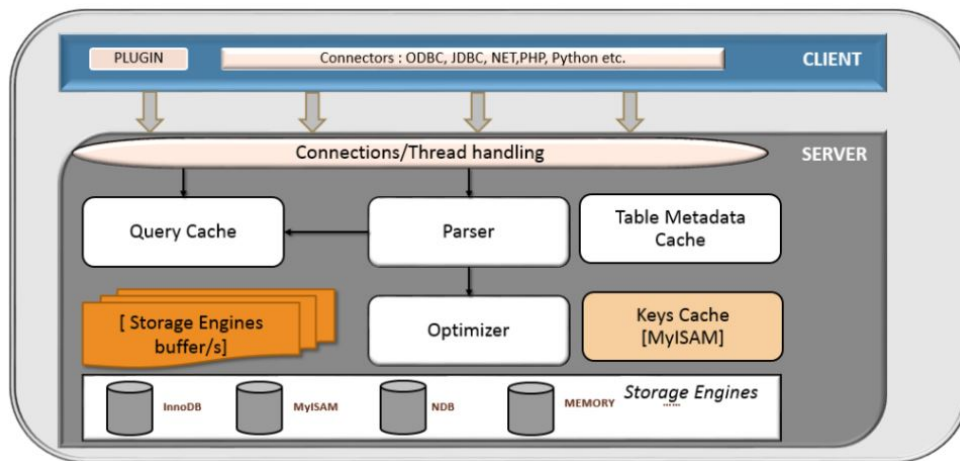


Figure 5: MySQL Storage Architecture with InnoDB Storage Engine[11]

- Multi-valued Client  
Functionality for connecting MySQL file storage and knowledge recovery. It can very well be done by connecting file thread managers such as ODBC or JDBC using the utility driver provided.
- Powerful Server  
MySQL instance where actual data is stored and data processing take place and clients are the applications that link to MySQL database server.
- MySQL Multithreading  
MySQL Server daemon software that runs in the background and handles incoming and outgoing requests from clients relevant to the database. The mysqld is a multi-threaded process that allows multi-session link listening to all connections and managing MySQL instance. Additionally, The server caches all the queries from that client that are executed within that specified thread, so they don't need to be generated and destroyed with each new connection.

- **High Speed Storage Engine**  
Storage engine designed for executing and extracting SQL statements from data files as well as handling physical data (file management) and locations. It is used as a plugin and can load/unload from MySQL server running. There are various storage engines like InnoDB, MyISAM, Memory, Archive, CSV and NDB available for MySQL. The InnoDB knows as high-speed query deployment, completely transactional ACID and provides transactions to REDO and UNDO. Using a shared file to store objects, and it's also applies logical structure, physically stored data, read physical data and construct a logical structure
- **MySQL Optimizer**  
Build an effective database execution plan according to the storage engine, and rewrite the database, order tables scanning and pick the correct indexes to use.

## 4 Comparative research theme of database features

In this section we show the comparative study of MongoDB and MySQL on the bases of High performance and Transaction Management.

### 4.1 High performance and availability

MongoDB provides High Performance and OnDemand Availability to Business sectors as well as industries. Moreover, It achieves performance through various dimensions such as data modelling and sizing of memory, Query patterns and profiling, Indexing, sharding, transactions and Hardware and OS configurations. As a consequence of this data localization, embedding offers improved performance for reading operations because of the ability to request as well as MongoDB works better when the working set (indexes and data most commonly accessed) of the application fits in the memory. For example, the sizing of RAM is the most important factor. Subsequently, MongoDB has the capacity to manage massive unstructured data and is magically faster over MySQL. (Keep [12], 2020, [13], 2020) MongoDB replicates data automatically to additional nodes for high availability and reliability and completes the failover in less than 5 seconds. ([8], 2019)

- **Faster Failover**  
MongoDB can detect errors natively, in most cases, automatically choosing a new primary node in less than five seconds. Hence, the Applications will continue to work when replacing the faulty node.
- **Data Guarantees**  
Capable of continuing to work in case of a significant database failure, in return for the risk of displaying outdated data

MySQL provides High Performance, flexibility, reliable data protection, high availability with smaller databases and datasets fro business sectors. Moreover, It achieves performance through various dimensions such as Query optimization and Storage Engine. This is a safer alternative for users with a low data volume and seeks a more general solution since it can not cope with massive and unstructured data volumes. In facebook MySQL performance, MySQL uses InnoDB table engine with more than a thousand documents, which is very sluggish. (MDawodi [14], 2019) MySQL can transfer data to another node, failover between nodes is a complicated, manual method that increases downtime of the application as well as does not support flexible performance guarantees, reducing the possibilities developers have to ensure that their applications are accessible even if a number of database nodes are down. ([8], 2019 , [13])

## 4.2 Consistency and Transaction Management

MongoDB transactions provide ACID guarantees accuracy, and a single snapshot of the data is used for the duration of the transaction. Once a transaction starts with a snapshot at cluster time, no subsequent writings outside the context of that transaction will be seen in the transaction after the cluster time. However, MongoDB transactions guarantee that data will be read and written in the failed environment. Once a transaction begins, its snapshot view of the data will be preserved until either commit or abort. When a transaction commits, all data changes made in the transaction are saved, and the transaction is visible out of the side.(A. Kamsky[15],2019)

On the other Hand,MySQL Relational Database provides efficient consistency and transaction management than NoSQL databases. Moreover, the consistency levels for flexibility for both DBAs, who can use them to set up their infrastructure, and for developers who can use the degree of consistency that better fits the needs of their applications as compared to MongoDB not support consistency levels.In MySQL, Synchronization of transactions across a group to be in a read operation or write operation at the time. When data is synchronized at the time of reading, the current client session is waiting until a point is reached, which is the point in time where all previous update transactions have been performed before the execution can begin. However, only this session is affected by this method the all other simultaneous data operations are not affected.[16],2019

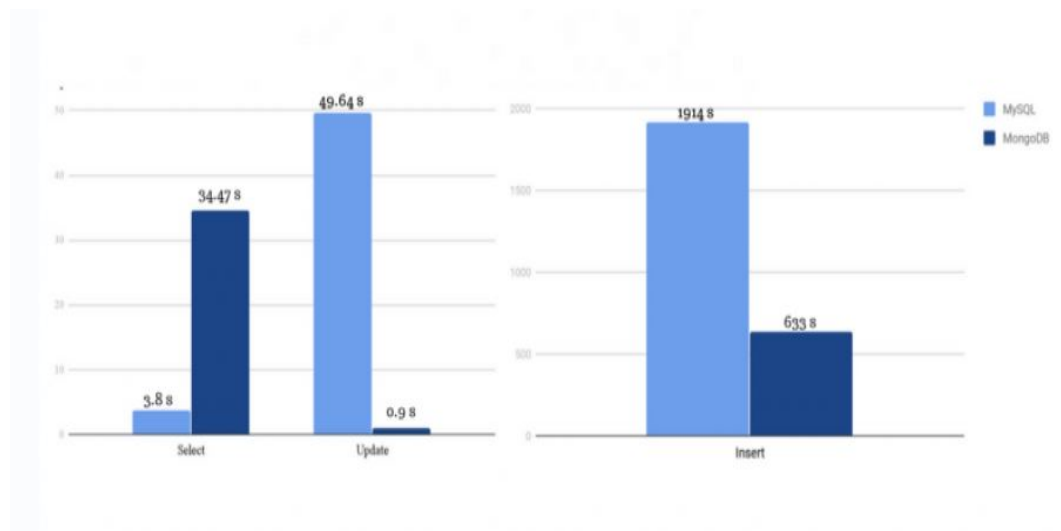


Figure 6: Query Optimisation performance on MongoDB and MySQL [13]

