National College of Ireland

Data storage and Management Project B

on

Performance analysis of MongoDB and HBase

Sindhujan Dhayalan
X17170265

MSc/PGDip Data Analytics – 2018/9

Submitted to: Vikas Tomer

Performance analysis of MongoDB and HBase

Sindhujan Dhayalan
X17170265

December 19, 2018

Abstract

Internet involvement in everyone's life is growing exponentially in the recent years. This is directly proportionate to generation of data, Data is collected in enormous rate. Traditional databases are not efficient in handling the huge amount of data collected each second. NoSQL databases provide the solution to handle, store and randomly access data from databases effectively. MongoDB and HBase are two top NoSQL databases used by many famous internet application giants. In this paper, The read, update and insert operation performances of the two databases are evaluated and compared using different workloads.

1 Introduction

Data storage and maintenance were accomplished using relational database management systems in the past. But with the rise of big data, companies need a solution for processing the big data and preferred NoSQL database like Hadoop. Which stores huge amount of data in a distributed file system(HDFS) and used MapReduce to process the big data. Hadoop performed well in handling all forms of data including unstructured and semi structured data.

Hadoop also was not sufficient enough for data to be accessed randomly, as it allowed only batch processing and can query the data sequentially. Hence the requirement of random access of the data in the database raised, resulting in the evolution of applications such as HBase, MongoDB, Dynamo, Cassandra and CouchDB. These NoSQL database management applications can store big data and perform queries to randomly access any part of the data.

These databases are more scalable compared to SQL databases, contains schema less data model and are secure. In this project, the performance of HBase and MongoDB will be performed, analyzed and compared against each other with both running on the same environmental setup and workload. George (2011)

2 Key Characteristics of MongoDB & HBase Data Storage Management Systems

2.1 MongoDB

Mongo DB is an open source, NoSQL Database that uses document-orientated data model primarily written in C++. It is a schema-less database, which can store over tera bytes

of data in its database. The documents in MongoDB can have different structures and fields. Data is treated in BSON data format (Binary JSON format). MongoDB can be queried using dynamic object-based language and java script.

Characteristics of MongoDB:

- MongoDB supports **document-based queries and ad-hoc queries.**
- **Indexing:** Any field in the document can be indexed.
- **Master slave repliation:** Native application maintains multiple copies of data. Avoiding database downtime is one of replica sets function as it has self-healing shard.
- **Multiple Servers:** The database can run over multiple servers. Data is duplicated to support a flawlessly running system in case of any hardware failure.
- **Auto-Sharding:** Automatic distribution of data across multiple shards resulting in an automatic load balancing feature.
- Supports **MapReduce and flexible aggregation tools.**
- **Failure Handling:** Increased protection and data availability against database downtime like rack failures, multiple machine failures and data centre failures or even network partitions due to the huge number of replicas in MongoDb.
- Stores files of any size without complicating the stack by using **GridFS.** intellipaat.com (2018)

2.2 HBase

HBase is an open source, NoSQL database primarily written in Java. It runs on top of HDFS (Hadoop distributed file system) enhancing throughput and performance of the distributed cluster set up. Resulting in faster random reads and writes operations and can be queried using Map-reduce. HBase can be used for storing large collection of sparse data sets in a fault tolerant way. HBase stores data in the form of key/value pairs in a columnar model, all the columns are grouped together as Column families. HBase provides flexible data model and low latency access to small amounts of data stored in large data sets.

Characteristics of HBase:

- It is a part of the Hadoop ecosystem that provides **random real-time read/write access** to data in the Hadoop File System.
- **Atomic read and write**: While performing one read or write process, all other processes are prevented from performing any read or write operations on row level. Thus, performing **consistent read and write operations.**

- **Linear and modular scalability:** As datasets are distributed over HDFS, HBase is linearly and modularly scalable across various nodes.
- **Automatic and configurable sharding of tables:** HBase tables are distributed across clusters further these clusters are distributed across regions. The regions and clusters are split and redistributed as the data grows.
- **Selective replication** Amount of information exchanged between the replicas can be restricted.
- **Easy to use Java API for client access.** Sinha (2018)

3 Database Architectures

3.1 MongoDB Architecture:

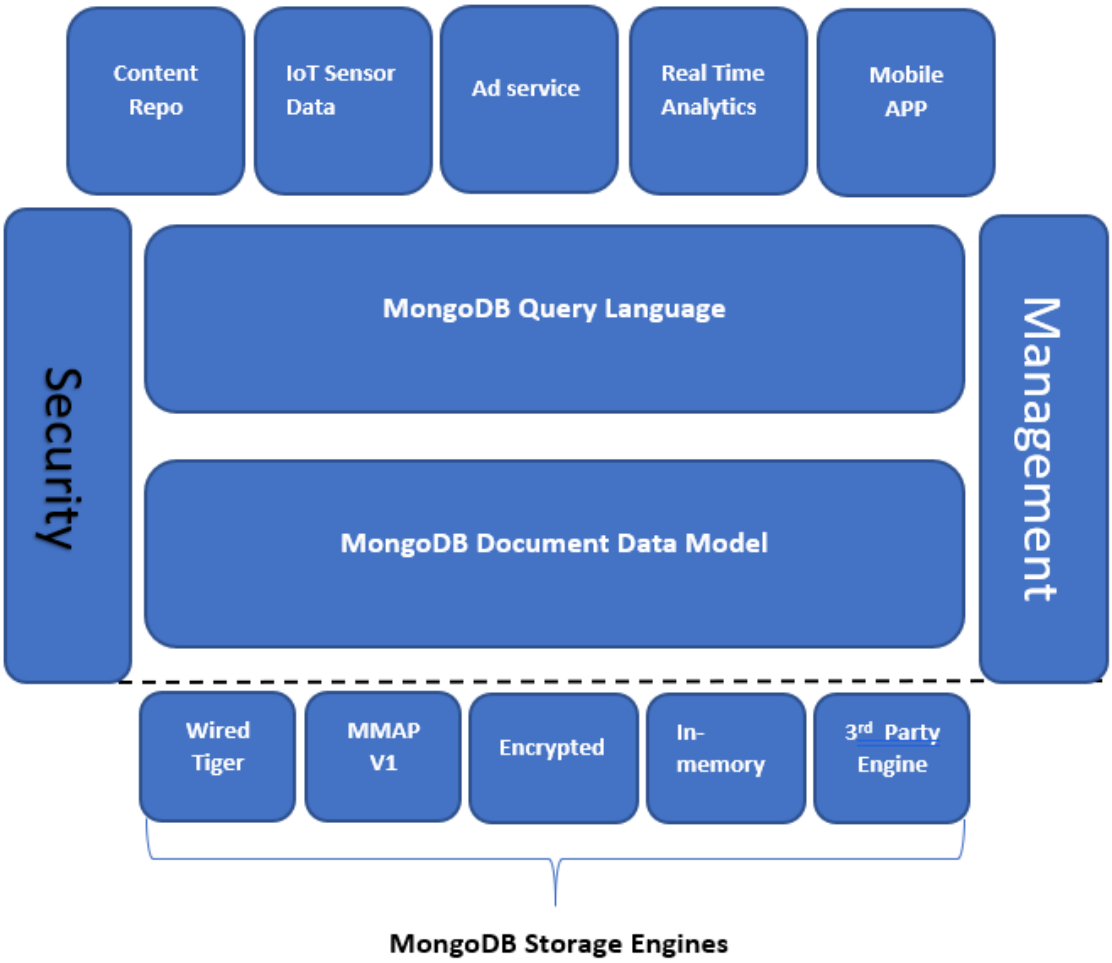

Figure 1: MongoDB Architecture, Source: MongoDB (2015)

MongoDB storage engines:
MongoDB 3.2 version is packed with four compatible storage engines. A single MongoDB

replica set can consist of all the four storage engines together. This feature provides easy evaluation and migration between them and optimization as per the specific requirement of the application.
Wired Tiger Storage engine: Provides the best all-round performance and storage efficiency to vast range of application with its granular concurrency control and native compression.
MMAPv1 storage engine is an improved version of pre-3.x MongoDB releases.
Encrypted storage engine requires the MongoDB Enterprise Advanced application. It provides protection of highly confidential data without any performance or management overhead of separate file system encryption.
In-Memory storage engine requires MongoDB Enterprise Advanced application. It provides extreme performance paired with real time analytics for the highly demanding, latency-sensitive applications.MongoDB (2015)

MongoDB Document data model:
As discussed in the section 2, MongoDB stores data as documents in BSON format.BSON format is an extension of JSON format used in Java script.BSON documents contains one or more fields and each field contains a value of a specific data type.Documents that have a similar structure are organized as collections.The document data model provides rich data structure and Dynamic schema. This type of data model provides Agility and flexibility to change according to meet new requirements,Reduces the need for Joins making programming and query performance better.

