Quantitative Performance Evaluation of Cloud-Based MySQL (Relational) Vs. MongoDB (NoSQL) Database with YCSB

Technical Report · April 2019

DOI: 10.13140/RG.2.2.29628.59528

Darshankumar Gorasiya
National College of Ireland

Data Storage and Management Project

on

Quantitative Performance Evaluation of Cloud-Based MySQL (Relational) Vs. MongoDB (NoSQL) Database with YCSB

Darshankumar Vinubhai Gorasiya
x18134751

MSc/PGDip Data Analytics – 2019/20

Submitted to: Prof. Iqbal Muhammad

Quantitative Performance Evaluation of Cloud-Based MySQL (Relational) Vs. MongoDB (NoSQL) Database with YCSB

Darshankumar Vinubhai Gorasiya
x18134751

April 22, 2019

Abstract

NoSQL was built to satisfy the need of Big Data applications. Modern application demands a database management system that can deal with large volumes of rapidly changing data with geo-distributed architecture supporting easy scale-out. However, there is a substantial number of business applications running on low-volume environments having structured and gradually changing information and data isn't growing exponentially. NoSQL (schema-less) structure does not hold the ACID properties (Atomicity, Consistency, Isolation, Durability) and loses the referential trustworthiness and consistency assurance of data which conventional SQL (relational) databases provide and it is vital to numerous mission-critical small-scale applications.

Migration of existing highly structured database to NoSQL for better performance has been trending among medium scale businesses however it is an expansive and complex process. This report replicates low-volume and high-volume application operations on NoSQL and Traditional SQL database with Yahoo! Cloud Serving Benchmark (YCSB) for analyzing quantitative performance differences between NoSQL document-oriented database MongoDB and relational database MySQL.

1 Introduction

The traditional databases like MySQL, Microsoft SQL Server, Oracle Database, PostgreSQL or IBM DB2 are still being utilised by large number of organisations and individuals for the purpose of holding relational data due to its ease of use and need for routine analysis, industry-wide support, SQL standards, reliability of transactions and large developer community support. The Digital revolution and increases in embedded devices generating data and growing applications of ML (AI) have caused a constant rise of distributed NoSQL (Not Only SQL) databases implementation demand which compromises consistency while provides availability and flexibility of scaling horizontally as mention in Huang Y. (2014).

The Stack Overflow Developer Survey StackExchange (2018), the worlds largest developer survey with over 100,000 respondents indicates that MySQL is the most famous database and still widely being used with the greater part of the respondents (58.7%) confirms utilizing MySQL where the same survey unveils that MongoDB is the most wanted database. There is growing demand for NoSQL like better performance and SQL like flexibility, consistency and strong transactional support in medium and high volume traditional databases application and large NoSQL commercial and open source databases vendors are increasingly working on with different design approach to address the urging. MongoDB is one of the vendor to provide support for multi-document ACID transactions in version 4.0 of its NoSQL database.

In the Big Data world major trade-off between all the different distributed database can be categorised based on CAP Theorem and its attributes are as follow,

1. Consistency -
   Data is consistent throughout when a transaction begins and when it closes and all the user should be able to see same information.

2. Availability -
   At least one copy of data should be always available to users whenever requested.

3. Partition Tolerance -
   Irrespective of deployed platform system performance should be tolerance to network partitions.

Any distributed database management system cannot achieve more than 2 attributes of CAP at a time. MySQL prioritises Consistency & Availability (CA) where loses on partition tolerance and MongoDB relays on consistency & partition (CP).

In this study we will compare performance of MySQL (Consistency & Availability - CA) with MongoDB (consistency & partition - CP). Yahoo! Cloud Serving Benchmark (YCSB) automated workloads used for quantitative comparison with large and small data volume.

2 Key Characteristics of The Data Storage Management Systems

2.1 NoSQL: MongoDB

1. Aggregation Framework (MapReduce) -
   Aggregation operations in MongoDB process data records and returns computed results. It is built on the idea of information handling pipelines where documents are given as input and gets aggregated data as result. MongoDB also provides efficient map-reduce operation support for aggregation for efficient data summarization.

2. Horizontal Scalability (Sharding) -
   MongoDB bolsters horizontal scaling through Sharding, disseminating data over multiple machines which in turn helps to facilitate high throughput on high volume datasets. Sharding permits adding additional instances to expand the capacity of the database as and when required.

3. Capped Collections -
   MongoDB supports the creation of broadly utilized capped collections where we can confine the collection's maximum size. This collection follows the kind of circular queue concept where once collection size reaches max dispensed size limit, it starts making space for new data by supplanting the oldest document in the collection.

4. Support for Multiple Storage Engines -
The storage engine in MongoDB guarantees how data is being put away in-memory and on disk. MongoDB provides freedom of picking different storage engines as per the application requirement as any single engine might not be the most efficient solution for all sort of uses.

5. Documents Indexing -
Indexes stores the collections datasets in a structure that is easy to traverse. MongoDB supports single, compound, multikey, geospatial and hashed indexes on data. It helps discover documents matching query requirements without executing the entire collection scan.

6. Master Slave Replication (Replica Sets) -
MongoDB underpins replication by making various copies of data over different locations to protect data when the database suffers any loss. Concept replica set in MongoDB gives consistency for all read operations from the primary node as at least one redundant copy will be accessible in secondary node copies.

7. Load Balancing (Read-Write) -
MongoDB scalable architecture is upheld by the load balancer to dynamically balance the operation load of queries and manages balanced documents spread over multiple nodes for reads and write tasks.

8. High Performance -
MongoDB uses the binary object BSON for storing data in documents. Binary improves the overall performance of MongoDB with support of replica sets, Sharding and load balancer.

9. Schema-less Database -
MongoDB is Schema-less database where different set of data and fields with different types can be dynamically stored and it makes MongoDB schema migrations simple in large organizations. Any sorts of data can be adjusted in the database at ease.

2.2 Relational: MySQL

1. Robust Transactional Support (ACID) -
MySQL holds amongst the most robust transactional database engines. It bolsters row-level locking, multi-version transaction and server-enforced referential trustworthiness to achieve full ACID (atomic, consistent, isolated, durable transaction) qualities.

2. Replication (Built-in) -
MySQL is one of few databases who guarantees nonstop uptime as a profoundly available solution. It can be implemented by configuring cluster specialized servers and master-slave replication architecture. There are also additionally third-party vendors in the market giving out of box availability solution for MySQL.

3. Highly Secure System -
MySQL offers an exceedingly secure security framework that reinforces both clients

based and host-based verification. It likewise offers support of SSH/SSL for secure connections to the database with object-level privilege framework to accomplish high data security.

4. SQL Compatibility -
   MySQL is one of the most established and popular open source relational database systems. It supports widely accepted ANSI standard SQL where large developer community utilizing it. MySQL additionally gives a range of MySQL connectors and drivers like ODBC and JDBC to support seamless external application integration and connectivity.