## 5 Literature Review: Critical Analysis of Related works

The (Patil, Mayur and Hanni[5],2017) presents an innovative qualitative evaluation of the performance of MySQL and MongoDB on the bases of insertion and retrieval operations to show a mechanism of auto-load balance queries. The experiment aimed to focus on the web application by describing the sharding feature of MongoDB positive benefits for the business applications as compared to MySQL. Moreover, the author considered two different datasets (MongoDB and MySQL) for analysis of web application as well as tools like Robomongo for MongoDB and PhpMyAdmin for MySql insertion of data and evaluation of load time. Overall final results proved for loading data dynamically. MySQL takes more time as compared to MongoDB. The results are accurate and well presented by the author as well as forgot to make a conclusion and future scope.



In (Deari, Raif and Zenuni[17],2018) describes the analysis and assessment of data storage and management by considering Relational MySQL and document-based MongoDB on the bases of its features and limitations. The significant purpose is to compare and contrast the key elements of each database type and to try to illustrate situations where NoSQL is a feasible alternative to adopt in business. Furthermore, the author presents comparison, including data organisation, transaction support, query language and security aspects. For the evaluation of performance, the author used CRUD Operations(create, read, update, delete) and simple data scheme as well as tested on the Ubuntu operating system. The experimental results show by considering CRUD operations, MongoDB shows better performance, mainly when working with extensive data and getting access to it at the same time, whereas MySql performs slow. the limitations of this research author haven't clearly mentioned about MySql

	<b>MySQL</b>	<b>MongoDB</b>
<i>Authentication</i>	<i>Username and password SCRAM x.509 Certificate Authentication LDAP proxy authentication Kerberos authentication</i>	<i>Username and password SCRAM (pluggable) x.509 Certificate Authentication(pluggable) LDAP proxy authentication(pluggable) Kerberos authentication (pluggable)</i>
<i>Authorization</i>	<i>Grant/Revoke Roles and Privileges (read, write) on different levels (database, collection, ...)</i>	<i>Grant/Revoke Roles and Privileges (read, write) on different levels (database, tables, records, fields, ...)</i>
<i>Transport</i>	<i>TLS/SSL</i>	<i>TLS/SSL</i>

Figure 7: Detailed comparison of security features in MongoDB and MySql [17]

The (Ongo, Gregorius [3],2018) introduced a novel social media website application for comparing the performance of RDBMS SQL and present a hybrid model of RDBSM MySQL and NoSQL MongoDB. The website has features like user-chatting and profile browsing. The aim to focus of this experiment was simple social networking website is selected as a test case to demonstrate only MySQL and MySQL and MongoDB hybrid database models success in handling mixed structured and unstructured data, commonly found in web applications. Furthermore, for the assessment author considered two different websites, one for MySQL and second for Hybrid database(Mysql and MongoDB) with tools like HeidiSQL for Mysql and RoboMongo for MongoDB using CodeIgniter framework. Additionally, Performance evaluation on bases of Read and write operations, Database based on User profiles and Disk usage, RAM and CPU. Finally, analyse results show the Hybrid Database MySQL and MongoDB faster than MySQL. The limitations of this experiment are author not review framework in detail.

The (Fraczek, Konrad and Plechawska-Wojcik[1],2019) describes the evaluation of the performance on the basis of storage architectures, principles and differences, and manipulation operations(insert, update, select ) by considering different relational databases such as Oracle, MySQL, and MS SQL as well as non-relational oriented databases like Mongo, Redis, GraphQL and Cassandra. Moreover, for the experiment purpose author selects Slovakia trains database. The performance valuation can be carried out using Slovakia trains website passengers database. Moreover, the results analysed and compared by showing query execution time on different relational and non-relational databases. Overall, results show the non-relational databases more effective than relational databases. The drawbacks of this experiment are the author not analyse accurate results.

The (Shetty, Sirish and KC [4],2019) presents the performance analysis and evaluation of

structured Oracle and an unstructured MongoDB on the bases of queries such as insert, update, delete and select. The aimed of this experiment is to provide solutions to the loss of information. Moreover, the author used an employee database for evaluation as well as two different tables in Oracle and singe in MongoDB for performance evaluation. The results analysed and compared using the execution time of queries. Finally, Non-relational databases perform better than relational databases, like MongoDB, only for addition, deletion and selection, while Oracle performs better for the updating process.

In (Gizem Kiraz [2],2017) introduced the novel performance analysis and evaluation of Relational MySQL and non-relational MongoDB on the bases of growing and storing a large amount of IoT data. The author aims to focus of this experiment is the utilization of IoT data by comparing MongoDB and MySQL. Moreover, The author describes IoT platform architecture where sensor generates data and servers stores and process them. The performance evaluation on ubuntu completed on the bases of reading and writing operations and executions time for storing data in IoT. Overall, results show In terms of writing and reading operations in the IoT environment, MongoDB has greater efficiency and throughput than MYSQL.

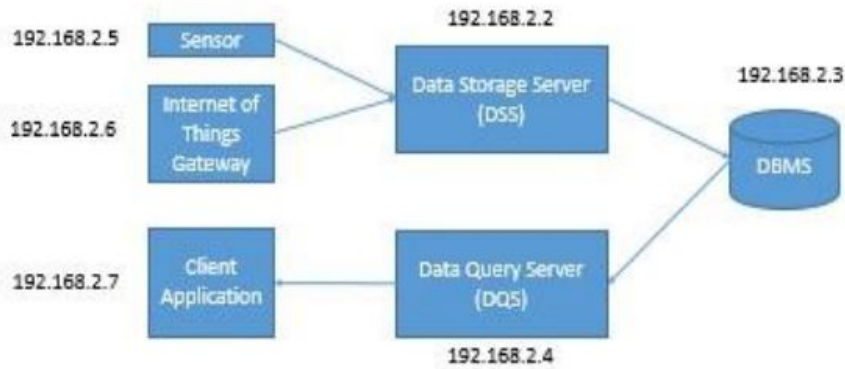


Figure 8: Proposed architecture of IoT Data Storage [2]

## 5.1 Summary of related works

According to all critical analysis and kinds of literature, We aim to focus on novel performance evaluation of Relational MySQL and Document-based on MongoDB using YCSB benchmarking tool and testharness workloads. The summary of the literature review includes objectives, architectures, results and analysis proposed by previous authors. Moreover, Previous authors focused on to evaluate performance on the bases of

- Auto-load balance queries
- Time analysis of CRUD Operations (Insert, Update, delete and select)
- Read and write operations, Database based on User profiles and Disk usage, RAM and CPU.
- Storage architectures, principles and differences, and manipulation operations(insert, update, select)
- Sensor data from IoT devices

In this research theme of performance comparison, We are planning to use YCSB benchmarking tool for database efficiency and calculate results on the bases of different workloads like read, write, update and throughput and compare time analysis between MySql and MongoDB.