MongoDB Query language:
MongoDB provides native drivers for all famous programming languages and frameworks. MongoDB query model is built as function or methods within the API of specific programming language. MongoDB supports different types of queries such as key-value queries, Range queries, Geospatial queries, Test search queries, Aggregation framework queries and MapReduce queries. These queries can be used to return a document, a portion of specific fields in a document or complex aggregation against many documents as required.

MongoDB Data management:
Autosharding technique: MongoDB uses a technique called sharding which provides horizontal scale out for databases in a low-cost commodity hardware or cloud infrastructure. Sharding partitions the data across multiple shards and MongoDB automatically administers the data in sharded cluster as the size of the data grows or size of the cluster increases or decreases. Sharding is automatic and built into the MongoDB database.
Consistency: MongoDB is ACID compliant at the document level. One or more fields can be written in a single operation, including updates. MongoDB (2015)
Availability (Replication): MongoDB maintains multiple copies of data called replica sets using native replication. It is completely self-healing shard that helps prevent database downtime and can be used to scale read operations. Replica failover is fully automated, eliminating the requirement of manual intervention by administrators. The replica sets provides operational flexibility by providing a way to upgrade hardware and software without requiring the database to go offline eliminating the downtime caused due to these activities in RDBMS.

Security:
MongoDB Enterprise Advanced application provides abilities to defend, detect, and control access to data through its native authentication, authorization, auditing, and encryption features.Leite (2015)

3.2 HBase Architecture:

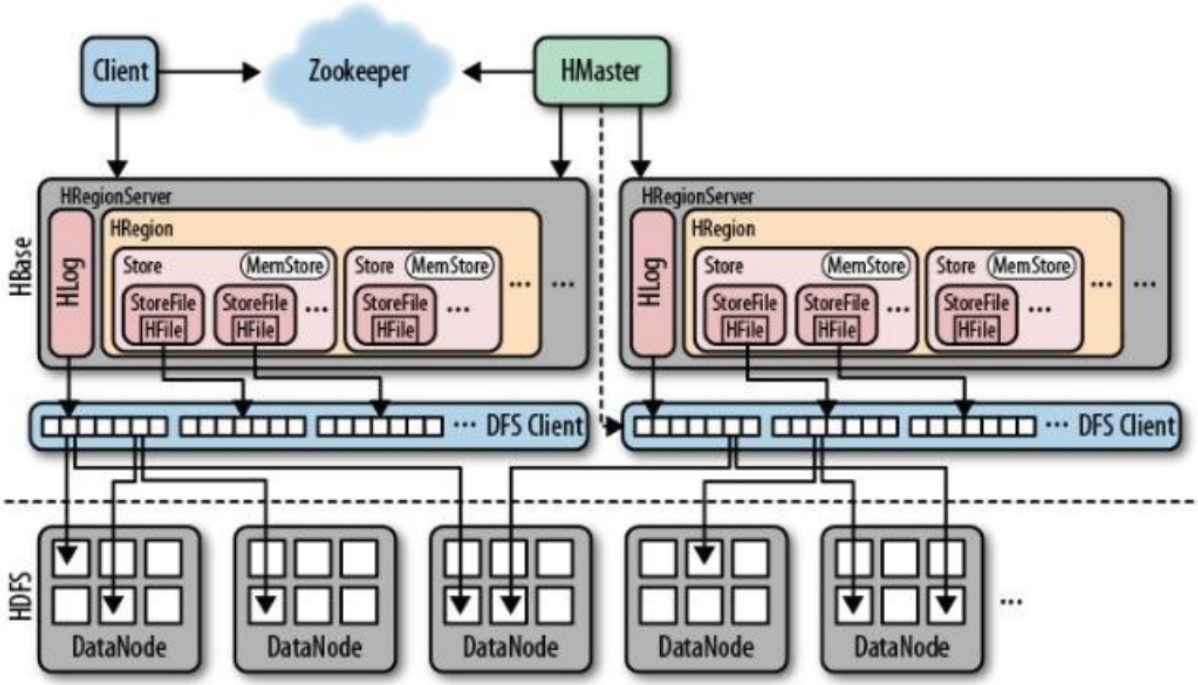

Figure 2: HBase Architecture, Source of the figure: George (2011)

HBase architecture consists of Master node called the HMaster and several slaves as region servers. Each region server (slave) takes care of a set of regions, and a region can be served only by one region server. Whenever a client sends a write request, HMaster receives the request and assigns it to the corresponding region server. dezyre.com (2016) HBase architecture has three main components- HMaster, Region Server and ZooKeeper.

HMaster: HBase HMaster is a master node that assigns regions to region servers in the Hadoop cluster for load balancing. HMaster is responsible for managing and monitoring the Hadoop cluster, performing administration of tables, Controlling the failover, Handles DDL operations and changing the schema or the metadata operations.

Region Server: Region server is the slave nodes which handle read, write, update, and delete requests from clients assigned by the HMaster. Region Server runs on HDFS Data Node and consists of the following components: Block cache, Memstore, WAL and HFile. The Block cache stores the most frequently read data and acts as the read cache. Recently used data is evicted when the block cache is full. Memstore stores the new data that is still not written to the disk. All the column family in a region consist of a Memstore. WAL is Write Ahead Log and stores the new data that is not required for permanent storage. HFile is the file that stores the rows as sorted key values on a disk.

Zookeeper: Zookeeper is used by HBase as a distributed coordination service for region assignments and to recover any region server failures by loading them onto other region servers that are functioning. ZooKeeper maintains configuration information and provides distributed synchronization. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes. ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about the number of region servers and details of the region servers on holding of DataNode. HMaster contacts ZooKeeper to get the details of region servers. dezyre.com (2016)

4 Security features of MongoDB and HBase

MongoDB five core security features are discussed below:

4.1 MongoDB Authentiction:

MongoDB provides multiple authentication methods to be chosen to meet requirement of different environment.
In Database Authentication: MongoDB authenticates on a per-database level using the SCRAM IETF RFC 5802 standard. Authentication command is used to authenticate users and Key files are used to authenticate database nodes to the MongoDB cluster.
LDAP Authentication: LDAP provides a simple way to manage large amount of users across internal systems and applications. It also ensures that the internal security policies are flexible with corporate and regulatory guidelines. MongoDB Enterprise Advanced application with LDAP integration can directly authenticate and authorize users against existing LDAP infrastructure to leverage centralized access control architecture.
Kerberos Authentication: Kerberos is an industry standard authentication protocol for large client/server systems, this allows client and server to verify each others identity. MongoDB with Kerberos integration can take advantage of existing authentication infrastructure and processes.
x.509 Certificate Authentication: With integration of x.509 certificates MongoDB can be integrated with existing information security infrastructure and certificate authorities supporting both user and inter-node authentication. Enabling users to be authenticated using client certificates instead of password.MongoDB (2018)

4.2 MongoDB Authorization:

Administrators in MongoDB can define the specific permissions an application or user has and also control the data to displayed when queried by different users.
Role-based access control (RBAC): Role-based access control is core to MongoDB. Built-in roles have been available in MongoDB since version 2.6. The administrator has the access to limit the specific actions a particular user is allowed, enabling the administrator to have control over the activities the a specific user can view and execute. The five main built-in MongoDB roles are Read, ReadWrite, dbOwner, dbAdminAnyDatabase and root. MongoDB (2018)
LDAP Authorization:

LDAP Authorization supported by MongoDB Enterprise Advanced allows existing user privileges stored in the LDAP server to be linked to MongoDB roles, Users dont have to be recreated in MongoDB. MongoDB will permit user authentication via LDAP, Active Directory, Kerberos, or X.509 without needing local user documents in the external database when configured with an LDAP server for authorization.
Log Redaction:
MongoDB Enterprise Advanced can be integrated with log redaction, which prevents potential sensitive information from being written to the databases diagnostic log.MongoDB (2018)

4.3 MongoDB Auditing:

All the access and actions run against the database is logged in the MongoDB Enterprise Advanced auditing framework logs. Administrative actions (DDL) such as schema operations as well as authentication and authorization activities, along with read and write (DML) operations to the database are captured by the auditing framework.MongoDB (2018)

4.4 MongoDB Encryption:

MongoDB data in transit can be encrypted over the network and MongoDB data at rest can be encrypted in permanent storage and backups by Administrators.
Network Encryption Support for TLS allows clients to connect to MongoDB over an encrypted channel. Clients are the entities capable of connecting to the MongoDB server, including: Users and administrators Applications MongoDB tools Nodes that make up a MongoDB cluster.
Disk Encryption: the MongoDB Encrypted storage engine is an integral feature of the database that provides protection of data at-rest. By natively encrypting database files on disk, administrators avoid both the management and performance overhead of external encryption mechanisms.MongoDB (2018)