5. Fully relational DBMS -
   MySQL is a Relational database management system and it isolates or groups related data with help of logical database structure like a table, columns and views and relates them by linking tables with each other in the database while physical data is composed into a disk as files.

6. Platform Independence -
   MySQL is platform independence and can be used on the number of operating systems without worrying about compatibility issues caused due to the difference in operating system.

7. Multi-Threaded SQL Server -
   MySQL server supports multi threaded task management which allows users process to be distributed with multiple small threads on different processing units of machine.

8. Supports Geographic Information System (GIS) functions -
   MySQL provides functions to process GIS (geographic information system) data objects. Objects having spatial attributes like geographical coordinates. Any data having more than one dimension can be treated as a spatial entity

3 Database Architectures

3.1 MongoDB

MongoDB is constructed based on the design philosophy to utilize capabilities of relational databases with the addition of best of innovative NoSQL technologies. Its unique performance capabilities make to the choice of many organizations as a storage solution for Big Data applications and advanced real-time analytics MongoDB (2019).

1. Storage Engines -
   MongoDB storage engine enables utilization of the best of various storage engines. There are four predominantly supported storage engines accessible and all four can work in a single replica set MongoDB. It comes with default WiredTiger storage engine that furnishes all-around performance with local compression support. Encrypted storage engine ensures sensitive data protection without influencing the performance and the in-memory storage engine (ISE) provides wide support for real-time analytics applications.

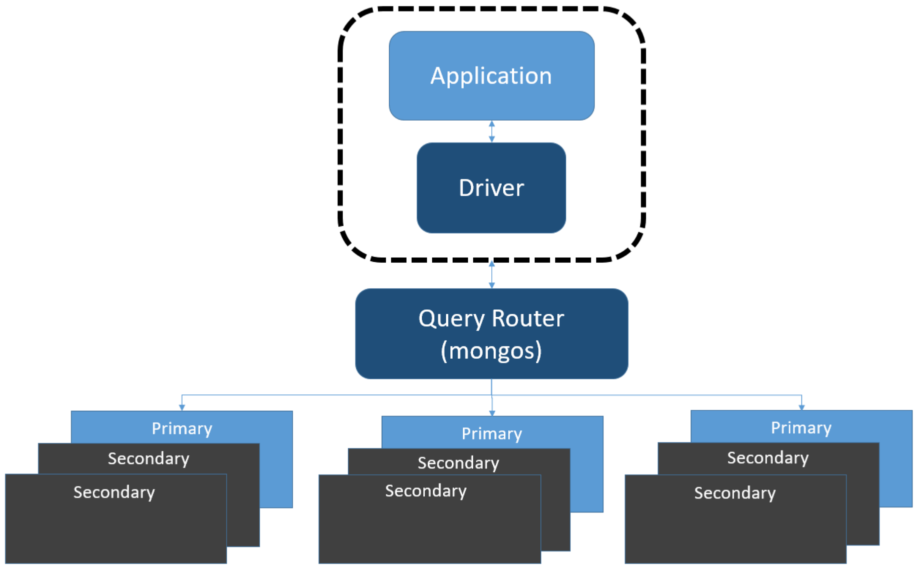

Figure 1: MongoDB Architecture with Sharding

2. Sharding -
   MongoDB supports horizontal scale-out architecture to address hardware limitation for databases by utilizing commodity hardware to distribute data among those hardware and this technique is called Sharding and those commodity hardware nodes are called shards. In case of addition or removal of shards or exceptional data growth, Sharding is automatically overseen by MongoDB.

3. Replica Sets -
   As part of Sharding implementation, MongoDB makes numerous copies of data on different hardware, those copies are called Replica Sets. It is accomplished with native replication support where one replica set performs the role of a primary set and other act as a secondary set. All the read and write operations to a replica set are made through the primary set and then copies are distributed on secondary sets. In case of primary sets downtime, any of the secondary sets gets elected as primary to take request from the application. Sets are self-healing shards which means that it automatically takes care of failure and reduces the database downtime.

4. Query Model -
   MongoDB query model consists query router, it provides transparency which implies that regardless of the number of shards the query code has to be triggered the same way. It optimizes query performance with help of caching, query plan and indexes. It also provides support for query optimization at run time by utilizing more than one index.

3.2 MySQL

MySQL standout amongst the most adaptable and flexible relational database architecture and its behaviour can be tuned dependent on optimum utilization of the diverse components. It is mainly divided into 3 layers, first is where all the connection management happens, second is core processing unite of architecture where all the processing takes place and third is data engines where data storage and retrieval takes place Oracle (2019).

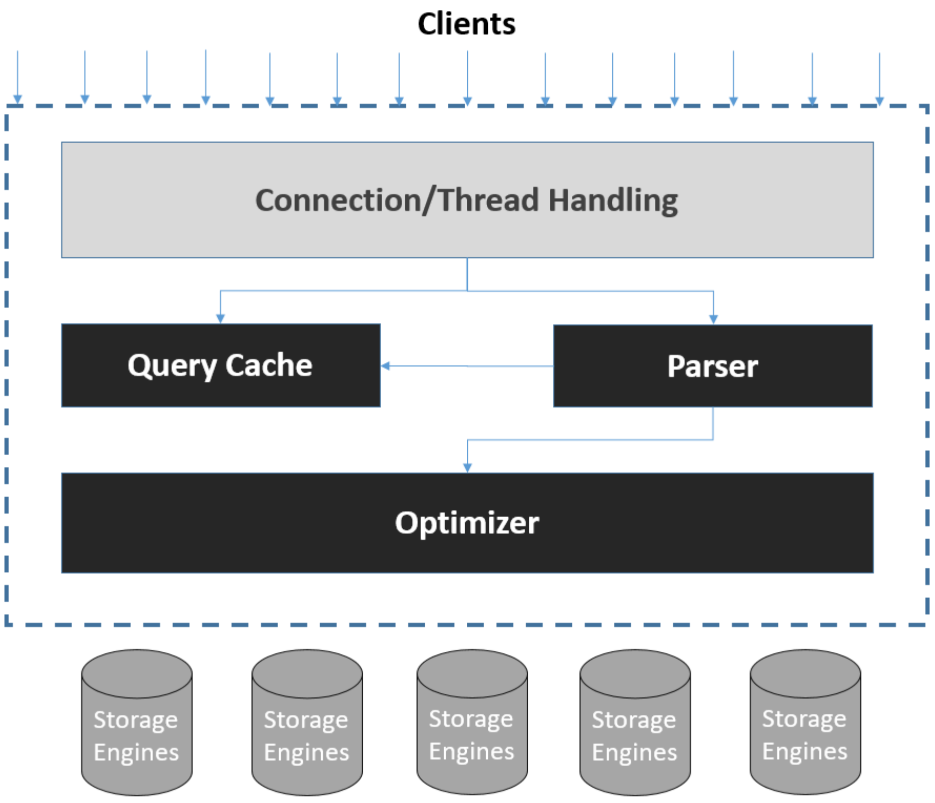

Figure 2: MySQL Architecture with core components

1. Client -
   The client is where all the application accessing MySQL lives and access server for storage and retrieval of information. It very well may be accomplished by using utility driver provided to connect server thread managers like ODBC or JDBC.