## 6 Performance Test Plan for MySQL(Relational) and MongoDB(NoSQL)

This experiment tends and aims to focus on performance evaluation of MYSQL(Relational) and MongoDB(NoSQL) databases on the basis of read and write operations, latency and throughput as well as CRUD operations such as insert and update using rich Yahoo! Cloud Serving Benchmark (YCSB) evaluation tool and show accurate test results with analysis and comparison through visualization of graphs. The final test results will show the accurate performance analysis of two databases.

### 1. Host System Configuration

- Processor: Intel (R) Core(TM) i5-8250U CPU 3.4 GHz
- Specs: 8GB Memory, 1 TB HDD + 256GB SSD
- OS: Windows 10 Home Single Language
- System Type: 64-bit OS, x-64 based processor

### 2. Cloud-based Guest System Configuration

- Cloud: Openstack
- Instance Name: x18182852\_DSMPProj
- Flavour Name: m1.medium, (Floating ID: 3)
- Image Name: Ubuntu Bionic 18.04
- RAM: 4GB
- Specs: VCPUs: 2 VCPU and 40 GB Disk
- IP Address: 192.168.112.97, (Floating: 87.44.4.143)

### 3. Database Configurations

#### (a) MySQL (Relational)

- Server Version: 5.7.29
- Database: BenchTest
- Table: usertable

#### (b) MongoDB (NoSQL)

- shell version: 3.2.10
- Database: YCSB 0.001GB
- Collection table: usertable

### 4. Benchmarking tool Specs with Testharness

- Tool: Yahoo! Cloud Serving Benchmark
- Version: ycsb-0.14.0
- Testharness configuration: Bash script (runtest.sh) with YCSB core workloads

### 5. Workload Specs

- Workload A: Metadata environment with Mix of read Or write operations with ratio 50-50%.
- Workload D: Metadata environment with recently inserted records. Example application includes Changes to the user status as well as people who want to read the latest records.

- Workload E: Metadata environment with queries based on Short Range instead of individual records.
- Opcounts Load: We consider four different opcounts
  - (a) 70000
  - (b) 1,70,000
  - (c) 1,45,000
  - (d) 2,67,000

NCIRL provided testharness script uploaded on google drive, set all full access permissions and downloaded in Ubuntu instance by using, ***wget -no-check-certificate 'https://docs.google.com/uc?export=download&id=1PLAa46G4MyGOTc8KkkQztyTVZkN70RVc' -O testharness.tgz***

## 6.1 Flow Chart Test Plan

The flow chart explains the step by step and detailed procedure of this experiment from beginning to end. This can help the proper execution of the experiment.

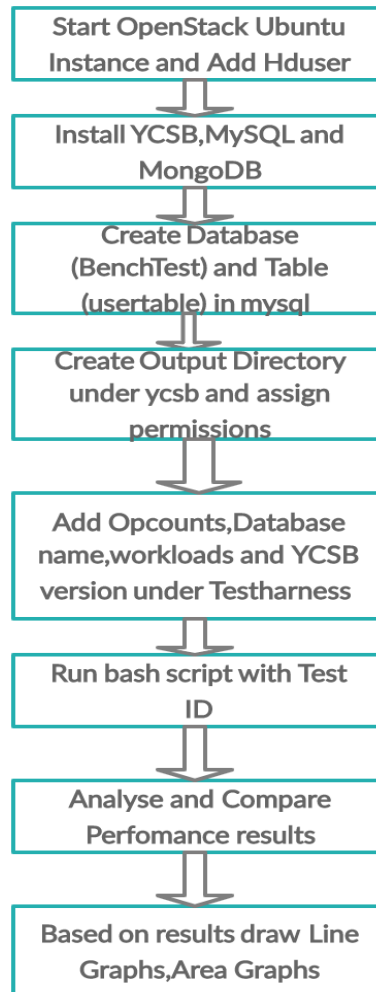


Figure 9: Process Flow Chart

## 7 Evaluation and Results

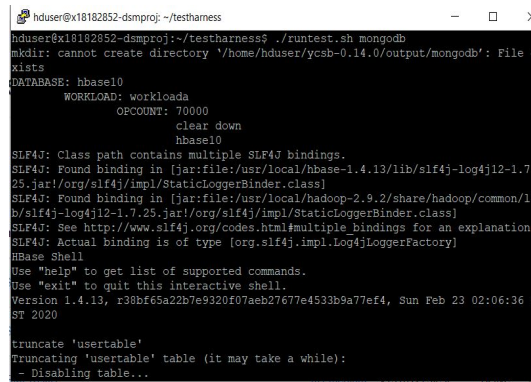
In this data accessibility comparison theme, we evaluated the performance of two distinct databases such as Relational MySQL and Non-Relational MongoDB by considering four opcodes such as 70000,170000,145000 and 267000 as well as evaluation can be complete by using YCSB tool. The following are three different workloads added on Workload as well as opcodes file under testharness and applied on databases.

- Workload A: indicates Heavy workload with 50-50% ratio of Read or write operations.
- Workload D: indicates latest Inserted records.
- Workload E: indicates Short Range records.

We used testharness bash script to analyse and compare performance . In that script we mentioned latest version of YCSB (0.14) as well as under ycsb directory we created *Output* directory where all records inserted through script. We provided permissions to output directory by using following command

```
sudo chown -R hduser:hadoopgroup /home/hduser/ycsb-0.14.0/output
```

Finally, run script by using *./runscript.sh* with two different test ID *mysql* for MySQL records and *mongodb* for MongoDB records.



```
hduser@x18182852-dsmproj: ~/testharness
hduser@x18182852-dsmproj:~/testharness$ ./runtest.sh mongodb
mkdir: cannot create directory '/home/hduser/ycsb-0.14.0/output/mongodb': File exists
DATABASE: hbase10
WORKLOAD: workloada
OPCOUNT: 70000
clear down
hbase10
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-1.4.13/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.9.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.13, r38bf65a22b7e9320f07aeb27677e4533b9a77ef4, Sun Feb 23 02:06:36 PST 2020
truncate 'usertable'
Truncating 'usertable' table (it may take a while):
- Disabling table...
```

Figure 10: Run testharness bash script on Openstack Ubuntu Instance

- The above figure shows the running testharness script with opcount 70000 and workload A as well as it will end on last opcount 267000 and Workload E.
- The individual opcounts as well as workload records are stored under directories under Output, i.e. Mysql and MongoDB

### 7.1 Workload A: Heavy workload

In this workload A, we considered 70000,170000,145000 and 267000 opcodes and evaluate the performance of Mysql and MongoDB

### 7.1.1 Opcounts evaluation with overall Throughput(ops/sec)

In that, we show the four different opcounts and Overall Throughput (ops/sec) of both databases as well as compare performance.