4.5 MongoDB Governance:

Governance is a process of validation of documents in MongoDB and verifying the sensitive data . It ensure where sensitive data is stored and preventing sensitive data from being introduced into the system.MongoDB (2015)

The security features of HBase are discussed below:

4.6 HBase Authentication

The process of client authenticating with server to establish credentials ensures secured access to the database. the main three components for HBase authentication is discussed below: 1. Client Authentication: HBase uses Kerberos and SSL security protocols that enable clients to authenticate with a database. 2. Server authentication: HBase uses shared key-file server authentication method to enable database servers to authenticate with each other to ensure a secure operating environment. 3. Credential Store: HBase database account credentials may be stored in the external file securely. mediawiki (2018)

4.7 HBase Role Based Security

Role based security provides support in authorizing access to the database contents for authenticated clients.HBase role based security features are discussed below: 1. HBase Role based security greatly simplifies security administration and operations. 2. HBase Role based security can supports default roles and custom roles features that support ease of administration. 3. Scope of Roles: HBase consist of cluster, database, collection, field scopes for roles. mediawiki (2018)

4.8 HBase Database Security and Logging

The various features for security and logging available at a database server in HBase are discussed in the below section: 1. HBase Database Encryption: Encryption of database on a disk is very useful in highly sensitive application. 2. Logging: Logging of security events is essential for secure operations, detailed event investigations and auditing= In HBase supports configurable event logging, Fixed event logging. mediawiki (2018)

5 Learning from Literature Survey

The NoSQL databases were introduced to produce high performance in terms of speed, storage size and high availability, effective enough to lose the ACID (Atomic, Consistent, Isolated, Durable) trait of the relational databases in exchange with keeping a weaker BASE (Basic Availability, Soft state, Eventual consistency) featureTudorica & Bucur (2011). NoSQL Document model databases, followed by column-family model databases, have a good average performance as they have efficiency and Scalability. At the same time, it was found that database framework in master-master mode has more advantages over the master-slave architecturesTang & Fan (2016). Different NoSQL databases have different features according to their model, which are advantageous to specific types of operations. It is better to examine the different NoSQL databases before making a choice on database suitable for an application. HBase, A column based NoSQL Database has the autonomous functioning of slave nodes in a master-slave architecture that influences the scalability. MongoDB stores the data as BSON (Binary JSON) documents and this could be the reason for the better throughput performance for read only operationsSwaminathan & Elmasri (2016). As a global analysis, in terms of optimization, the databases can be divided into two categories, databases optimized for reads and databases

optimized for updates. Thus, MongoDB, is a database optimized to perform read operations, while HBase have a better performance on execution of updates Abramova et al. (2014*a*). In tests performed in Abramova et al. (2014*b*) MongoDB is the database that showed largest increase in the execution time directly related to the increase of the number of updates performed. This database uses locking mechanisms, which increase execution time. As per the learning in the paper, Decision has been made to choose HBase and MongoDB for performing load in YCSB against different workload to evaluate database performance.

6 Performance Test Plan of MongoDB and HBase

Performance test of HBase and MongoDB is carried out by using the YCSB tool. The Yahoo! Cloud Serving Benchmark (YCSB) is an open-source tool used to compare relative performance of NoSQL database management systems. Mainly for evaluating the performance of different "key-value" and "cloud" serving stores. In this project, YCSB is used to benchmark HBase and MongoDB. Comparing and running HBase and MongoDB on the same hardware configuration and with 3 different workloads. The output files are visualized (example, as latency versus throughput curves) to analyze the performance of one database system against other.

Different workloads used:

- Workload A: Update heavy workload: 50/50% Mix of Reads/Writes.
- Workload B: Read mostly workload: 95/5% Mix of Reads/Writes.
- Workload D: Read the latest workload: More traffic on recent inserts(95/5% read-/insert ratio).

Different Operation counts used:

- 50000
- 100000
- 150000
- 200000
- 300000

Hardware configuration:

- Machine model: Dell XPS15 9750.
- Processor: Intel Core i7-8750H @ 2.20GHz 2.21GHz.
- RAM: 16.00 GB.
- Operating system: Windows 10 - 64-bit operating system.

Virtual machine:

- Operating system version: Ubuntu 18.04.
- RAM: 4GB.
- Disk: 40GB.
- VCPUs: 2 VCPU.

7 Evaluation and Results

7.1 LATENCY OF READ OPERATIONS FOR HBase and MongoDB

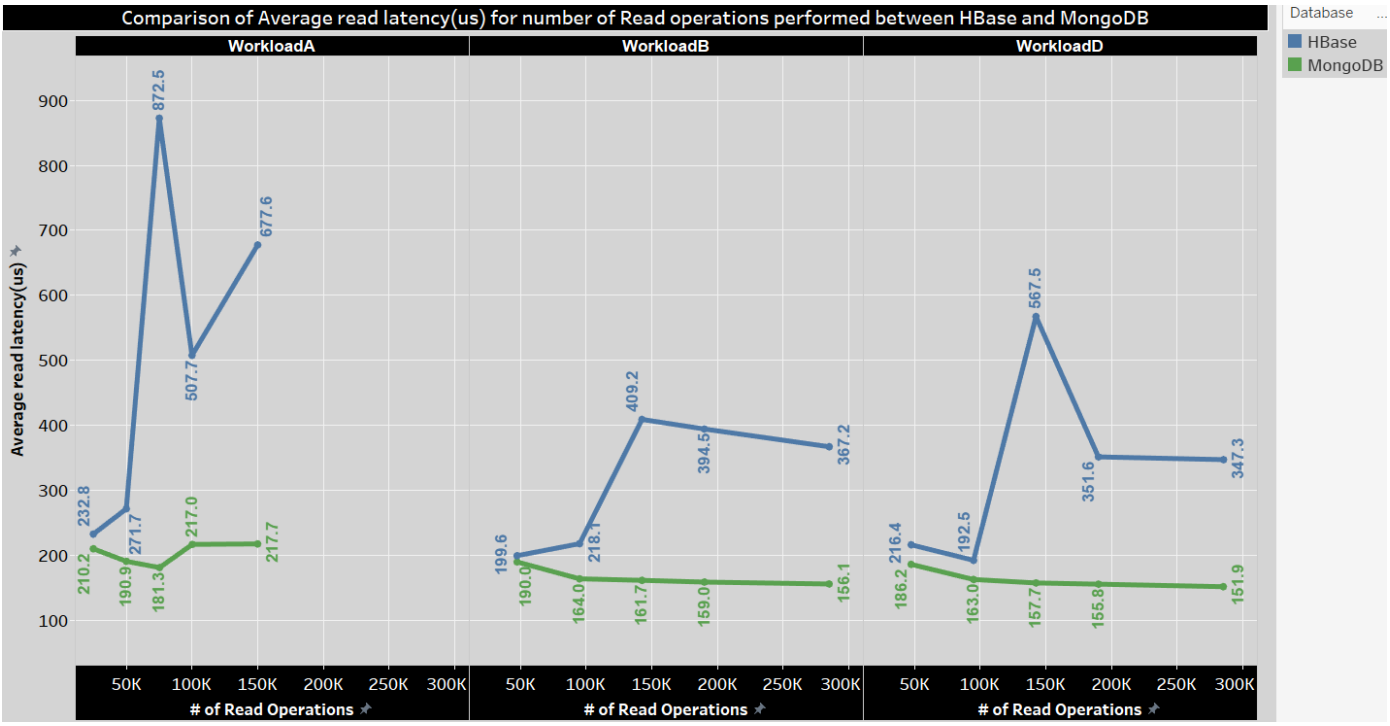

Figure 3: Read average latency VS Total read operations

Figure 3 illustrates the average latency for read operations VS the total number of read operations performed by MongoDB and HBase databases for different workloads namely, Workload A, Workload B and Workload C. The values displayed are average of 3 individual tests run on same environmental setup. Detailed analysis of performance is discussed in this section:

Workload A: Update heavy workload: 50/50% Mix of Reads/Writes.
Both the database systems perform 50% Read operation of the total operational counts. In case of HBase, the Average read latency grows gradually as the number of read operation increases. There is a sudden peak of average latency at 872.5 micro seconds while performing 75000 read operation (50% of 150000), this is due to the presence of outlier value of 1500 micro seconds in Average read latency during run2. Which is 3 times of the values observed in run 1 and run3. The same is highlighted in Figure 4. The presence of

the outlier creating a higher mean value. Hence ignoring the value for 2nd run for normal cases, It is inferred that the Average read latency grows gradually with the increase in Read operational counts.