2. Connection Management -
   Every client connection gets one dedicated thread in the server and within that connection, the query gets executed for the client. The server also manages caches threads for reducing overhead for connection management. The server allows access to each client connection request with username, password and host-based authentication.

3. Optimization/Execution -
   MySQL as part of query execution takes request from thread received from the client and parses queries to create a parse tree which is similar to execution plan and then optimizes it by rewriting query busing variety of methods utilising indexes and cache. The storage engine is affected by how the server optimizes the user request for best performance, hence optimizer in server consults with storage engines to generate the most agent execution plan.

4. Metadata Cache -
   Cache in MySQL assumes the indispensable role for performance and it is mainly

used in the form of Query and Key cache. When a client sends similar query then the server does not parse and execute that identical query again it just extracts execution plan from query cache whereas Key cache is used for storing table indexes which later used by storage engines.

5. Storage Engine -
   The storage engine is a crucial part of MySQL server as it is responsible for the data management in the physical disks. It executes write and read operation on the physical data files in disks. InnoDB is one the most broadly utilized and oracle recommended default storage engine for almost all MySQL requirements.

4 Capabilities of the Database Management Systems

This section describes and compares MongoDB and MySQL on the scale of Scalability, Availability, Reliability and Transaction Management as briefly discussed by Shah (2017).

4.1 Scalability, Availability and Reliability

As per the CAP theorem any distributed data management system cannot achieve more than two attributes from CAP at the same time and same is the case for both of the database systems used in this study. MongoDB priorities CP (consistency & partition) which means that MongoDB design compromises on availability. While making sure that data is consistently available for user requests by deprioritizing availability also with flexibility on schema and inbuilt replica sets scalability can be easily achieved. Opposite to it in MySQL technical architecture strictly follows CA (consistency and availability) which means that data will be available on and consistence across node but on partition tolerance hence may not better perform in case of partition between multiple nodes.

4.2 Transaction Management

MySQL is open source relational database management system which strictly follows ACID property of transaction (Atomic, Consistent, Isolated and Durable) model. Which is highly suitable for application highly relays on the transitions completeness. Which is not readily available in MongoDB NoSQL architecture however it can be achieved with newer version recognizing need of ACID in NoSQL framework but it is still requires difficult implementations and complex configuration. MongoDB design follows BASE (Basically Available, Soft-State and Eventual Consistency) characteristics there is lack of atomic transactions but it focuses on availability most of the time by making multiple copies over various physical disks and achieves eventual consistency by making sure data is consistent across node at.

5 Findings from Literature Survey

The dramatic increase in Big Data applications has exploded the demand for better performing, salable and flexible database management system. This demands in the market have led to the explosion of vendors coming out with various innovative database architecture and solutions and NoSQL is one of that concept that got the breakthrough for the Big Data applications Panda et al. (2015). Many studies has happened which rules out the idea that NoSQL can replace the Relational database due to its ACID compliance and complex operations support. As rightly said in the study by Wei Sun (October 28, 2013), that when contrasted with RDBMS, NoSQL frameworks are built to address the interest of system that can process a large amount of data with high adaptability, accessibility and faster performance while waving off the ACID compliance.

However, RDBMS solutions have been in place for many years and still being used by many organizations. It becomes crucial to understand how both compare in performance with the help of benchmarking data storage tools. There are many benchmarking tools available in market. The study by Bermbach D. (2017) says that the most famous benchmark standard used to be TPC however that was only used for the RDBMS. With the advancement of Big Data and NoSQL applications that might not be the right choice. At the moment the widely accepted choice for NoSQL and Relational database comparison is Yahoo! Cloud Serving Benchmark (YCSB) as it does not require the system to have a relational schema or transactional support hence easy integration of NoSQL. In this project, we are using YCSB to evaluate the performance of Relational and NoSQL database system on the cloud.

Lately, in NoSQL era YCSB has been used in multiple academic and institutional studies, One such is an experiment by JeonDong-cheol et al. (2018) for the relative examination of NoSQL with RDBMS dependent on medicinal data shows that for medical information processing systems, NoSQL has better execution performance for large volume data sets as compared to relational. However, there has been similar research done by Abramova et al. (2015) for SQL or relational (MySQL) and NoSQL (MongoDB) platforms for the purpose of performance evaluation on extracting decision support by complex queries on both system and results indicate that NoSQL missed the mark amid execution as MongoDB like NoSQL system built for simple key-value pair data retrieval and insertion.

MySQL is one of the widely used Relational databases till date and MongoDB represents the best of NoSQL, Research by Aboutorabi et al. (2015) studies the performance of the MySQL and MongoDB database to replicate the large system having e-commerce data and it directs that the MongoDB performs way more efficiently as compared to MySQL over multiple reads and write operation tests. However, there are also study like R. Panda (2015) which opens discussion with the argument that performance benchmarking also depends on different database implementation design and hence may require various benchmarks before deriving the conclusion. Research by Cornelia Gyrdi (2016) studying MySQL and MongoDB as a backend system that processes a large volume of the user simultaneously accessing the system. It supports the previous argument that MongoDB offers the best possible solution as far as the speed at which diverse tasks can be performed in parallel is a concern.

This research papers and studies happened previously helps us get an overview of what are all the possible options for the purpose of cloud-based databases benchmarking and what were the results generated from YCSB for applications specific tests replicating medical, e-commerce and large back end online systems. In this project, we are studying the performance of MYSQL and MongoDB by reproducing application environment using large (150000), small (50000) and in between size of data volume with YCSB update heavy and read-modify heavy workloads.

6 Performance Test Plan

To evaluate test performance of two database systems we are using Yahoo! Cloud Serving Benchmark where one database is Relational (MYSQL) and second is NoSQL (MongoDB). This test will extract results from low volume data sets to as high as 1,50,000 record data sets load to broaden coverage of applications.

1. Local Machine
   - Processor: Intel(R) Cire(TM) i7 CPU @2.80GHz
   - RAM: 16GB
   - OS: Windows 10
   - OS Type: x64bit based OS
   - CPU Cores: Octa-Core (8)
2. Cloud Hosted Virtual Machine
   - OS: Ubuntu 18 (Bionic)
   - Cloud Host: Openstack
   - Instance Name: x18134751-DSMPROJB
   - Specs: m1.medium (2 VCPU, 40GB Disk)
   - RAM: 4GB
   - IP Addresses: 192.168.101.155 (Floating: 87.44.4.63)
3. MongoDB Configuration
   - Version: v3.2.10
   - Database: ycsb
   - Test Collection: usertable
4. MySQL Configuration
   - Version: MySQL Ver 14.14 Distrib 5.7.25
   - Database: BenchTest
   - Test Table: usertable
5. Benchmarking and Testharness
   - Tool: Yahoo! Cloud Serving Benchmark
   - Version: ycsb-0.15.0
   - Testharness Setup: NCI provided Unix shell based bash script setup to automate YCSB test workloads
6. Workloads
   - Workload A: To replicate environment of application having read and write ratio of 50-50%.