Workload	Database	Opcounts	Throughput(Ops/Sec) Overall
Workload A	MySQL	70000	2197.526213348402
Workload A	MySQL	170000	2401.7744875037088
Workload A	MySQL	145000	2275.3664124533517
Workload A	MySQL	267000	2255.048521549649
Workload A	MongoDB	70000	2078.014605474084
Workload A	MongoDB	170000	2410.73201168496
Workload A	MongoDB	145000	2390.588897764132
Workload A	MongoDB	267000	2789.7716133577824

Table 1: Opcounts VS Overall Throughput(Ops/Sec)

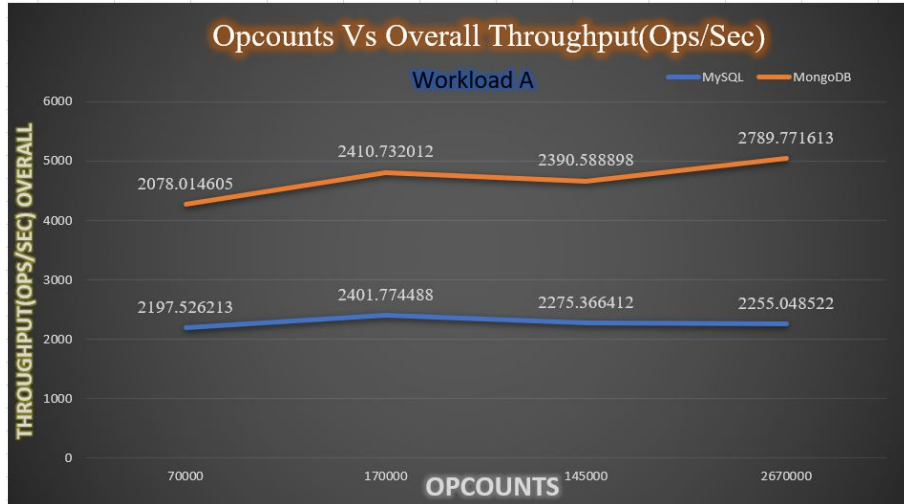


Figure 11: Opcounts VS Overall Throughput(Ops/Sec)

- The figure 11 depicts Area Chart with accurate analysis and comparison of two databases in terms of Throughput including 50 % reads and 50% Updates as well as opcounts in Workload A.
- Moreover, the Throughput (2390.588) of MongoDB is an upward trend with 267000 op-count as compared to Throughput (2401.77) of MySQL show slightly upward with 170000 opcount after that remain stable until the last opcount.
- Overall, the Throughput performance of MongoDB is higher than MySQL.

### 7.1.2 Read Operations evaluation with Average Latency( $\mu s$ )

This section contains accurate analysis and comparison of Overall Average latency as per READ operations.

Workload	Database	READ Operations	Average Latency( $\mu s$ )
Workload A	MySQL	34904	323.3809592023837
Workload A	MySQL	84814	528.7641545027943
Workload A	MySQL	72624	520.2296348314607
Workload A	MySQL	133639	649.3356280726434
Workload A	MongoDB	35095	335.63453483402196
Workload A	MongoDB	84697	535.6050863666954
Workload A	MongoDB	72406	519.0857801839627
Workload A	MongoDB	133762	633.093786943975

Table 2: READ Operations Vs Overall Average Latency( $\mu s$ )

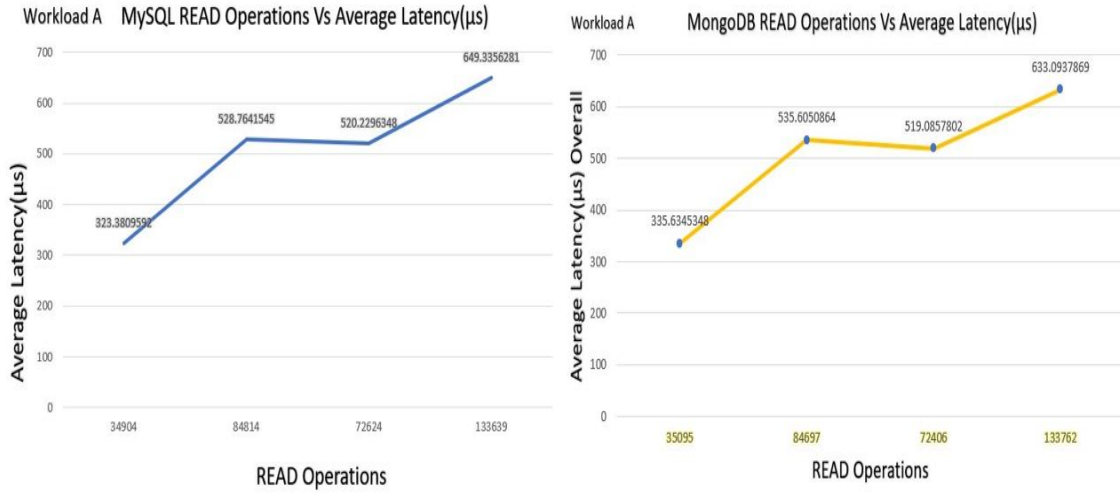


Figure 12: READ Operations Vs Overall Average Latency ( $\mu s$ )

- The above Line Graph figure 12 shows the accurate performance of READ Records with Average Latency of MySQL and MongoDB.
- As we can see, The MySQL READ Operation(133639) with Average latency 649.3356281 ( $\mu s$ ) is abruptly increasing as compare to MongoDB READ Operation(133762) with Avg. Latency 633.0937869 ( $\mu s$ ) is comparatively slow.
- Average latency is referred to the amount of time taken to satisfy an I/O request, and the lower the average latency is, the better for the performance.
- Overall, As per above visual representation the MongoDB is faster in Read Operations than MySQL.

### 7.1.3 INSERT Operations evaluation with Average Latency

This section contains Overall Average latency( $\mu s$ ) as per INSERT operations. Moreover, The Insert operations contains opcounts and the overall latency of both databases.



