

PERFORMANCE EVALUATION OF NoSQL DATABASES

Quantitative analysis of MongoDB and Cassandra

Tejas Phopale

Student Id: x17162301, Email: x17162301@student.ncirl.ie

DSM – Project B

INTRODUCTION

NoSQL databases has turned out to be essential and generally adopted platforms in cloud. NoSQL has great characteristics like the ability to manage huge volume of data, schema independent data model and horizontal scalability makes peremptory request. Every industry demands for large volume of data which requires data management system to be capable not only fetch the data in time scale but also to grow data storage potential. There are many Bigdata applications like Google's BigTable, Amazon's Dynamo, Cassandra, Oracle's NoSQL DB, MongoDB and Apache's HBase (Băzăr et al.,2014) has driven the advancement which shows significance image to handle big data. The horizontal scalability (Okman et al., 2011) is one of the principle feature of NoSQL databases which helps to increase in number of machines on existing clusters.









Type	Example	
Key-Value Store	 redis	 riak
Wide Column Store	 H-BASE	 cassandra
Document Store	 mongoDB	 CouchDB relax
Graph Store	 Neo4j	 InfiniteGraph The Distributed Graph Database

Figure 1: Types of NOSQL databases (Source:
<https://www.pinterest.ie/pin/536632111831770892/>)

In this report, we focused to evaluate performance by doing the quantitative analysis of MongoDB (document based) and Cassandra (key value based) (Klein et al.,2015) on its features with CRUD operations, operation counts and scalability with various dataset sizes. Nowadays, the popular benchmarking tool is “Yahoo! Cloud Service Benchmark (YCSB)” (Abramova et al.,2013) which has utilized for our current experimentation and analysis and we have compared them in term of key characteristics, database architectures and security.

OVERVIEW

MongoDB is leading open source, document-oriented database which has MapReduce as query language and built in C++. MongoDB stores JSON-like documents and objective are filling gaps between key-value stores. It has replica set and master-slave replication with sharding partitioning (Investigation and Comparison of Distributed NoSQL Database Systems, n.d.).

Apache Cassandra is a leading open-source built in Java technology with MapReduce as query language for Hadoop integration. It creates databases which spread across the nodes for more than one data source due to high Scalability and availability based on Google BigTable and Amazon Dynamo (Investigation and Comparison of Distributed NoSQL Database Systems, n.d.). It has selectable replication factor and sharding partitioning with secondary indexing which provides high write throughput.

KEY CHARACTERISTICS

The key characteristics of MongoDB and Cassandra are compared in terms of the following parameters:

1) **Object Model:**

MongoDB is highly document-oriented model in which multiple objects can able to nest into multiple levels and its object-oriented modelling (OOM) can be elaborate in any type of business domain.

Cassandra in other hand, consist traditional structure format data with rows and columns which is defines during development.

2) **High Availability:**

MongoDB defines “master-slave” replication model wherein one master node controls many slave nodes. If master goes off then it selects any one slave as master which takes around few seconds to recover. It creates its drawback as write operations gets stop during this timeline.

But in Cassandra, it persists selectable replication factor i.e. multi-master format which helps to continue write operations if any node goes off. The uptime for write operation is 100% (Source: <https://scalegrid.io/blog/cassandra-vs-mongodb/>).

3) **Write Scalability:**

The MongoDB “master-slave” replication format allows master i.e. primary node to write first and keeps other node for read operation.

In Cassandra, it increases number of servers in cluster with scalability and allows write operation to any of the server due to multiple-master replication factor

(Source: <https://scalegrid.io/blog/cassandra-vs-mongodb/>).

4) **Query Language:**

MongoDB structured with JSON fragments with MapReduce as integration language with Hadoop.

Cassandra structured with CQL which behaves similarly as SQL (Hossen et al., 2017).

5) **Brewer’s CAP Theorem:**

MongoDB follows CP i.e. Consistency and Partition Tolerance whereas, Cassandra is AP i.e. Availability and Partition Tolerance on CAP theorem (Manoj et al., 2014).

6) **Schema dependency:**

MongoDB is schema independent model in which no mandate schema is getting enforced on documents stores.

Cassandra follows CQL query language so each column need to define upfront (Abramova et al.,2013).