AverageLatency(us) for Run1	AverageLatency(us) for Run2	AverageLatency(us) for Run3	Total number of operations
242.610	222.30	233.492	50000
271.837	284.71	258.629	100000
532.237	1500.42	584.896	150000
416.027	541.89	565.201	200000
1001.307	545.88	485.485	300000

Figure 4: Outlier for average read latency during run2

Whereas for MongoDB, the average read latency is closer to HBase at read operation count 25000(50% of 50000) but with the increase in read operation the average latency is gradually reducing to a low of 181.3 in 75000 read operation and then rises gradually to maintain a constant range at 217 average latency for 100000 and 200000 read operations.

Workload B: Read mostly workload: 95/5% Mix of Reads/Writes.
Both the database systems perform 95% Read operation for all operational counts. In case of HBase, the Average read latency rises substantially while performing 142500 read operations and then falls gradually at 200000 and 300000 total operational counts. Where as for MongoDB, Average latency is similar to HBase at 47500 read operations but with the increase in read operation the average latency is gradually reducing and are performing in the range of 160s dropping to 156.1 micro seconds at 285000(300000*95%) read operations.

Workload D: Read the latest workload: More traffic on recent inserts.(95/5% read/insert ratio).
Both the database systems perform 95% Read operation for all operational counts.In case of HBase there is a steep rise of average latency while performing read operations at 150000 total operational count and then falls for read operations at 200000 Operational count, later gradually decreasing from 350 to 347.3 range between 200000 and 300000 total operational counts. Comparing to MongoDB at 50000 total operational count, the average read latency is closer to HBase but with the increase in read operation the average read latency is gradually reducing from 160s to 151 micro seconds between 100000 to 300000 total operational count.

Comparatively MongoDB has very low average latency for read operation. Especially for higher read operations than HBase as per average values of the 3 runs over the workloads A, B and D.

7.2 LATENCY OF UPDATE OPERATIONS FOR HBase and MongoDB

Workload A: Update heavy workload: 50/50% Mix of Reads/Writes.

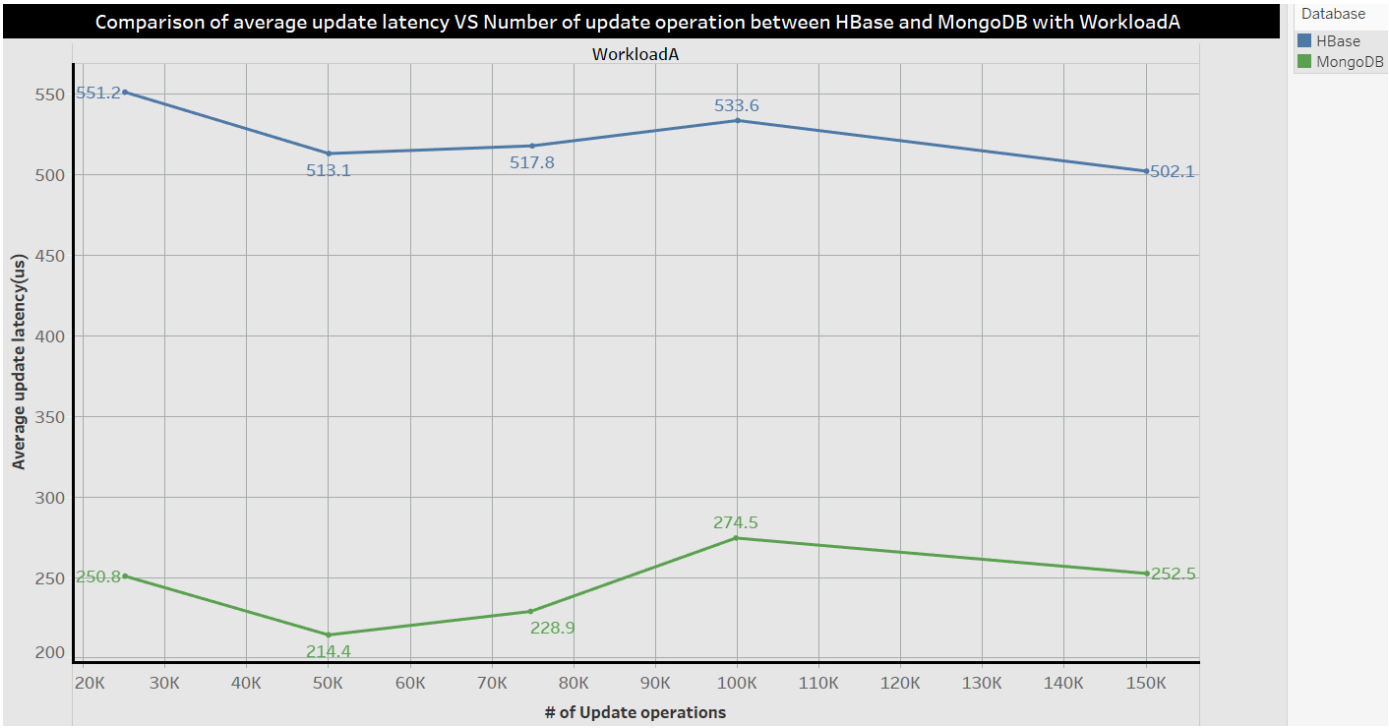

Figure 5: Update average latency VS Total update operations

The Figure 5 represents the Update average latency VS Total update operations between HBase & MongoDB with workload A, Both the database systems perform 50% Update operation of the total operational counts 50,000;100000;150000,200000 and 300000. In case of HBase, the Average update latency gradually decreases from 551.2 microseconds at 25000 update operations to 502.1 microseconds at 150000 update operations. Whereas for MongoDB, the Average update latency fluctuates with the low of 214.4 microseconds at 50000 to the high of 274.5 microseconds at 100000 micro seconds between 25000 to 150000 update operations.

Workload B: Read mostly workload: 95/5% Mix of Reads/Writes.
The Figure 6 illustrates the graphical representation of Update average latency VS Total update operations between HBase & MongoDB with workload B, Both the database systems perform 5% update operation for operational counts. In case of HBase, the Average update latency gradually decreases from 882.8 microseconds at 2500 update operations to 680.6 microseconds at 15000 update operations. Whereas for MongoDB, Average update latency gradually decreases from 306 microseconds at 2500 update operations to 223.4 micro seconds at 15000 update operations.

Comparatively MongoDB has very low latency for update operation than HBase as per average values of the 3 run over the workloads A and B.