Workload	Database	INSERT Operations	Average Latency( $\mu s$ )
Workload A	MySQL	70000	431.37408571428574
Workload A	MySQL	170000	404.28327058823527
Workload A	MySQL	145000	419.72195862068963
Workload A	MySQL	267000	434.20783895131086
Workload A	MongoDB	70000	455.7227857142857
Workload A	MongoDB	170000	402.3977823529412
Workload A	MongoDB	145000	412.2106620689655
Workload A	MongoDB	267000	405.3854943820225

Table 3: INSERT Operations Vs Overall Average Latency( $\mu s$ )

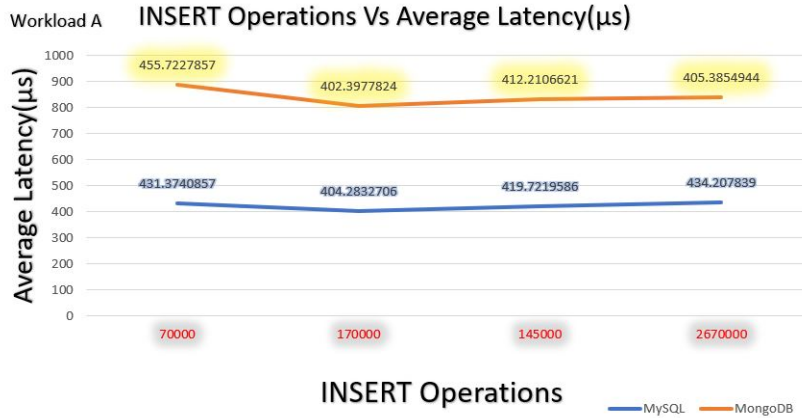


Figure 13: INSERT Operations Vs Overall Average Latency( $\mu s$ )

- The above Line Graph figure 13 shows the accurate performance of INSERT Records with Average Latency of MySQL and MongoDB.
- As we can see, The MySQL INSERT Operation 2670000 with Average latency 434.2077839 ( $\mu s$ ) is abruptly increasing as compare to MongoDB READ Operation 2670000 with Avg. Latency 405.3854944 ( $\mu s$ ) is significantly drop.
- In INSERT Operations, Average latency is referred to the amount of time taken to satisfy an I/O request, and the lower the average latency is, the better for the performance.

#### 7.1.4 UPDATE Operations evaluation with Average Latency( $\mu s$ )

This section contains Overall Average latency ( $\mu s$ ) metadata as per UPDATE operations.



Workload	Database	UPDATE Operations	Average Latency( $\mu s$ )
Workload A	MySQL	35096	405.0670447
Workload A	MySQL	85816	424.8743455
Workload A	MySQL	72376	420.6884713
Workload A	MySQL	13361	436.0418638
Workload A	MongoDB	34905	411.5474001
Workload A	MongoDB	85303	409.8125857
Workload A	MongoDB	72594	419.4198556
Workload A	MongoDB	133238	426.8811901

Table 4: UPDATE Operations Vs Overall Average Latency( $\mu s$ )

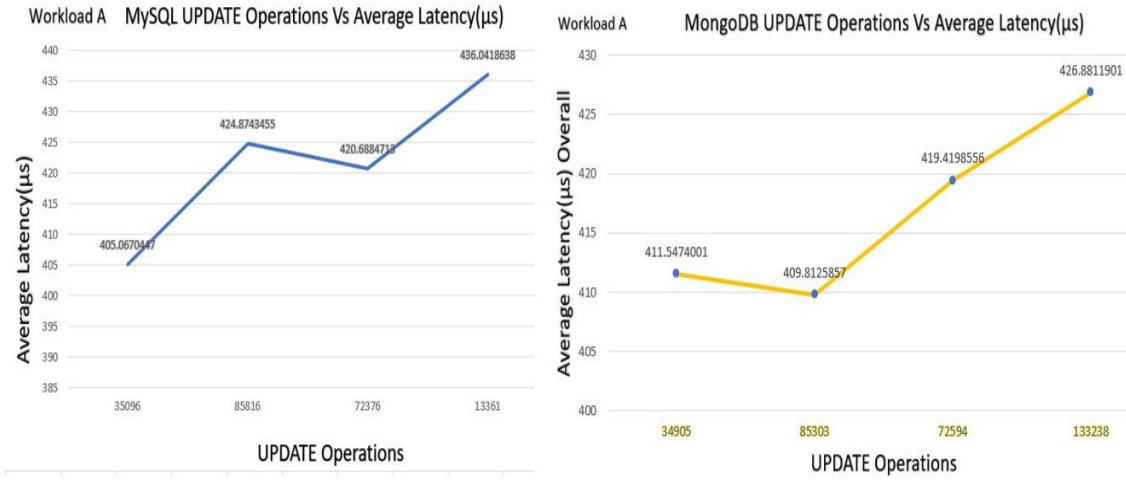


Figure 14: UPDATE Operations Vs Overall Average Latency( $\mu s$ )

- The above Line Graph figure 14 Depicts the accurate performance of UPDATE Records with Average Latency of MySQL and MongoDB.
- As we can see, The MySQL UPDATE Operation 13361 with Average latency 434.0418638 ( $\mu s$ ) is abruptly increased as well as MongoDB UPDATE Operation 133238 with Avg. Latency 426.8811901 ( $\mu s$ ) is close to MySQL
- In this UPDATE operation, MongoDB avg. latency is slow that means performance higher as compare rise in MySQL Average Latency.

## 7.2 Workload D: latest Inserted records

By consideration of testharness script, we executed workload D contains Throughput, Read Operations as well as information about the latest Inserted records. Moreover, The following Accurate Analysis shows the proper evaluation of performance.

### 7.2.1 Opcounts evaluation with overall Throughput(ops/sec)

In that, we show the four different opcounts and Overall Throughput (ops/sec) of both databases as well as compare performance.

Workload	Database	Opcounts	Throughput(Ops/Sec) Overall
Workload A	MySQL	70000	2188.525871502267
Workload A	MySQL	170000	2353.2343129247934
Workload A	MySQL	145000	2341.731266149871
Workload A	MySQL	267000	2292.1996653165296
Workload A	MongoDB	70000	2136.947827945172
Workload A	MongoDB	170000	2363.934699780293
Workload A	MongoDB	145000	2231.4901737484422
Workload A	MongoDB	267000	2283.5931954054445

Table 5: Opcounts VS Overall Throughput(Ops/Sec)

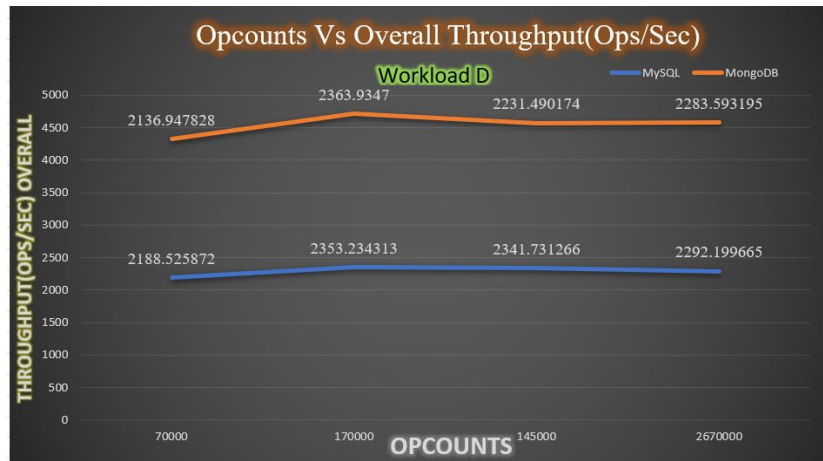


Figure 15: Opcounts VS Overall Throughput(Ops/Sec)

- The Figure 15 shows the accurate performance evaluation of the overall Throughput (ops/sec) as per Opcounts of MongoDB and MySQL databases in Workload D.
- The graphical representation depicts both databases throughput remain stable.
- Furthermore, MongoDB throughput is slightly hit a high of MySQL Throughput with the same opcount 267k.
- Additionally, Both databases throughput performance show a minimum rise from opcount 17k.