Features	Cassandra	MongoDB
Description	Wide-column store based on BigTable & DynamoDB	One of the most popular doc stores
Implementation Language	Java	C++
Database Model	Wide column store	Document store
Schema	Schema-free	Schema-free
SQL	No	No
APIs and access methods	Proprietary protocol	Proprietary protocol
Languages supported	Many	Too Many
triggers	No	No
Replication Model	Selectable replication factor	Master/Slave
MapReduce	Yes	Yes
Consistency Model	Eventual or Immediate	Eventual or Immediate
Foreign Key support	No	No
Concurrency support	Yes	Yes

Table 1: Few key characteristics of MongoDB & Cassandra (Source: <https://www.wikitechy.com/tutorials/couchdb/couchdb-tutorial>)

DATABASE ARCHITECTURE

The database architecture has been described on the following parameters:

1) **Topology:**

There are different types of topologies and each lead to different level of availability and scalability. The MongoDB provides master-slave replication as described above so if master goes off they slave node takeover its place and during this time write operations hold for some seconds. It has sharded cluster which houses metadata which distributes the requests wherein Cassandra is with no master and each node has a unique token which groups several machines into a framework of data centres. It has snitch which work as informer to each node. So, MongoDB is less clustered than Cassandra (Srinivas et al., 2015).

2) **Consistency:**

MongoDB's primary node is fully consistent that mean it provides both read and write operation but secondary node is partial consistent because it provides only read service. The Cassandra is fully consistent by all means due to no master-slave structure. It determines up to date raw data and synchronized with others (Source: <https://scalegrid.io/blog/cassandra-vs-mongodb/>) (Srinivas et al., 2015).

3) **Fault Tolerance:**

MongoDB is less fault tolerant than Cassandra due to master-slave architecture as it becomes single point of failure if configurations are unavailable. The Cassandra has practitioner which uses MD5 hash which is eventually distributed across the cluster (Srinivas et al., 2015).

4) **Storage:**

MongoDB stores everything in JSON file format which stores as document and provides extra space to grow as per updates. New record is being added if document exceeds the threshold limit (Zahid et al., 2014). The Cassandra has rows, tables and required partition key with column-oriented storage approach. It also has secondary indexes but in hidden table. It improves latency & read throughput if cache is enables and after row is being inserted into a table it enters logs in Memtable (Investigation and Comparison of Distributed NoSQL Database Systems, n.d.) (Zahid et al., 2014).

SECURITY

The following security features based on parameters:

1) **Authentication:**

MongoDB uses its own open source authenticating version called MongoDB-CR which secures both cluster and intra-cluster communication with SSL (X.509) certificate.

Cassandra uses MD5 for storing password based authentication for intra node communication between users and clusters so there might be chances of data getting tampered or stolen because user bypasses the client side authentication but its secure by SSL/TSL certificate (Source:

<https://www.mongodb.com/blog/post/atrest-encryption-in-mongodb-3-2-features-and-performance>).

2) **Access Controls:**

The MongoDB has base system defined role which can be customize as well but within the sharded cluster (Source: <https://www.mongodb.com/blog/post/atrest-encryption-in-mongodb-3-2-features-and-performance>).

The role wise hierarchy is maintained in Cassandra for access control by administrator. By using create, alter or drop methods in CQL query language, administrator can able to generate "Role Based Access Control" (RBAC) (Okman et al., 2011).

Both database provides per-database access authentication for read, update, create and drop objects (Okman et al., 2011).

3) **Transaction Control:**

The transaction control can be maintained by Consistency and Concurrency at both physically and logically level.

In MongoDB, suppose a query is incorrectly manipulated then data is sent to every slave nodes asynchronously. Data is sent out to all slave nodes asynchronously and it allows update transactions to all the nodes together so MongoDB offers eventual consistency with no concurrency control.

The Cassandra provides MVCC stands for multi-version concurrency control. As per the business requirement few consistent read-write operations can able to configure in it (Okman et al., 2011).

4) **Encryption of Data:**

The MongoDB provides data encryption using SSL at transport level (Okman et al., 2011). The new 3.2 version of MongoDB has new feature called "at-rest encryption" (Source: <https://www.mongodb.com/blog/post/atrest-encryption-in-mongodb-3-2-features-and-performance>). which states that data getting stored at persistent storage i.e. disk or tape and process by CPU or RAM.

The Cassandra has optional transparent data level encryption for security which only secure disk data by encrypting it. The Secure Socket Layer is install for internal node and internal framework communication. The disadvantage is that the encryption

certificate has been stored locally so it can be hack easily. The motion data is not secured with SSL (Okman et al., 2011).

5) **SQL Injections:**

In MongoDB, the developer should implement security injection techniques/methods while writing the code because of server-side programming methodology.

The Cassandra has CQL query language wherein precaution might need to be taken while developing to secure from SQL injections (Source:

<https://www.mongodb.com/blog/post/atrest-encryption-in-mongodb-3-2-features-and-performance>).

6) **Audit:**

The Audit is useful for maintaining the actions by users and can track if any issue occurs during workflow.