- Workload F: To replicate environment of application having 50% read and 50% Read-modify-write activities. Example of application, Where users entries or records are read and modified by the users.
- Load and Run Operations:
  1) 50,000
  2) 1,00,000
  3) 1,20,000
  4) 1,30,000
  5) 1,50,000

7. Visualization Tool
   - Tool: Tableau Software
   - Version: 2019.1.0

7 Evaluation and Results

To evaluate performance of the NoSQL and Relational database we have used Yahoo! Cloud Serving Benchmark which allows to trigger load and run test with various ratio of read and write operations to capture latency and throughput for the database system.

For this study we are going to evaluate performance with below two workloads of YCSB 15,

- Workload A: Update heavy workload (Read/Update ratio: 50/50)
- Workload F: Read-modify-write workload (Read/Read-Modify-Write ratio: 50/50)

7.1 Workload A: Update Heavy Workload

With help of Testharness scripts, Workload A has been executed against five operation sets (50000,100000,120000,130000,150000) to extract Read, Write operation performance.

7.1.1 Overall run operations throughput (ops/sec) comparison with operation counts

This section describes overall throughput (ops/sec) of both the database to analyze how both performed when executed for multiple data sets.

Workload	Database	Total Operations	Throughput(ops/sec) Total
Workload A	MySQL	50000	13.5465377758752
Workload A	MongoDB	50000	4762.35831983998
Workload A	MySQL	100000	17.4918903223493
Workload A	MongoDB	100000	4532.26976069615
Workload A	MySQL	120000	8.08997017632494
Workload A	MongoDB	120000	4808.46289469466
Workload A	MySQL	130000	11.5190999081661
Workload A	MongoDB	130000	4401.70650775377
Workload A	MySQL	150000	14.0697635154148
Workload A	MongoDB	150000	3621.87613183629

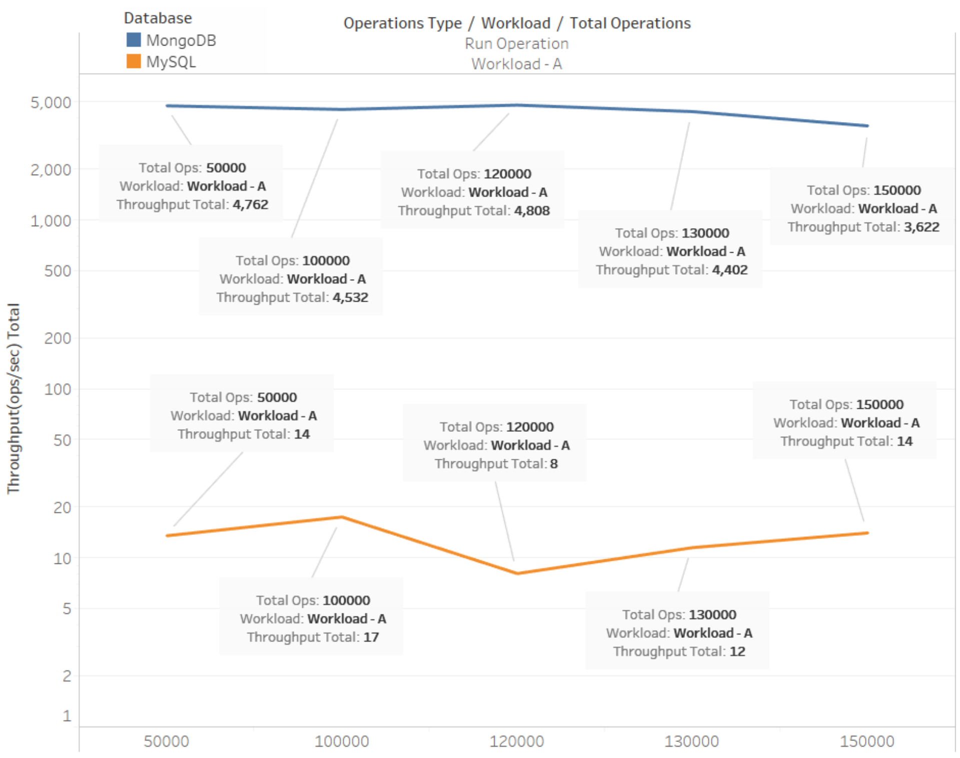

Figure 3: Overall Throughput (ops/sec) Vs Operation Counts

Overall it includes total run throughput of workload A (Insert,Read,Update). It is visible in Figure 3 that MongoDB's overall throughput is between rang from 3622 to 4808 operations per second where as MySQL ranges from 8 to 17 operations per second which is way more less than of MongoDB. MongoDB gives best throughput when executed largest data set (1,50,000) where as MySQL shows it's best throughput at 1,00,000 operation count.

7.1.2 Average latency (μs) comparison with total READ operations

This section describes Average latency (μs) for various data operations of READ activity.

Workload	Database	READ Operations	Throughput(ops/sec) Total
Workload A	MySQL	24917	299.836256371152
Workload A	MongoDB	25110	174.844802867383
Workload A	MySQL	49980	4132.02160864345
Workload A	MongoDB	49896	204.738355780022
Workload A	MySQL	59842	15605.8340630326
Workload A	MongoDB	60061	173.016882835783
Workload A	MySQL	65278	8123.6801678973
Workload A	MongoDB	65172	199.977505677284
Workload A	MySQL	75009	6417.72324654374
Workload A	MongoDB	75216	251.615879600085

The bar graph in Figure 4 shows that for the Read operation of Workload A, MySQL gives best performance for small data set and average latency keep on increasing with