### 7.2.2 Latest INSERT Operations evaluation with Average Latency( $\mu s$ )

This section contains Overall Average latency as per Latest INSERT operations.

Workload	Database	Latest INSERT Operations	Average Latency( $\mu s$ )
Workload A	MySQL	70000	430.6788285714286
Workload A	MySQL	170000	412.14645294117645
Workload A	MySQL	145000	413.2812827586207
Workload A	MySQL	267000	426.2298876404494
Workload A	MongoDB	70000	438.973857142857123
Workload A	MongoDB	170000	406.5654470588235
Workload A	MongoDB	145000	426.406475862069
Workload A	MongoDB	267000	427.96294756554306

Table 6: latest INSERT Operations Vs Overall Average Latency( $\mu s$ )

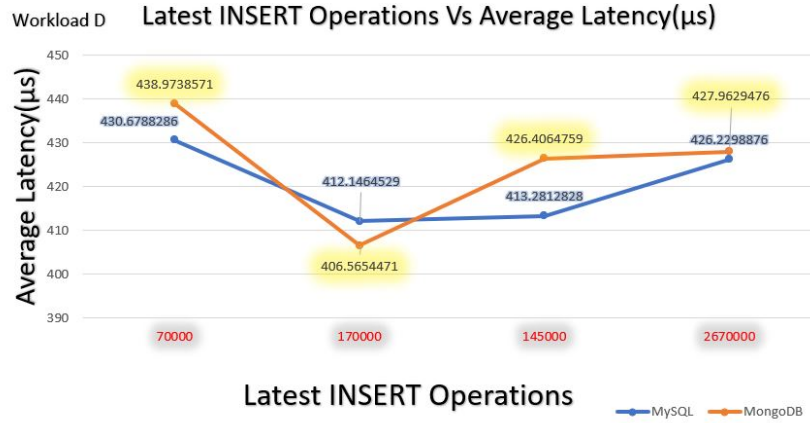


Figure 16: latest INSERT Operations Vs Overall Average Latency( $\mu s$ )

- The above line graph analysis figure 16 shows a detailed view of latest Insert records as per average latency in terms of opcounts of both databases in workload D.
- The graphical analysis clearly show only single point difference (Average Latency) in both databases. The MongoDB shows a downward trend in opcount 17k.
- Besides, MongoDB shows the worst performance as compare MySQL show less average latency.
- Finally, MySQL performs well in updated INSERT records compare to MongoDB

### 7.2.3 Read Operations evaluation with Average Latency( $\mu s$ )

This section contains Overall Average latency as per READ operations. Moreover, The READ operations contains the overall Average Latency( $\mu s$ ) of both databases.

Workload	Database	READ Operations	Average Latency( $\mu s$ )
Workload A	MySQL	66547	251.2059376739847
Workload A	MySQL	161697	390.6666171914136
Workload A	MySQL	137858	414.88042768646
Workload A	MySQL	253545	438.11936737068373
Workload A	MongoDB	66605	242.86687185646724
Workload A	MongoDB	161958	557.5278963848563
Workload A	MongoDB	137775	419.2654908365088
Workload A	MongoDB	2543433	481.1051283771253

Table 7: READ Operations Vs Overall Average Latency( $\mu s$ )

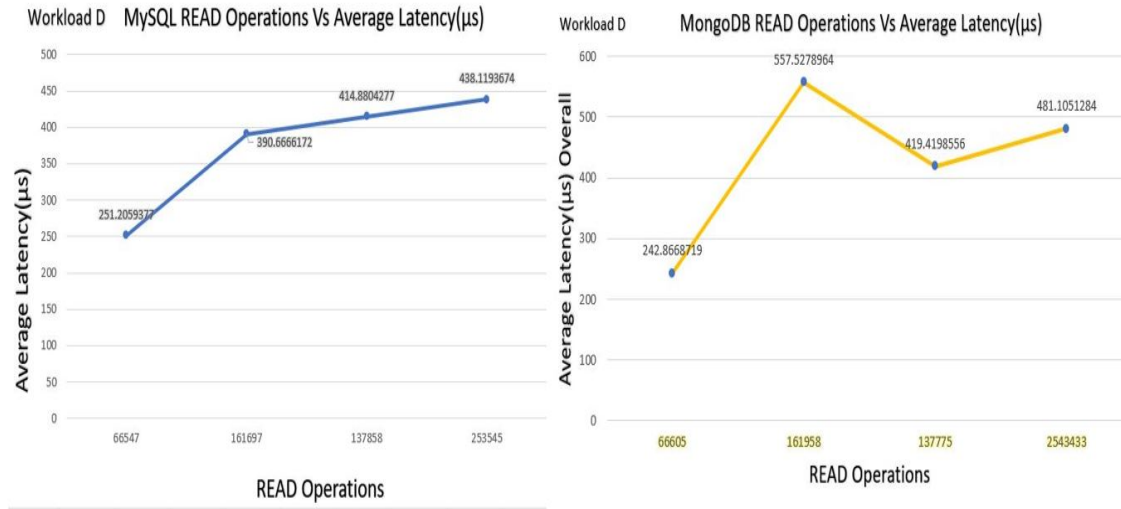


Figure 17: READ Operations Vs Overall Average Latency( $\mu s$ )

- The above visualisation figure 17 presents a detailed evaluation of the performance of Overall Average latency in terms of Read Operations of MongoDB and MySQL in Workload D.
- Both databases average latency show an increasing trend from beginning from different average latency's 251.2059372 ( $\mu s$ ) and 242.8668719 ( $\mu s$ ).
- Meanwhile, the MongoDB dramatically fall in 419.4198556 with read operation 137775 as well as rise up to 481.1051284 whereas MySQL is continuously increasing at peak point 438.1193674 with read operation 253545.

### 7.3 Workload E: Short Range records

This section includes analysis and comparison Throughput as well as short Range SCAN Operations by taking consideration of testharness script.

### 7.3.1 Opcounts comparison with overall Throughput(ops/sec)

In that, we show the four different opcounts and Overall Throughput (ops/sec) of both databases as well as compare performance.