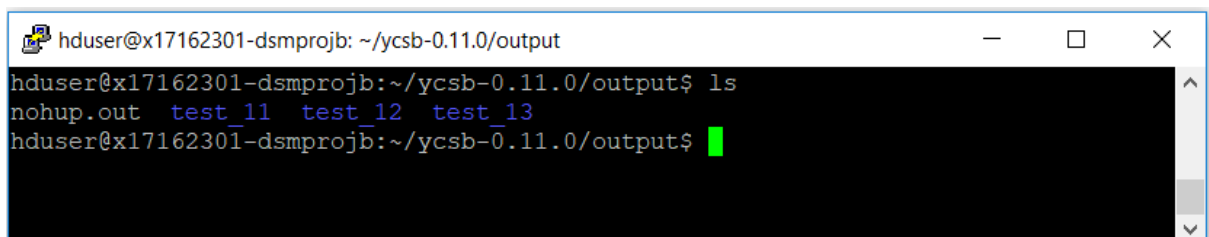
MongoDB offers some basic options for auditing in syslog, BSON or JSON file.

Cassandra do not provide any tool or marker wherein auditing can able to track.

To summarize, Cassandra shows great achievement in terms of authorisation and encryption however we may get new featured version to handle injection control, auditing, consistency etc (Okman et al., 2011).

PERFORMANCE TEST PLAN

In our experiment, evaluation done on MongoDB and Cassandra databases. We have used most popular YCSB (Yahoo! Cloud Serving Benchmark) benchmark. The evaluation has been differentiated in two parts: one is for workloads generator (we have used two workloads as A and B) and other is for different range of workloads (range of 25000,5000,7500 and 95000). These workloads are with read, write and update operations and we are going to measure based on average latency and overall throughputs. We have two workloads named as “workloada” and “workloadb”. Each has different significance of heaviness for read and update operations. Total three tests are performed named “test_11”, “test_12” and “test_13” (below figure 2).



```
hduser@x17162301-dsmprojb: ~/ycsb-0.11.0/output
hduser@x17162301-dsmprojb:~/ycsb-0.11.0/output$ ls
nohup.out  test_11  test_12  test_13
hduser@x17162301-dsmprojb:~/ycsb-0.11.0/output$
```

Figure 2: Output tests

The version of YCSB used is 0.11.0. The experiment was conducted in Ubuntu 16.04.3 LTS. The “workloada” has Read/Update ratio: 50/50 with 1000-1000 record and operation count respectively and zero scan and insert proportion (below figure 3).

```
# Yahoo! Cloud System Benchmark
# Workload A: Update heavy workload
#   Application example: Session store recording recent actions
#
#   Read/update ratio: 50/50
#   Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
#   Request distribution: zipfian

recordcount=1000
operationcount=1000
workload=com.yahoo.ycsb.workloads.CoreWorkload

readallfields=true

readproportion=0.5
updateproportion=0.5
scanproportion=0
insertproportion=0

requestdistribution=zipfian

hduser@x17162301-dsmprojb:~/ycsb-0.11.0/workloads$
```

Figure 3: “workloada” configurations

The “workloadb” has Read/Update ratio: 95/5 with 1000-1000 record and operation count respectively and zero scan and insert proportion (below figure 4).

```
hduser@x17162301-dsmprojb: ~/ycsb-0.11.0/workloads

# Yahoo! Cloud System Benchmark
# Workload B: Read mostly workload
#   Application example: photo tagging; add a tag is an update, but most operations are to read tags
#
#   Read/update ratio: 95/5
#   Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
#   Request distribution: zipfian

recordcount=1000
operationcount=1000
workload=com.yahoo.ycsb.workloads.CoreWorkload

readallfields=true

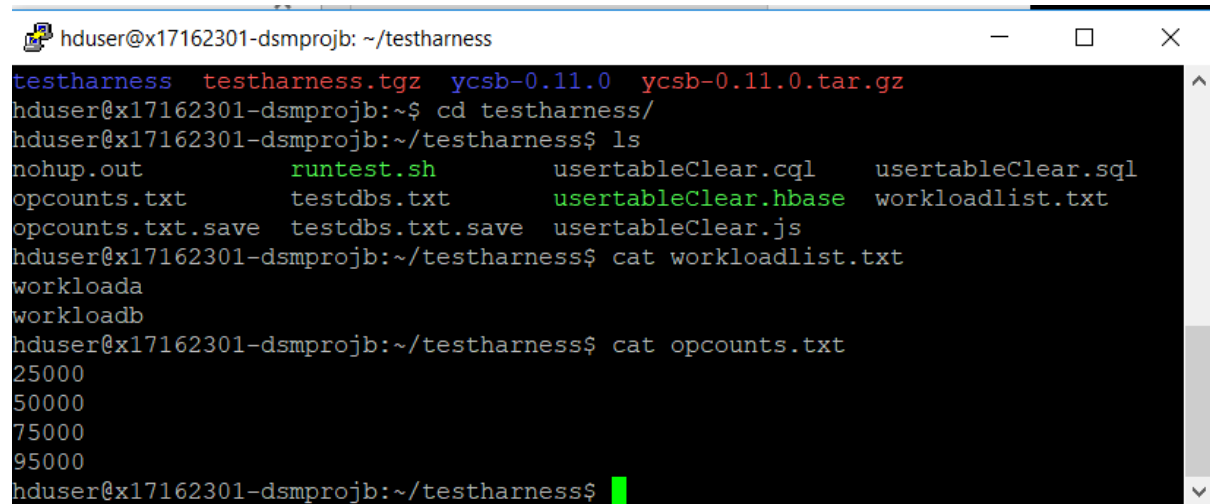
readproportion=0.95
updateproportion=0.05
scanproportion=0
insertproportion=0

requestdistribution=zipfian

hduser@x17162301-dsmprojb:~/ycsb-0.11.0/workloads$
```