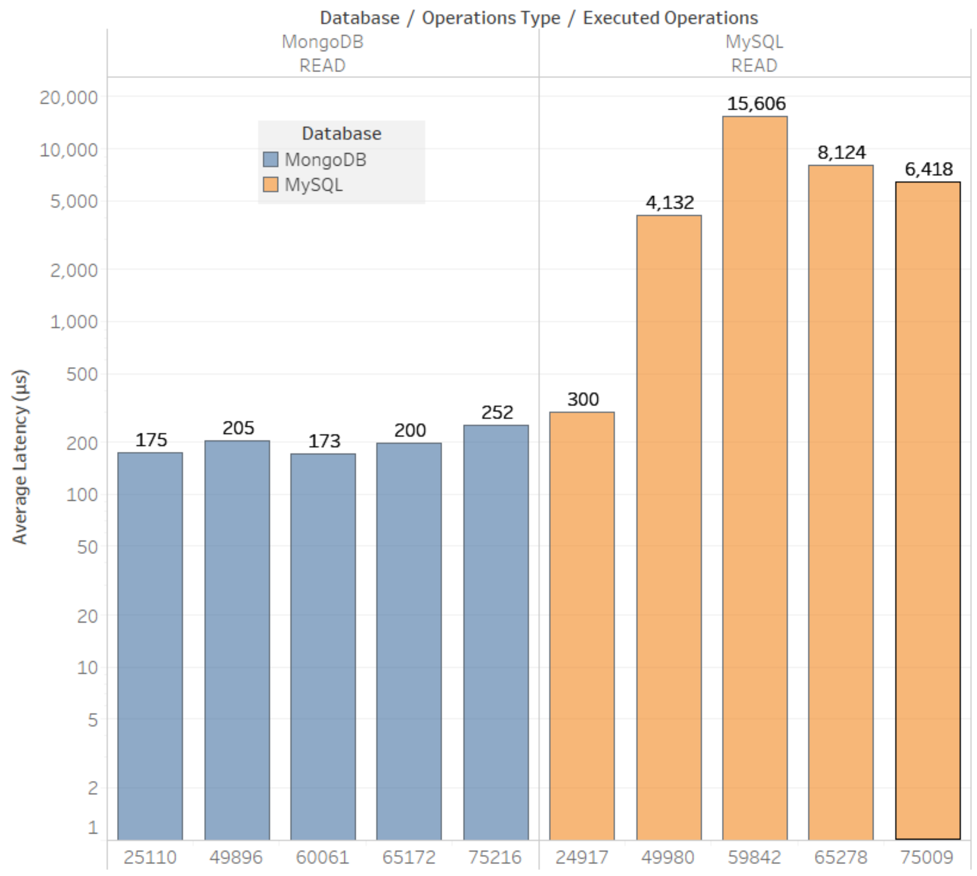

Figure 4: Average latency (μs) Vs Total READ Operations

increase in operation count. It is important because this read operation performance of MySQL is comparably close to NoSQL MongoDB's read operation latency. MySQL latency is highest for operation count 59842 whereas MongoDB stays stable across various operation counts.

7.1.3 Average latency (μs) comparison with total INSERT operations

This comparison describes Average latency (μs) for Insert operation on various records count.

Workload	Database	INSERT Operations	Throughput(ops/sec) Total
Workload A	MySQL	50000	84255.18922
Workload A	MongoDB	50000	157.84082
Workload A	MySQL	100000	143253.17659
Workload A	MongoDB	100000	230.53495
Workload A	MySQL	120000	229566.452766666
Workload A	MongoDB	120000	169.246091666666
Workload A	MySQL	130000	173113.661330769
Workload A	MongoDB	130000	186.373084615384
Workload A	MySQL	150000	136212.42236
Workload A	MongoDB	150000	154.533566666666

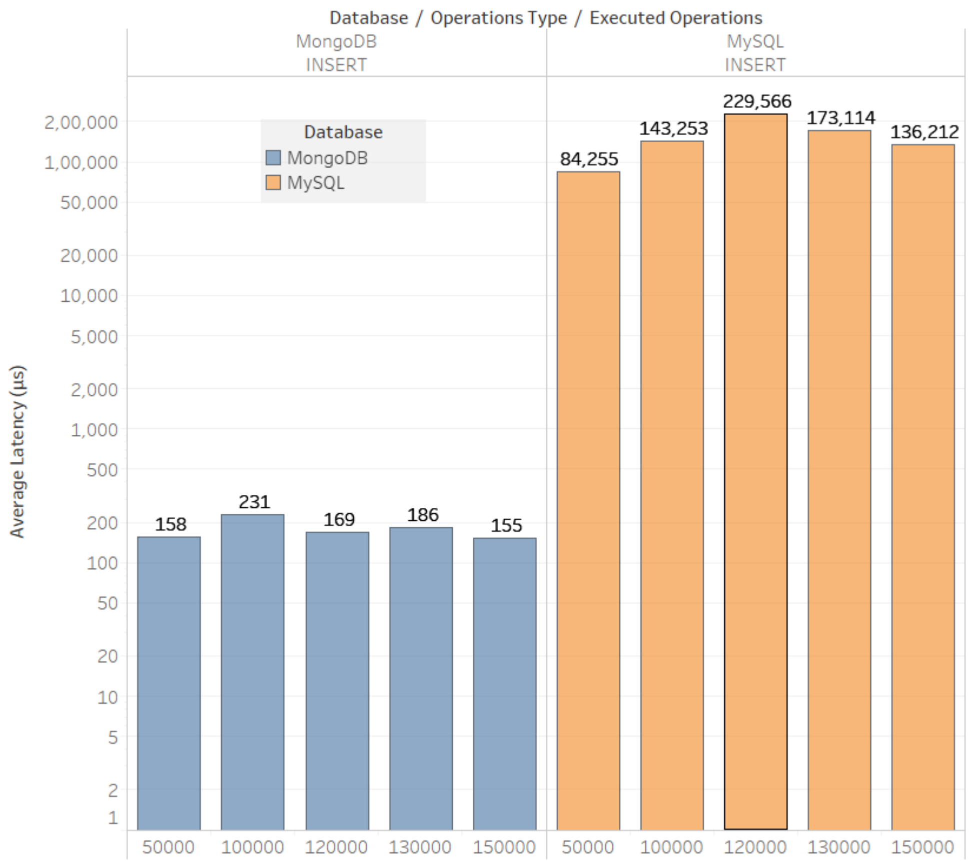

Figure 5: Average latency (μs) Vs Total INSERT Operations

For the workload A Insert operation Figure 5 shows, Average latency performance of MongoDB is comparatively having high latency at 100000 operation count and depicts least latency for operation count of 1,50,000. MySQL's latency is extremely high s compared to MongoDB for workload A insert operations where worst latency is coming for 1,20,000 operations at 229,566 (μs).

7.1.4 Average latency (μs) comparison with total UPDATE operations

This section compares Average latency (μs) on various data sets for Update operations.

Workload	Database	UPDATE Operations	Throughput(ops/sec) Total
Workload A	MySQL	25083	146816.669497269
Workload A	MongoDB	24890	205.417436721574
Workload A	MySQL	50020	110136.949240303
Workload A	MongoDB	50104	215.094902602586
Workload A	MySQL	60158	231020.341500714
Workload A	MongoDB	59939	221.28131934133
Workload A	MySQL	64722	166152.64990266
Workload A	MongoDB	64828	235.278706731659
Workload A	MySQL	74991	135723.834953527
Workload A	MongoDB	74784	284.776757060333