Workload	Database	Opcounts	Throughput(Ops/Sec) Overall
Workload A	MySQL	70000	2045.467827713167
Workload A	MySQL	170000	2244.994981775923
Workload A	MySQL	145000	2311.4200089268634
Workload A	MySQL	267000	2223.4621053771143
Workload A	MongoDB	70000	2024.5842371655822
Workload A	MongoDB	170000	2359.5708357044705
Workload A	MongoDB	145000	2275.116501655343
Workload A	MongoDB	267000	2300.5738510055317

Table 8: Opcounts VS Overall Throughput(Ops/Sec)

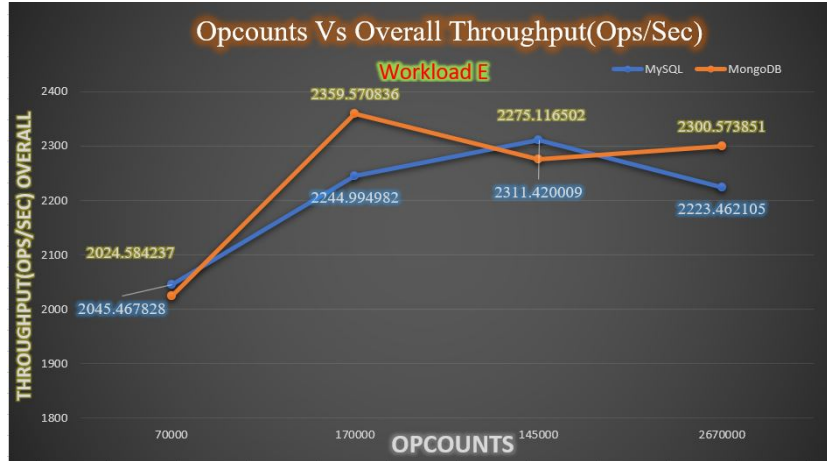


Figure 18: Opcounts VS Overall Throughput(Ops/Sec)

- The Figure 18 shows an accurate evaluation of the performance of Overall Throughput(ops/sec) with for opcounts of MongoDB and MySQL in Workload E.
- Both databases average latency show an increasing trend from beginning from opcount 70k. Subsequently, MongoDB falls in opcount 14.5k with throughput 2275.116502 ( $\mu s$ ) whereas MySQL Falls in opcount 26.7k with performance 2300.573851 ( $\mu s$ ) .
- In the end, MongoDB rises to 2300.575 as compared MySQL show worst.

### 7.3.2 SCAN Operations comparison with Average Latency( $\mu s$ )

This section contains Overall Average latency as per Short Range SCAN operations.

Workload	Database	SCAN Operations	Average Latency( $\mu s$ )
Workload A	MySQL	66500	1556.3206015037595
Workload A	MySQL	161615	1788.9140735699038
Workload A	MySQL	137936	1854.0232064145691
Workload A	MySQL	253669	1816.8365862600476
Workload A	MongoDB	66564	1564.6680788414158
Workload A	MongoDB	161396	1754.3299957867605
Workload A	MongoDB	137654	1782.0035959725108
Workload A	MongoDB	253655	1790.7535786797027

Table 9: SCAN Operations Vs Overall Average Latency( $\mu s$ )

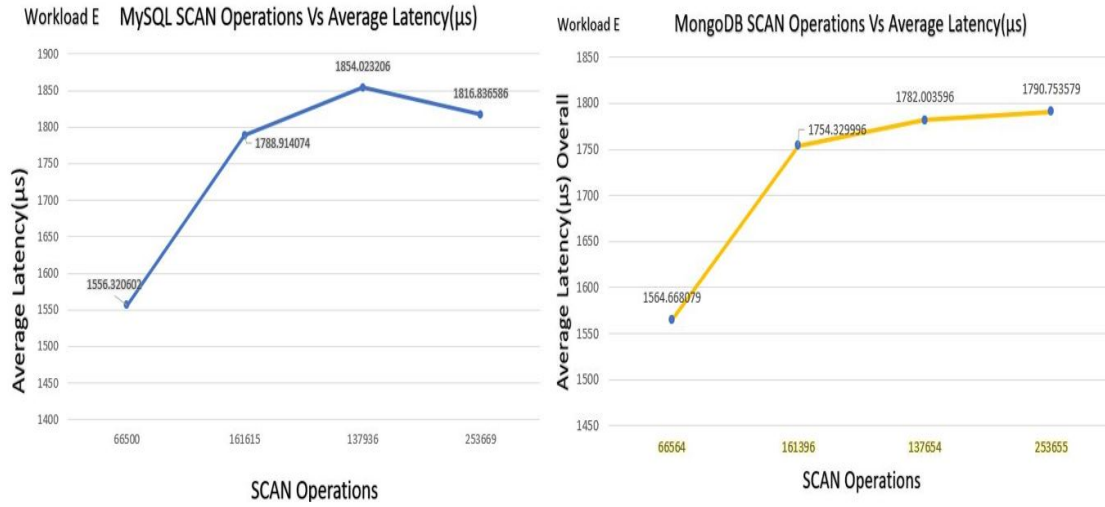


Figure 19: SCAN Operations Vs Overall Average Latency( $\mu s$ )

- The purpose of workload E is a present short-range of records referred to as SCAN operations instead of Individual records.
- Figure 19 shows Scan operations with an average latency of MongoDB and MySQL.
- The graphical representation illustrates a growing trend from the beginning SCAN operations 66k in both databases.
- Moreover, The MongoDB performance grows steadily at peak point 1790.753579 ( $\mu s$ ), whereas MySQL shows a downward direction to 1816.836586 ( $\mu s$ ) with scan operation 25k.

## 8 Results Discussion and Conclusion

This overall performance comparison theme aims to proposed solutions for proper data storage management in business sectors as well as Internet and telecommunications industries by considering an evaluation of highly available Relational MySQL and Non-Relational MongoDB Databases as these industries has lack of standards and mechanisms to handle large amount unstructured data coming from IoT sensors as well as database applications. Moreover, RDBMS MySQL helps to provide the solution for small databases storage management, whereas NoSQL

MongoDB not only provides solutions for data storage and management, It significantly gives automation for proper handling data in business sectors.

We can see the below figure 20 shows evaluations of Overall Runtime in milliseconds and Operational Counts between workload A, workload D and Workload E in MySQL and MongoDB. The overall graphical representation shows the MySQL takes more runtime for a load of various records as compared to MongoDB. The MongoDB perform faster than MySQL.