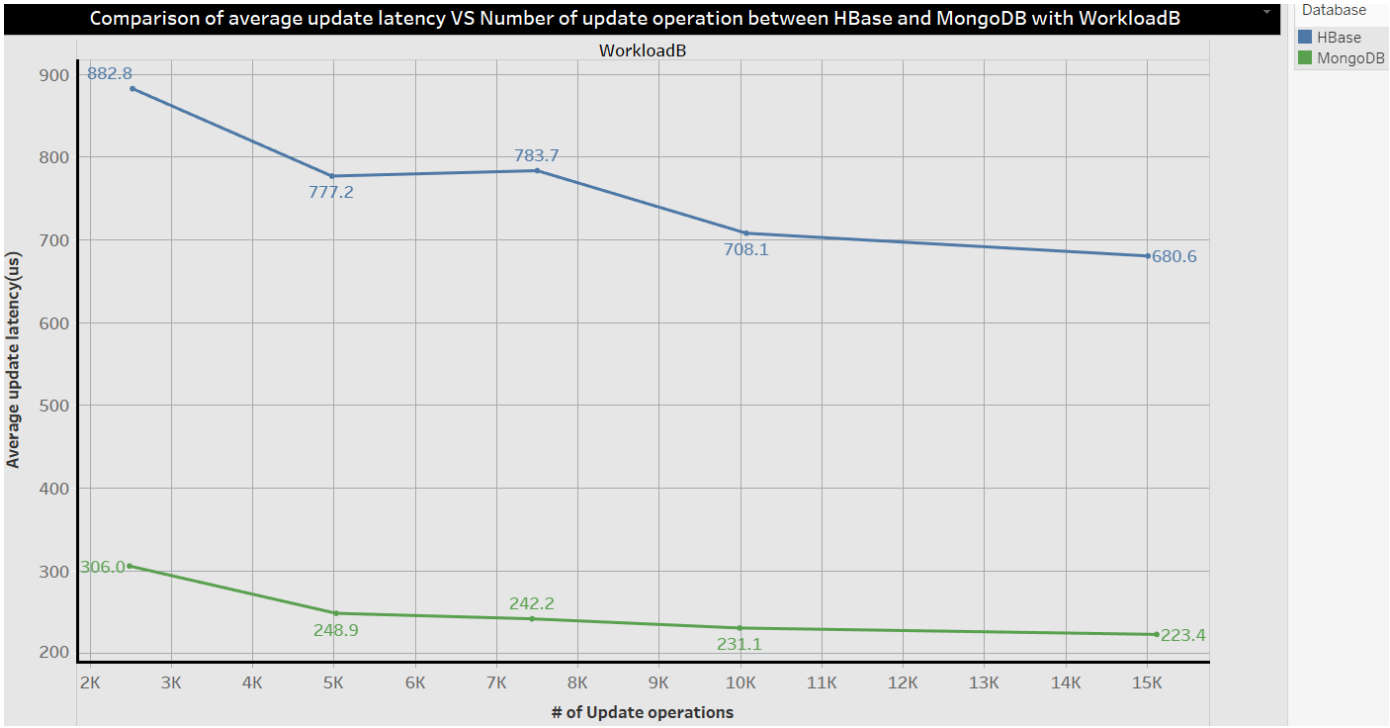

Figure 6: Update average latency VS Total update operations

7.3 LATENCY OF Insert OPERATIONS FOR HBase and MongoDB

Workload D: Read the latest workload: More traffic on recent inserts(95/5% read/insert ratio).

Both the database systems perform 5% Insert operation for operational counts 50,000;100000;150000,200 and 300000. In case of HBase, the Average latency is high as 1006 microseconds for lesser insert operations (2500 insert operations). The graphical representation illustrated in Figure 7 shows that the Value is fluctuating with high and low in the range of 780 to 920 during the total operational counts from 100000 to 300000. MongoDB average latency for insert operation starts with at a high of 267.5 for 2500 insert operation but later has gradual fluctuation ranging between 220 and 185 over the insert operation from 5000 to 15000.

Comparatively MongoDB has low latency rate for insert operation than HBase.

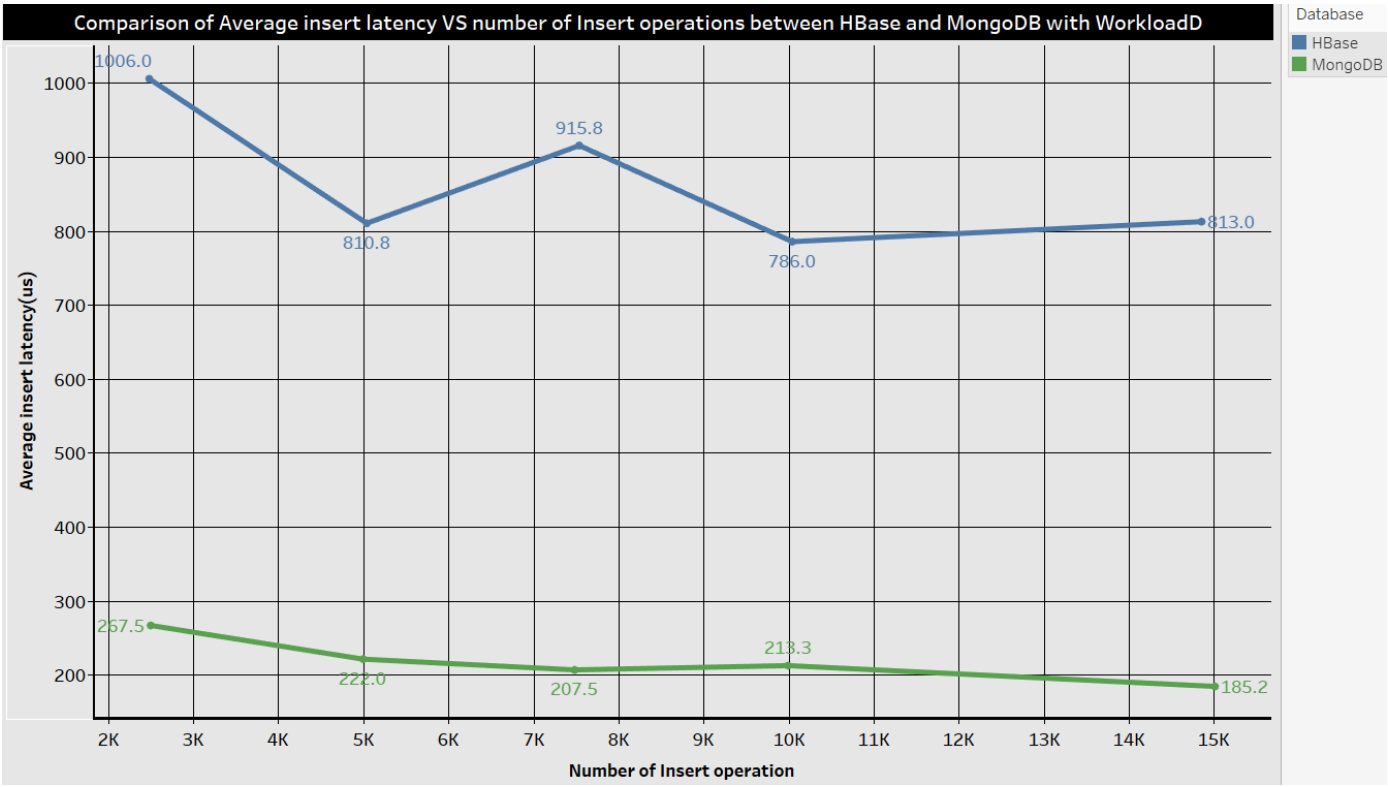

Figure 7: Insert average latency VS Total insert operations

7.4 Comparison of throughput(ops/sec) vs Read average latency(us) between MongoDB and HBase for Workload A (Update heavy workload: 50/50% Mix of Reads/Writes)

The graphical representation illustrated in Figure 8 is plot between the read average latency(us) vs the overall throughput(ops/sec) for number of operations for HBase and MongoDB database with workload A. The values are the average of three individual runs performed with the same environmental setup.

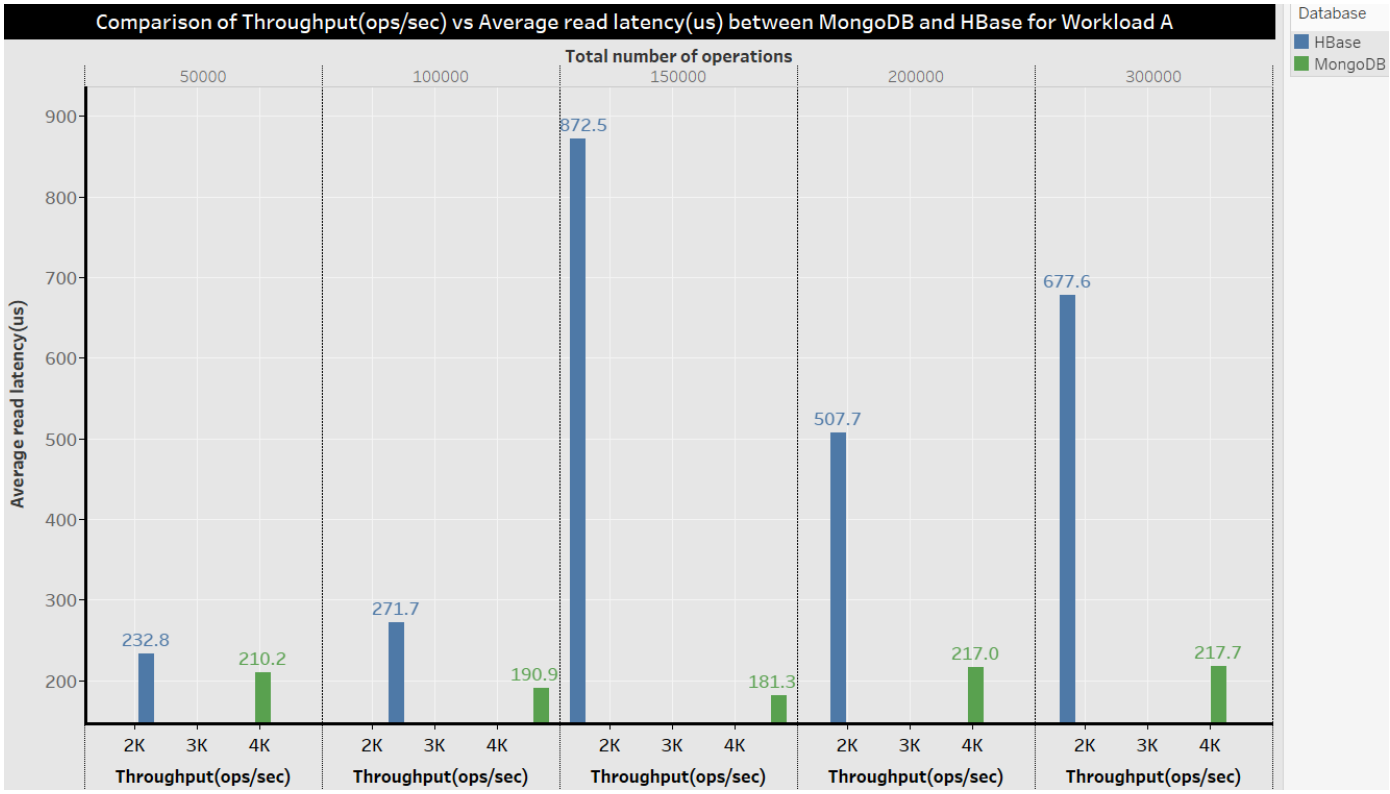

Figure 8: Throughput(ops/sec) vs Read average latency(us)