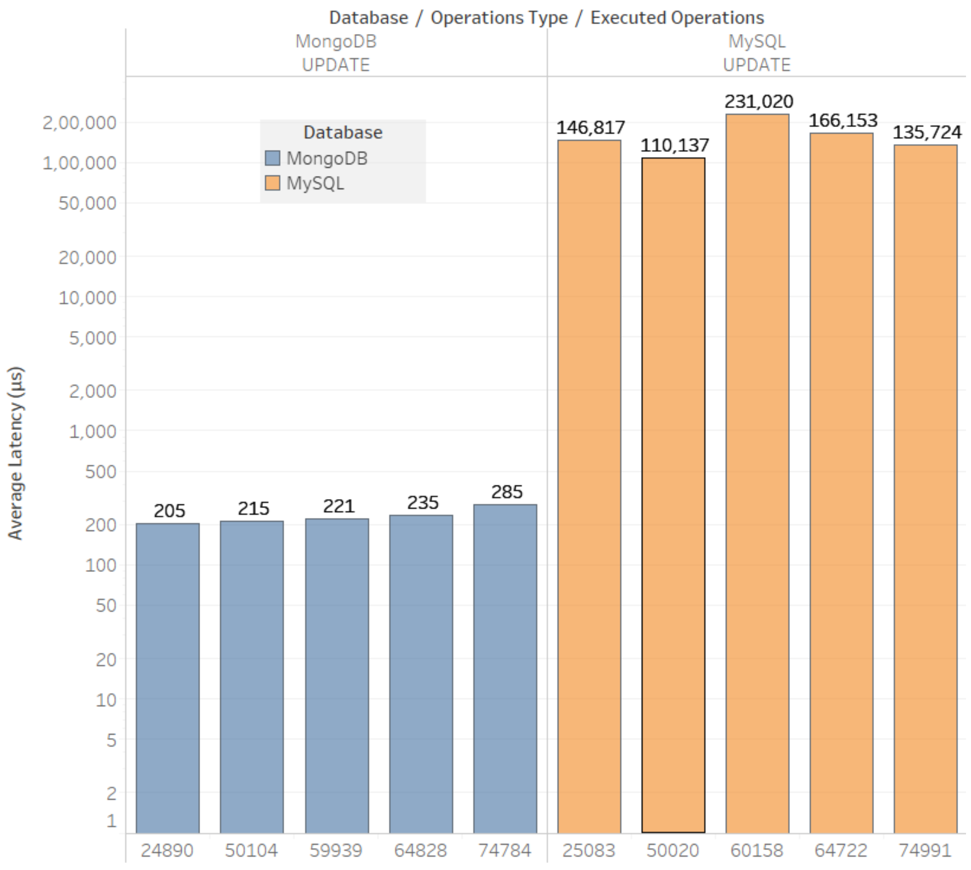

Figure 6: Average latency (μs) Vs Total UPDATE Operations

As depicted in Figure 6, Comparatively MongoDB stays way ahead of MySQL for all the records count of update operation. however MongoDB's average latency stays close to constant but MySQl's latency for update decrease after 60158 record operation where MySQL performs worst for 60158 record update operation.

7.2 Workload F: Read, Modify & Write Workload

With Testharness automated job, Workload F has been executed against five operation sets same as Workload A to extract Read, Write operation performance. In workload F Read operation and Read-Modify-Write operations are divided in 1:1 ratio.

7.2.1 Overall run operations throughput (ops/sec) comparison with operation counts

In this section we will compare performance by overall throughput (ops/sec) of both the database with number of operation executed on database. Operation includes Read, Insert and Read-Modify-Write combination.

Workload	Database	Total Operations	Throughput(ops/sec) Total
Workload F	MySQL	50000	24.3703909498116
Workload F	MongoDB	50000	3640.86506954052
Workload F	MySQL	100000	9.86670281826579
Workload F	MongoDB	100000	3623.05713561102
Workload F	MySQL	120000	11.993439588545
Workload F	MongoDB	120000	3668.60287373891
Workload F	MySQL	130000	11.3786892555238
Workload F	MongoDB	130000	4016.06425702811
Workload F	MySQL	150000	7.92883500570083
Workload F	MongoDB	150000	3627.30636229535

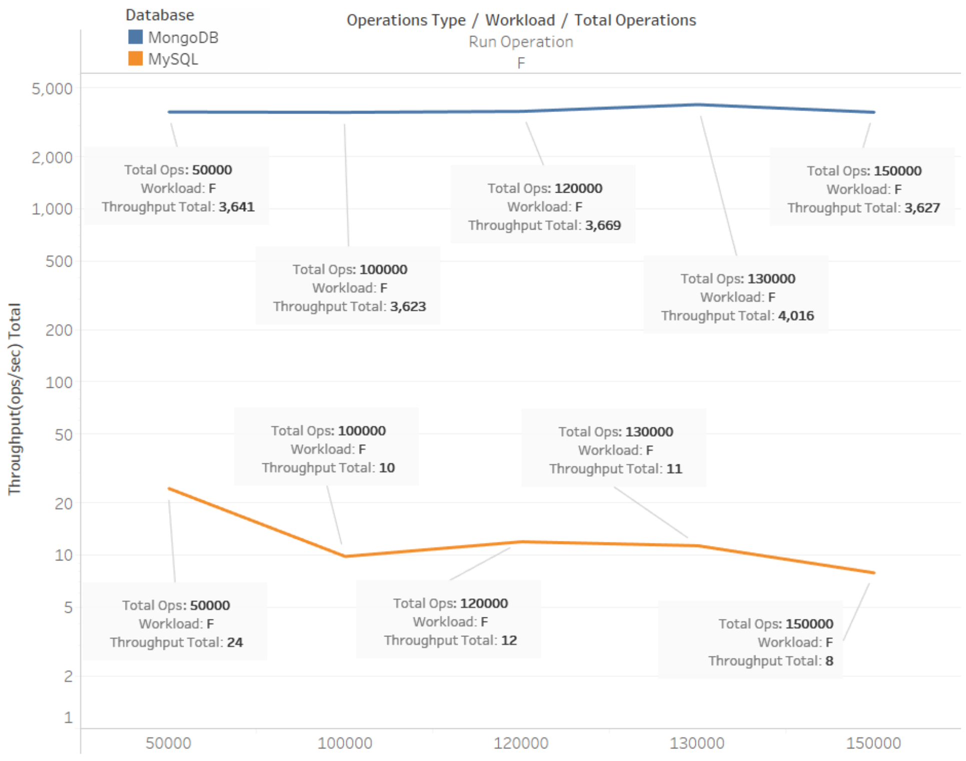

Figure 7: Overall Throughput (ops/sec) Vs Operation Counts

Figure 7 shows that MongoDB and MySQl's overall throughput for all the run operations and it is visible from line chart that MySQL performs maximum 24 operation for 50,000 operation count however if compered with MongoDB's throughput of 3641 for the same number of operation, MongoDB stays way ahead in comparison. There is sudden fall in performance visible for MySQL when increased number of operation from 50k to 100k.

7.2.2 Average latency (μs) comparison with total READ operations

This section compares Average latency (μs) on various data sets for READ operations of Workload F.