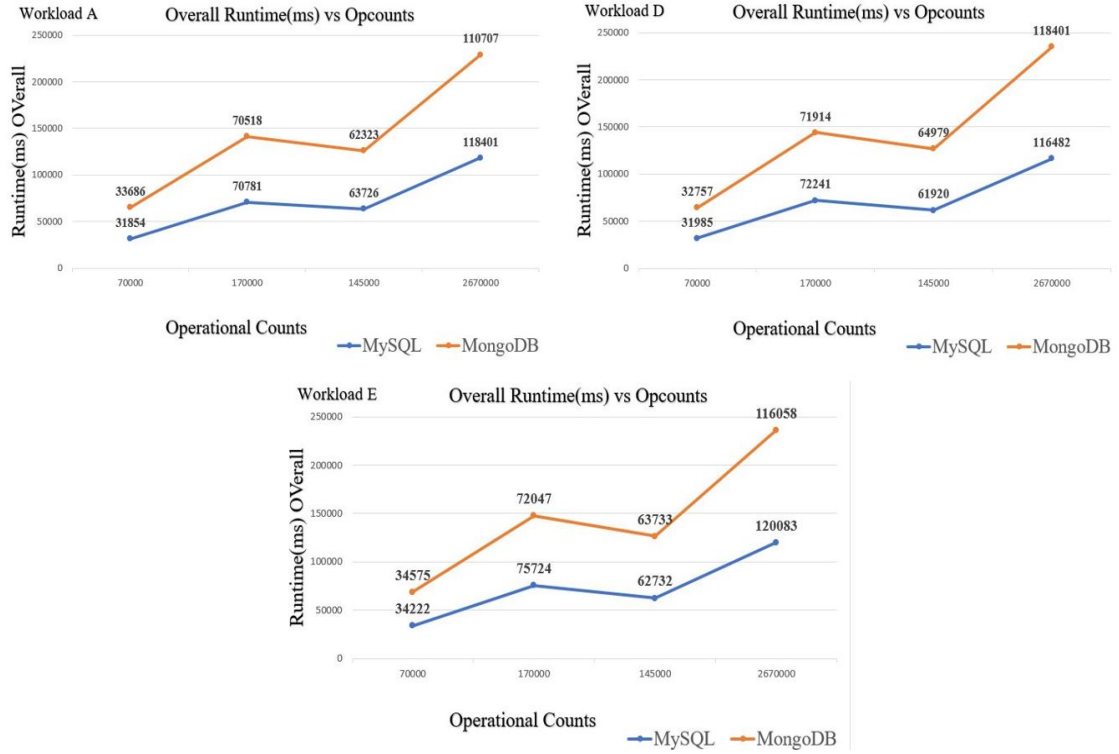


Figure 20: Overall Run-time(ms) between Workloads A,D and E

It is evident from the following Table 10 that, MongoDB takes lesser overall Runtime in milliseconds for different types of operations in Workload A, D and E as compared to MySQL takes more time for the Opcount 2.67k.

Workload	Database	Type of Records	Runtime(milliseconds)
Workload A	MySQL	READ,INSERT	149801
Workload A	MongoDB	READ,INSERT	145841
Workload D	MySQL	READ,INSERT	125592
Workload D	MongoDB	READ,INSERT	136588
Workload E	MySQL	INSERT,SCAN	475916
Workload E	MongoDB	INSERT,SCAN	470931

Table 10: Types of Records Vs Runtime for Opcount 2.67k

Some limitations also found in this experiment is both databases (MySQL and MongoDB) in data accessibility time, as well as data management still face latency issues that directly affect performance. Meanwhile, MongoDB performs better than MySQL in a cloud environment. This comparison experiment, we analyse the performance of these two bases using powerful YCSB tool and Testharness bash Script as well as consider three unique Workloads A, D and E with four Operational Counts 70000,170000,1450000 and 2670000. Moreover, We show graphical



analysis by taking considerations of Overall throughput, Average Latency and CRUD Operations (INSERT and UPDATE) as well a READ and SCAN operations. This experiment shows the MongoDB performs higher than MySQL. From all observations, the results and analysis illustrate the MongoDB is suitable for larger datasets and MySQL is a good fit for Small datasets. For future work, we aim to evaluate the performance in the federated cloud environment and analyse results.

## References

- [1] K. Fraczek and M. Plechawska-Wojcik, "Comparative analysis of relational and non-relational databases in the context of performance in web applications," pp. 153–164, 04 2017.
- [2] G. Kiraz, "Iot data storage : Relational non-relational database management systems performance comparison," 2017.
- [3] G. Ongo and G. Kusuma, "Hybrid database system of mysql and mongodb in web application development," pp. 1–9, 09 2018.
- [4] S. Shetty and A. KC, "Performance analysis of queries in rdbms vs nosql," 07 2019.
- [5] M. Patil, A. Hanni, C. Tejeshwar, and P. Patil, "A qualitative analysis of the performance of mongodb vs mysql database based on insertion and retrieval operations using a web/android application to explore load balancing — sharding in mongodb and its advantages," pp. 325–330, 02 2017.
- [6] D. Mahajan, C. Blakeney, and Z. Zong, "Improving the energy efficiency of relational and nosql databases via query optimizations," *Sustainable Computing: Informatics and Systems*, vol. 22, 03 2019.
- [7] D. Team, "9 new mongodb features - must learn to master in mongodb," Jan 2019.  
**URL:** <https://data-flair.training/blogs/mongodb-features/>.
- [8] "Mongodb vs. mysql - comparison differences."  
**URL:** <https://www.mongodb.com/compare/mongodb-mysql>.
- [9] "Mysql 8.0 reference manual :: 1.4 what is new in mysql 8.0."  
**URL:** <https://dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html>.
- [10] B. Jose and S. Abraham, "Exploring the merits of nosql: A study based on mongodb," in *2017 International Conference on Networks Advances in Computational Technologies (NetACT)*, pp. 266–271, 2017.
- [11] Lalit, "Mysql architecture and components," Feb 2019.  
**URL:** <https://lalitvc.wordpress.com/2016/11/03/mysql-architecture-and-components/>.
- [12] M. Keep, "Performance best practices: Mongodb data modeling and memory sizing: Mongodb blog," Jan 2020.  
**URL:** <https://www.mongodb.com/blog/post/performance-best-practices-mongodb-data-modeling-and-memory-sizing>.
- [13] "Mongodb vs mysql:a comparative study on databases," Apr 2020.  
**URL:** <https://www.simform.com/mongodb-vs-mysql-databases>.



- [14] M. Dawodi, M. H. Hedayati, J. A. Baktash, and A. L. Erfan, “Facebook mysql performance vs mysql performance,” in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0103–0109, 2019.
- [15] A. Kamsky, “Adapting tpc-c benchmark to measure performance of multi-document transactions in mongodb,” *Proc. VLDB Endow.*, vol. 12, p. 2254–2262, Aug. 2019.
- [16] “Mysql 8.0 reference manual.”  
**URL:**<https://dev.mysql.com/doc/refman/8.0/en/group-replication-understanding-consistency-guarantees.html>.
- [17] R. Deari, X. Zenuni, J. Ajdari, F. Ismaili, and B. Raufi, “Analysis and comparision of document-based databases with relational databases: Mongodb vs mysql,” pp. 1–4, 09 2018.
- [18] C. Győrödi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: Mongodb vs. mysql,” 06 2015.
- [19] P. Grover and R. Johari, *MVM: MySQL Versus MongoDB*, pp. 899–909. 03 2016.
- [20] S. Padhy and G. M. M. Kumaran, “A quantitative performance analysis between mongodb and oracle nosql,” in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 387–391, 2019.