From the Figure 8, it is inferred that MongoDB performs far better than HBase in read operation as it performs more than 4000 ops/sec throughputs in lesser read latency around 180 to 220 micro seconds across the total operation count range of 50000 to 300000. Whereas HBase throughputs are always lesser than 2500 ops/sec and the read latency rate increases gradually as the total operational count increases.

7.5 Comparison of throughput(ops/sec) vs Read average latency(us) between MongoDB and HBase for Workload B (Read mostly workload: 95/5% Mix of Reads/Writes)

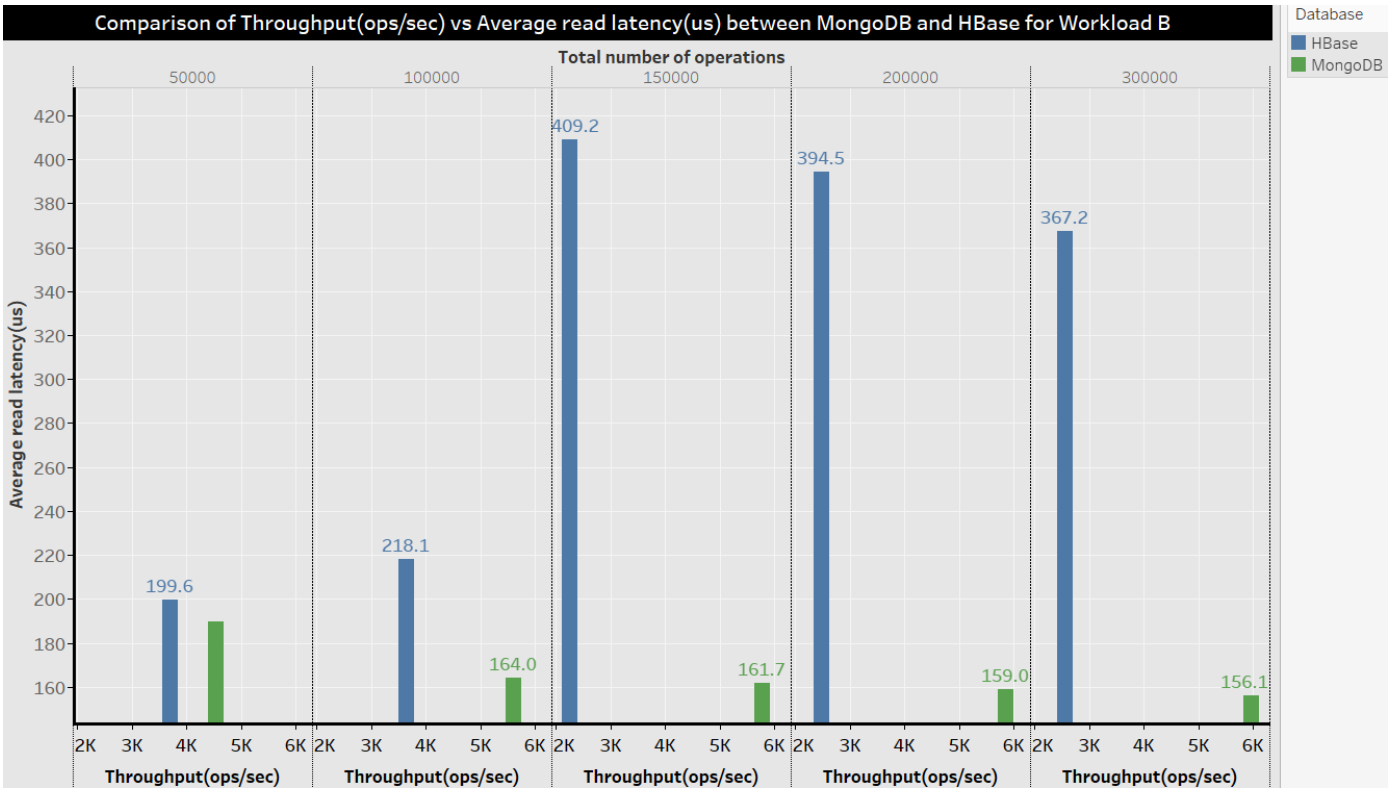

Figure 9: Throughput(ops/sec) vs Read average latency(us)

The values in Figure 9 are the average of three individual runs performed with the same environmental setup and for this load 95% of operational counts are read operation. it is inferred as per the graphical representation that MongoDB performs far better than HBase in read operation as it performs more than 4000 ops/sec throughputs at 50000 total operations (95% of it is read operations) and increases with the growth in read operation. the average read latency is also improving to from 190 micro seconds at 50000 total operations to 156 micro seconds at 300000 total operations. Whereas HBase throughputs are gradually decreasing from around 4000 ops/sec to range between 2000 to 3000 operations per second between 150000 to 300000 total operations. The average latency is also higher for HBase for all read operations compared to MongoDB.

7.6 Comparison of throughput(ops/sec) vs Read average latency(us) between MongoDB and HBase for Workload D: (95/5% read/insert ratio)

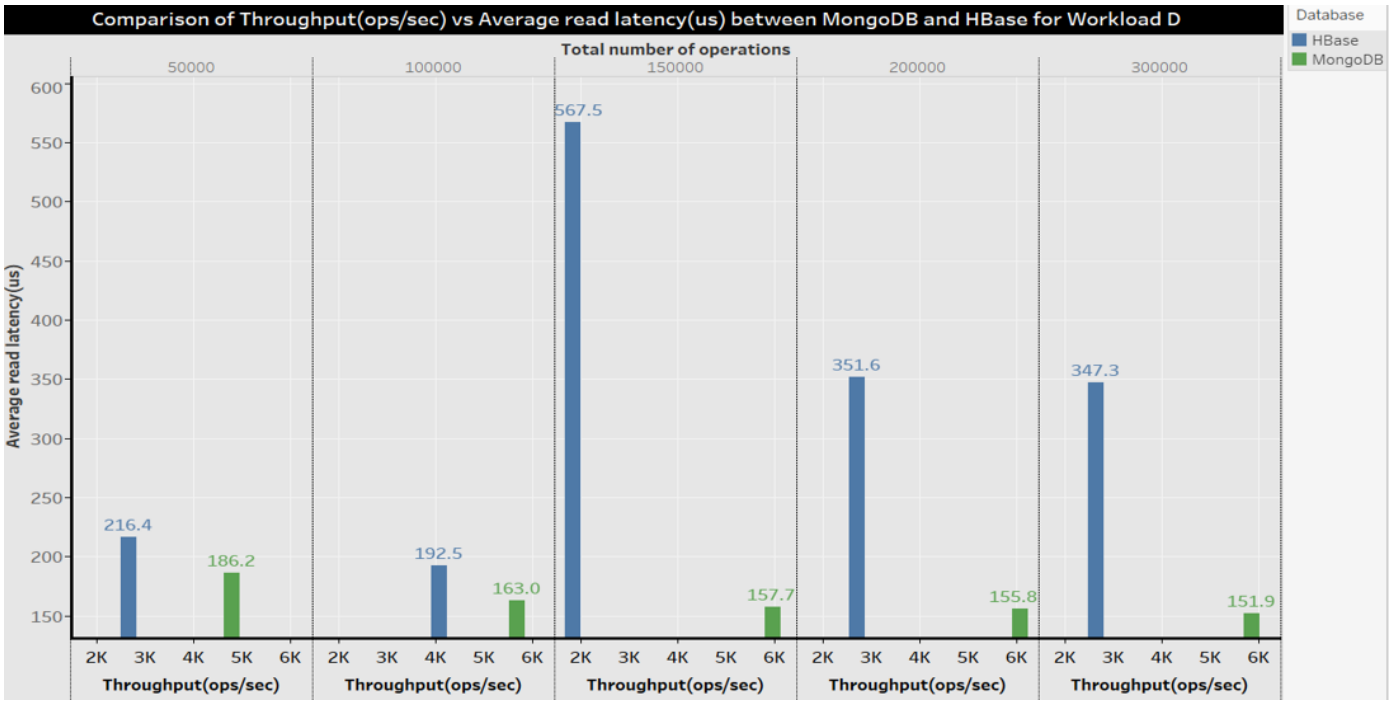

Figure 10: Throughput(ops/sec) vs Read average latency(us)

The graphical representation from the figure shows that the observations are similar to workload B in subsection 7.5. MongoDB performs far better than HBase with workload D in terms of read operations.