Workload	Database	READ Operations	Throughput(ops/sec) Total
Workload F	MySQL	50000	282.90992
Workload F	MongoDB	50000	159.0229
Workload F	MySQL	100000	8880.55071
Workload F	MongoDB	100000	166.43999
Workload F	MySQL	120000	7655.24389166666
Workload F	MySQL	130000	9051.26092307692
Workload F	MySQL	150000	13623.7114666666
Workload F	MongoDB	120000	163.086166666666
Workload F	MongoDB	130000	150.115146153846
Workload F	MongoDB	150000	169.987093333333

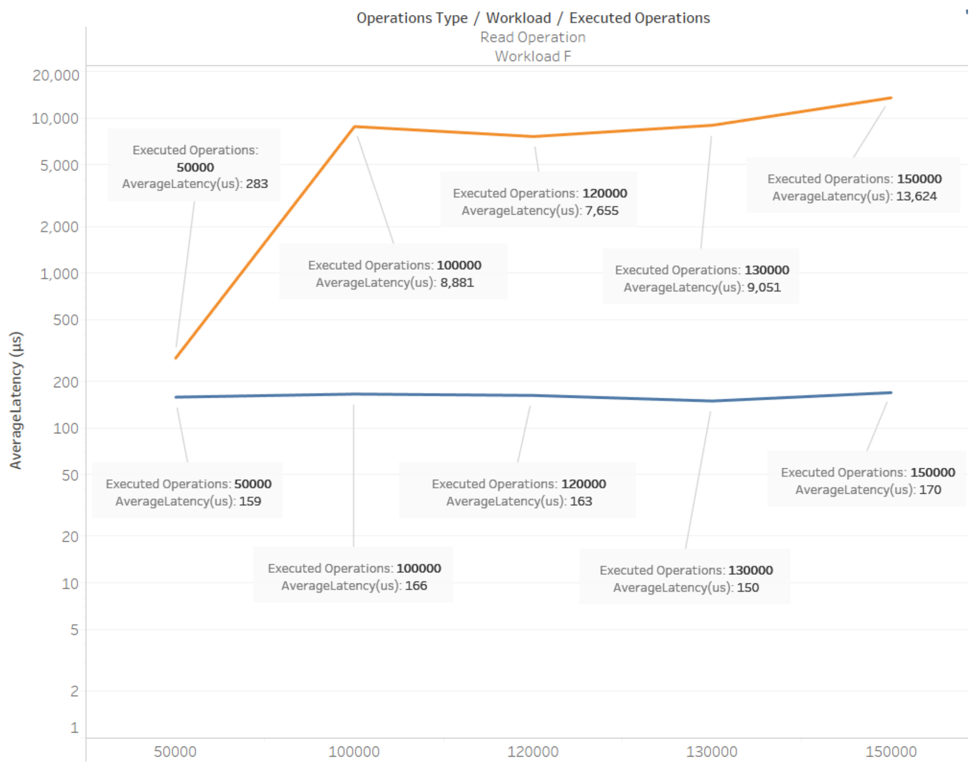

Figure 8: Average latency (μs) Vs Total READ Operations

For the Read operation of Workload F as in Figure 8 confirms findings in Workload A that MySQl's performance for Read operations is comparatively close to average latency of MongoDB when operation count is 50k. Read latency in MySQL after increase in record count to 100k suddenly increase close to 10k (μs) whereas MongoDB stays constant across various operation counts.

7.2.3 Average latency (μs) comparison with total INSERT operations

It describes how MongoDB and MySQL Insert operation performance differs for YCSB workload F on various data sets.

Workload	Database	INSERT Operations	Throughput(ops/sec) Total
Workload F	MySQL	50000	192370.22448
Workload F	MongoDB	50000	168.49062
Workload F	MySQL	100000	174048.88175
Workload F	MongoDB	100000	252.37215
Workload F	MySQL	120000	169414.859641666
Workload F	MySQL	130000	172377.422107692
Workload F	MySQL	150000	225773.478133333
Workload F	MongoDB	120000	149.165983333333
Workload F	MongoDB	130000	158.4471
Workload F	MongoDB	150000	154.42286

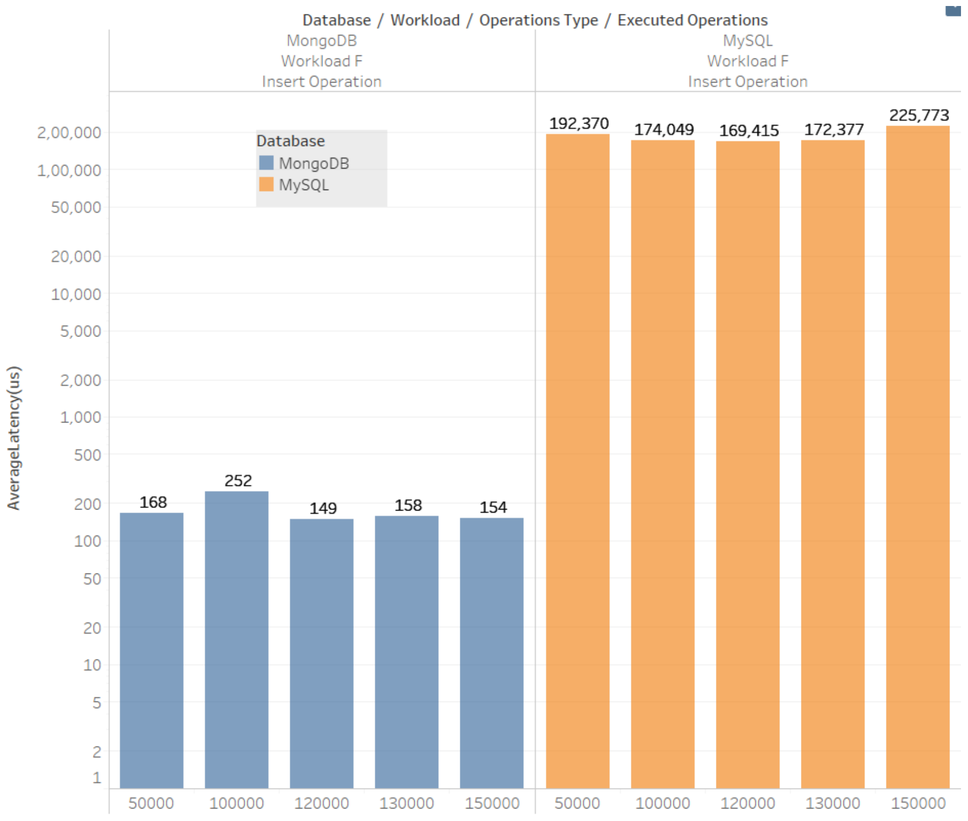

Figure 9: Average latency (s) Vs Total INSERT Operations

Overall average latency for insert operation of Workload F in MySQL stays constantly high where in MongoDB it is close to constant except for operation count 100K where average latency is high as compare to other load of MongoDB at 252 (μs) as shown in Figure 9. For workload F as well insert latency of MongoDB is trading high on performance as compared to MySQL.

7.2.4 Average latency (μs) comparison with total READ-MODIFY-WRITE operations

Read-Modify-Write operation hold weight of 50% for entire workload F and this section compares performance of those operations.