7.7 Comparison of throughput(ops/sec) vs Update average latency(us) between MongoDB and HBase for Workload A (Update heavy workload: 50/50% Mix of Reads/Writes)

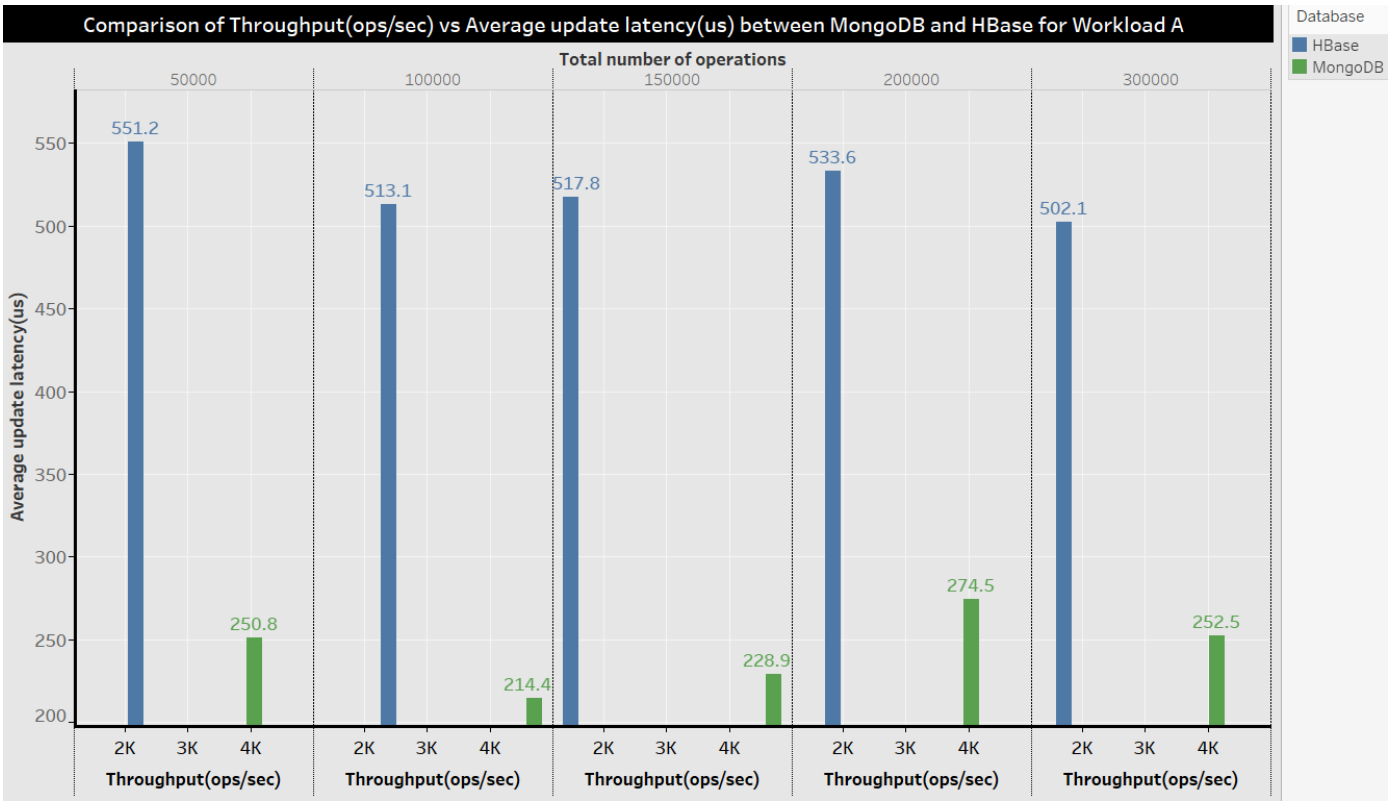

Figure 11: Throughput(ops/sec) vs update average latency(us)

The graphical representation illustrated in Figure 11 is plot between the update average latency(us) vs the overall throughput(ops/sec) for number of operations for HBase and MongoDB database with workloads A. The values are the average of 3 individual runs performed with the same environmental setup. From the Figure 11, it is inferred that MongoDB performs far better than HBase in update operation as it performs around 4000 ops/sec throughputs in lesser update latency fluctuating around the range of 210 to 270 micro seconds across the total operation count in range of 50000 to 300000. Whereas HBase throughputs are always lesser than 2500 ops/sec and the update latency rate fluctuates around the 500-550 micro seconds range.

7.8 Comparison of throughput(ops/sec) vs update average latency(us) between MongoDB and HBase for Workload B (Read mostly workload: 95/5% Mix of Reads/Writes)

Values represented in the graphical representation in Figure 12 are the average of 3 individual runs performed with the same environmental setup and for this load 5% of operational counts are update operation.

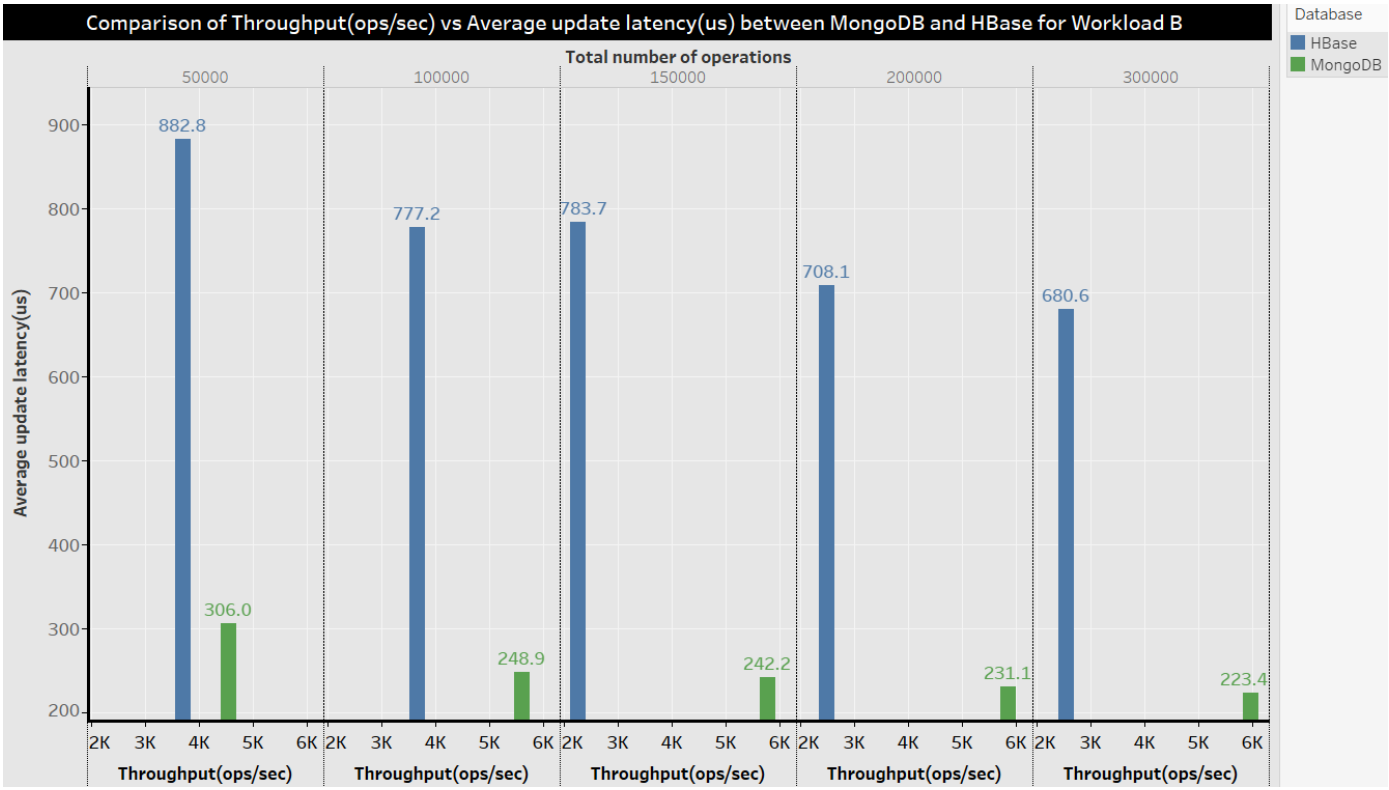

Figure 12: Throughput(ops/sec) vs update average latency(us)

From the Figure 12, it is inferred that MongoDB performs far better than HBase in update operation as it performs more than 4000 ops/sec throughputs at 50000 total operations (5% of it is update operations) and increases with the growth in update operation. the average update latency is also improving to from 306 micro seconds at 50000 total operations to 223 micro seconds at 300000 total operations. Whereas HBase throughputs are gradually decreasing from around 4000 ops/sec to range between 2000 to 3000 operations per second between 150000 to 300000 total operations. The average latency is also higher for HBase for all update operations compared to MongoDB.

7.9 Comparison of throughput(ops/sec) vs insert average latency(us) between MongoDB and HBase for Workload D: (95/5% read/insert ratio)

Values in the graphical representation in Figure 13 are the average of 3 individual runs performed with the same environmental setup and for this load 5% of operational counts are insert operation.

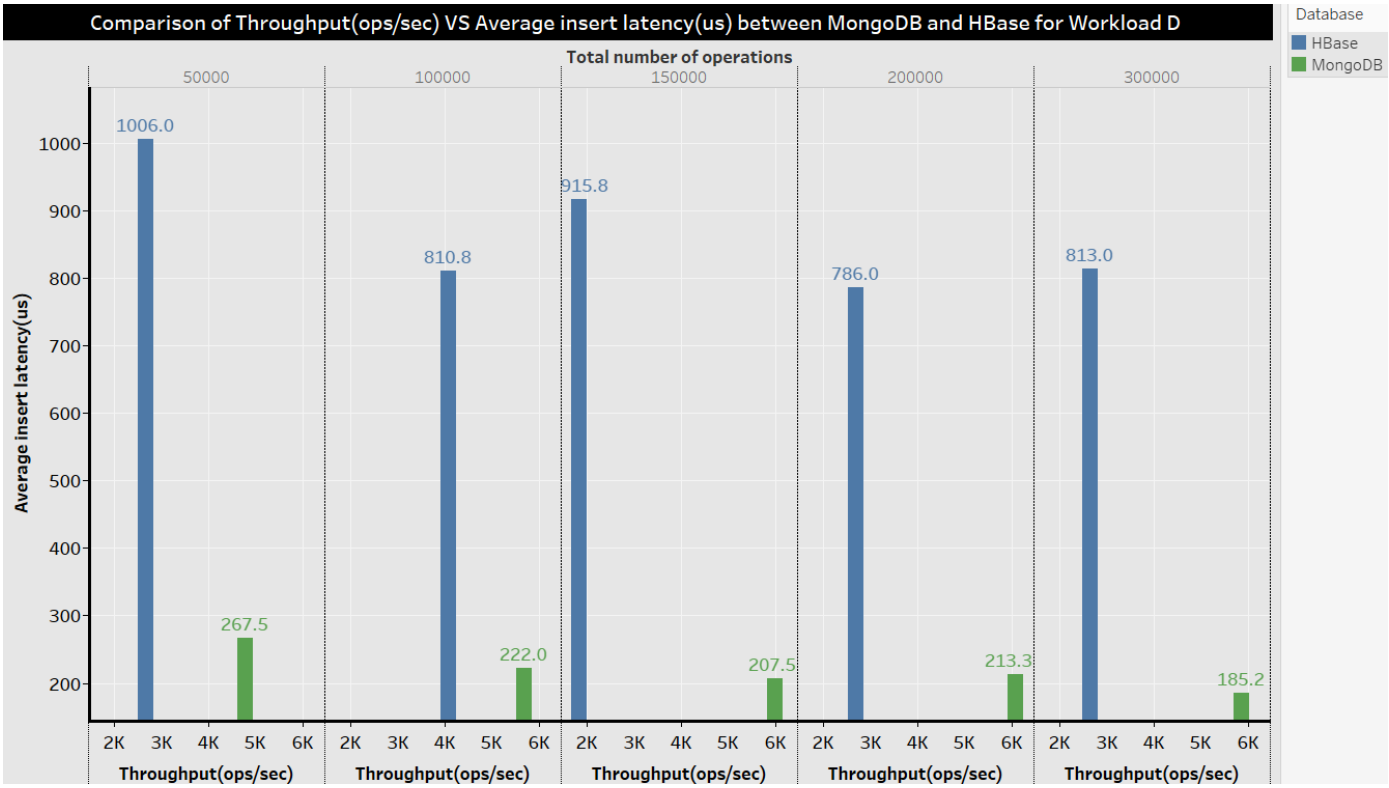

Figure 13: Throughput(ops/sec) vs insert average latency(us)

From Figure 13, it is inferred that MongoDB performs far better than HBase in insert operation as it performs more than 4000 ops/sec throughputs at 50000 total operations (5% insert operations) and increases with the growth in insert operation. the average insert latency is also improving from 267 micro seconds at 50000 total operations to 185 micro seconds at 300000 total operations. Whereas HBase throughputs are fluctuating below 4000 ops/sec to range between 2000 to 3000 operations per second between 150000 to 300000 total operations. The average latency is also higher for HBase for all insert operations compared to MongoDB.

8 Conclusion and discussion

MongoDB Database performed well in all the attributes compared to HBase database in the evaluation performed in this project. The results in the graph in section 7 shows that MongoDB is far better than HBase for the environmental setup it was tested on over the workloads A, B and D in terms of performing at low latency rate and high through put with the increase in operational count. From the research done in section 5, it is expected for MongoDB to outclass HBase in read operations. But it has performed better than HBase on update and insert operation which is contradictory as per the previous research.Abramova et al. (2014*a*). Possible reason for this behaviour may be the operational load run on the database was lesser as the operations were only 50000,100000,150000,200000 and 300000 total operation and the update operations measured were only 50% of update operation for workload A and 5% of update operation for workload B and 5% of insert operation for workload D. this might have not done justice to HBase update operation capabilities. Future work should be considered to perform the

evaluation of these databases against loads with higher update percentage and In heavier total operational counts such that both the databases have an even environmental setup to perform its functions.

References

Abramova, V., Bernardino, J. & Furtado, P. (2014*a*), 'Experimental evaluation of nosql databases', *International Journal of Database Management Systems* **6**(3), 1.

Abramova, V., Bernardino, J. & Furtado, P. (2014*b*), 'Which nosql database? a performance overview', *Open Journal of Databases (OJDB)* **1**(2), 17–24.

dezyre.com (2016), 'overview-of-hbase-architecture-and-its-components'.
URL: *https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295*

George, L. (2011), *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*, O'Reilly.
URL: *https://www.dawsonera.com:443/abstract/9781449315771*

intellipaat.com (2018), 'What is mongodb?'.
URL: *https://intellipaat.com/blog/what-is-mongodb/*

Leite, N. (2015), 'Mongodb architecture'.
URL: *https://www.slideshare.net/NorbertoLeite/mongodb-internals-55965341*

mediawiki (2018), 'Hbase security features'.
URL: *https://quabase.sei.cmu.edu/mediawiki/index.php/HBase*$_S$*ecurity*$_F$*eatures*

MongoDB (2015), 'Mongodb$_a$*rchitecture*$_g$*uide*'.
URL:*https://jira.mongodb.org/secure/attachment/112939/MongoDB*$_A$*rchitecture*$_G$*uide.pdfMongoI*

Murphy, D. (2017), 'The essential guide to mongodb security'.
URL: *https://www.infoworld.com/article/3164504/security/the-essential-guide-to-mongodb-security.html*

Sinha, S. (2018), 'Hbase tutorial: Hbase introduction and facebook case study'.
URL: *https://www.edureka.co/blog/hbase-tutorial*

Swaminathan, S. N. & Elmasri, R. (2016), Quantitative analysis of scalable nosql databases, *in* '2016 IEEE International Congress on Big Data (BigData Congress)', pp. 323–326.

Tang, E. & Fan, Y. (2016), Performance comparison between five nosql databases, *in* '2016 7th International Conference on Cloud Computing and Big Data (CCBD)', pp. 105–109.

Tudorica, B. G. & Bucur, C. (2011), A comparison between several nosql databases with comments and notes, *in* '2011 RoEduNet International Conference 10th Edition: Networking in Education and Research', pp. 1–5.