Workload	Database	READ-MODIFY-WRITE Operations	Throughput(ops/sec) Total
Workload F	MySQL	24912	82038.9455282594
Workload F	MongoDB	24967	354.608843673649
Workload F	MySQL	49890	194606.567187813
Workload F	MongoDB	50103	380.892980460251
Workload F	MySQL	59928	158978.945885062
Workload F	MySQL	64861	167095.747814557
Workload F	MySQL	74861	238461.344251345
Workload F	MongoDB	60185	361.683658718949
Workload F	MongoDB	64935	329.64623084623
Workload F	MongoDB	75002	363.48945361457

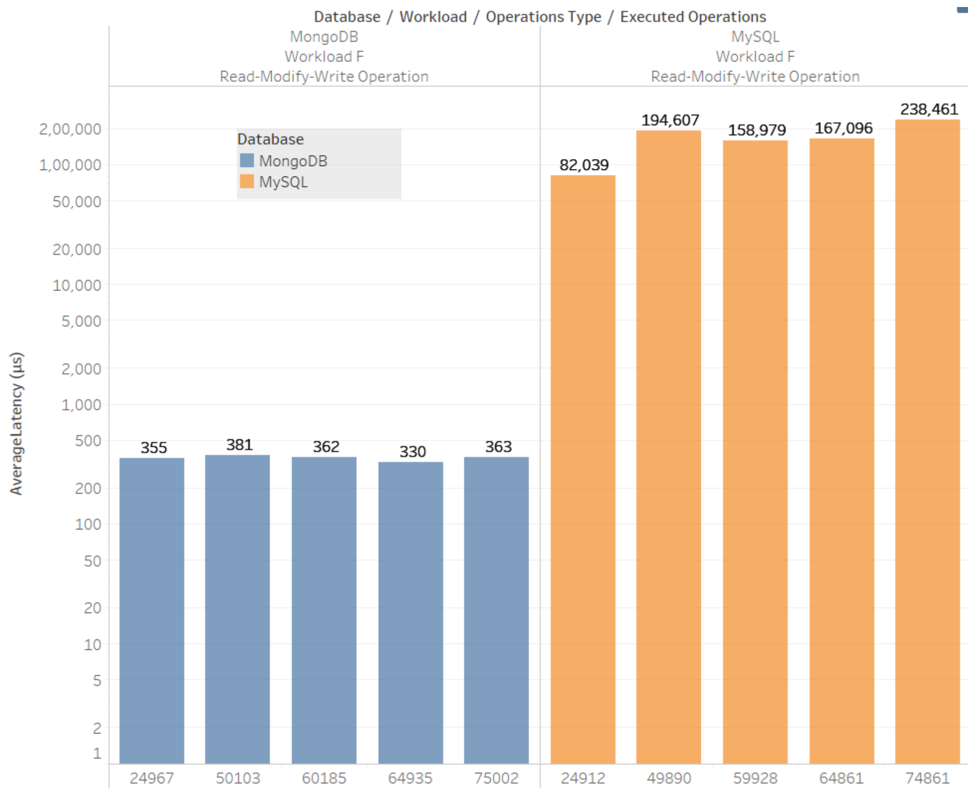

Figure 10: Average latency (μs) Vs Total READ-MODIFY-WRITE Operations

In workload F read-modify-write run, Average latency of MySQL for all the operation counts stands extremely high between 80k to 300k whereas MongoDB's latency is between 200 to 500. MongoDB's latency stays almost constant irrespective of operation count while MySQL indicates least latency when executed load for least operations and it shows worst performance when triggered for largest test volume data set.

8 Conclusion and Discussion

To conclude the project, Comparing NoSQL and Relational database system on cloud, we carried out various performance test on database with help of Yahoo! Cloud Serving

Benchmark where we used Workload A which replicates update heavy application workload with read and update ratio of 50/50 and Workload F that depicts performance of read-modify-write application workload with read/read-modify-write ratio of 50/50. We executed both workload on different five data volumes to test the performance with large and small volume data set. As previously concluded in many studies MongoDB surpasses way ahead in the test results as compared to MySQL on all the size of workloads except for the read operation where MySQL and MongoDB performance is close for low volume test data set. All the database on cloud were running on single node architecture for the purpose of this study in the future work, we aim to study the performance fluctuation of MySQL on shared multi-node cloud environment with MongoDB to further move the study towards the conclusion.

References

Aboutorabi, S., Rezapour, M., Moradi, M. & Ghadiri, N. (2015), 'Performance evaluation of sql and mongodb databases for big e-commerce data'.
URL: *https://doi.org/10.1109/CSICSSE.2015.7369245*

Abramova, V., Bernardino, J. & Furtado, P. (2015), 'Sql or nosql? performance and scalability evaluation', *International Journal of Business Process Integration and Management* **7**, 314.
URL: *https://doi.org/10.1504/IJBPIM.2015.073655*

Bermbach D., Wittern E., T. S. (2017), 'Getting started in cloud service benchmarking', *Cloud Service Benchmarking* **7**.
URL: *https://doi.org/10.1007/978-3-319-55483-9*$_1$4

Cornelia Gyrdi, Ioana Andrada Olah, R. G. L. B. (2016), 'A comparative study between the capabilities of mysql vs. mongodb as a back-end for an online platform', *International Journal of Advanced Computer Science and Applications* **7**(11).
URL: *https://doi.org/10.14569/IJACSA.2016.071111*

Huang Y., L. T. (2014), 'Nosql database: A scalable, availability, high performance storage for big data', *Pervasive Computing and the Networked World* .
URL: *https://doi.org/10.1007/978-3-319-09265-2*$_1$9

JeonDong-cheol, Mun, L. & HwangHeejoung (2018), 'Performance analysis of rdbms and mongodb through ycsb in medical data processing system based hl7 fhir', *Journal of Korea Multimedia Society* **21**(8), 934–941.
URL: *https://doi.org/10.9717/KMMS.2018.21.8.934*

MongoDB, I. (2019), 'Introduction to mongodb'.
URL: *https://docs.mongodb.com/manual/introduction*

Oracle (2019), 'Mysql documentation'.
URL: *https://dev.mysql.com/doc/*

Panda, R., Erb, C., LeBeane, M., Ryoo, J. H. & John, L. K. (2015), 'Performance characterization of modern databases on out-of-order cpus', pp. 114–121.
URL: *https://doi.org/10.1109/SBAC-PAD.2015.31*

R. Panda, C. Erb, M. L. J. H. R. L. K. J. (2015), ‘Performance characterization of modern databases on out-of-order cpus’, pp. 114–121.
URL: *https://doi.org/10.1109/SBAC-PAD.2015.31*

Shah, M. (2017), ‘Mongodb vs mysql: A comparative study on databases’.
URL: *https://www.simform.com/mongodb-vs-mysql-databases/*

StackExchange (2018), ‘Developer survey results 2018’.
URL: *https://insights.stackoverflow.com/survey/2018*

Wei Sun, A. P. (October 28, 2013), ‘Benchmarking availability of large scale data storage applications’, *CS848* **1**.