Figure 4: “workloadb” configurations

The Cassandra and MongoDB were evaluated in this experiment. The single replication has been used for MongoDB and Cassandra to run in the foreground and background respectively. The number of operations are 25000, 50000, 75000 and 95000. Using Zipfian strategy, those are executed with same identical sized dataset. The metrics which needs to evaluate the overall throughput and overall runtime for all the workloads. The run time and performance are indirectly proportional to each other whereas, the throughput and performance are directly proportional to each other.

A terminal window titled 'hduser@x17162301-dsmprojb: ~/testharness' with standard window controls. The terminal shows the following commands and output:

```
testharness testharness.tgz ycsb-0.11.0 ycsb-0.11.0.tar.gz
hduser@x17162301-dsmprojb:~$ cd testharness/
hduser@x17162301-dsmprojb:~/testharness$ ls
nohup.out          runtest.sh         usertableClear.cql  usertableClear.sql
opcounts.txt       testdb.txt         usertableClear.hbase workloadlist.txt
opcounts.txt.save  testdb.txt.save    usertableClear.js
hduser@x17162301-dsmprojb:~/testharness$ cat workloadlist.txt
workloada
workloadb
hduser@x17162301-dsmprojb:~/testharness$ cat opcounts.txt
25000
50000
75000
95000
hduser@x17162301-dsmprojb:~/testharness$
```

Figure 5: Operation Counts

EVALUATION AND RESULTS

The output files have been collected from different three the test and manipulated the below summary parameter by taking average of all the parameters from all the different test sets

The Output Summary for testharness with Workload A and Workload B:

Database	Overall		No. of Operations			Average Latency	
	Runtime(Ms)	Throughput (ops/sec)	Read	Update	Total	Read(Us)	Update(Us)
Cassandra	19628.66	1301.48	12490	12510	25000	605.83	515.68
MongoDB	9789	2551	12525	12475	25000	145.97	581.7
Cassandra	32270	1550.57	24951	25049	50000	597.68	424.68
MongoDB	26623.7	1875.33	24968	25032	50000	141.16	882.4
Cassandra	43224	1734.35	37398	37602	75000	582.83	388.36
MongoDB	63655	1174.189	37450	37550	75000	127.508	1546.533
Cassandra	50966.66	1864	47375	47625	95000	546.65	376.51
MongoDB	24673	3850.36	47426	47574	95000	121.26	372.95665

Figure 6: Output statistics of workload A

Database	Overall		No. of Operations			Average Latency	
	Runtime(Ms)	Throughput (ops/sec)	Read	Update	Total	Read(Us)	Update(Us)
Cassandra	19241.5	1296.3	23771	1229	25000	512.8	544.9
MongoDB	10304.9	2423.83	23789	1211	25000	153.79	4797.483
Cassandra	33773.6	1484.04	47533	2467	50000	543.673	467.8
MongoDB	11295.9	4424.2	47529	2471	50000	119.5	1995.2
Cassandra	48205.7	1554.75	71253	3747	75000	559.8	469.3
MongoDB	16353.3	4583.63	71191	3809	75000	120.44	1844.8
Cassandra	55295.4	1715.9	90249	4751	95000	515.4	437.52
MongoDB	18139.3	5237.32	90155	4845	95000	112.5	1495.1

Figure 7: Output statistics of workload B

As shown in above figure, the summarisation done by categorising in two workloads. The “workloada” is alias to Workload A and “workloadb” is alias to Workload B.

1) Workload A:

- Average Latency Vs Read Operations:** This evaluation is done for workload A compared by avg. latency Vs Read Operations wherein we can find that Cassandra shows more latency than MongoDB and both the database average latency is getting decreases as total no. of operations increases.

Workload A : **Average Latency Vs Read Operations**

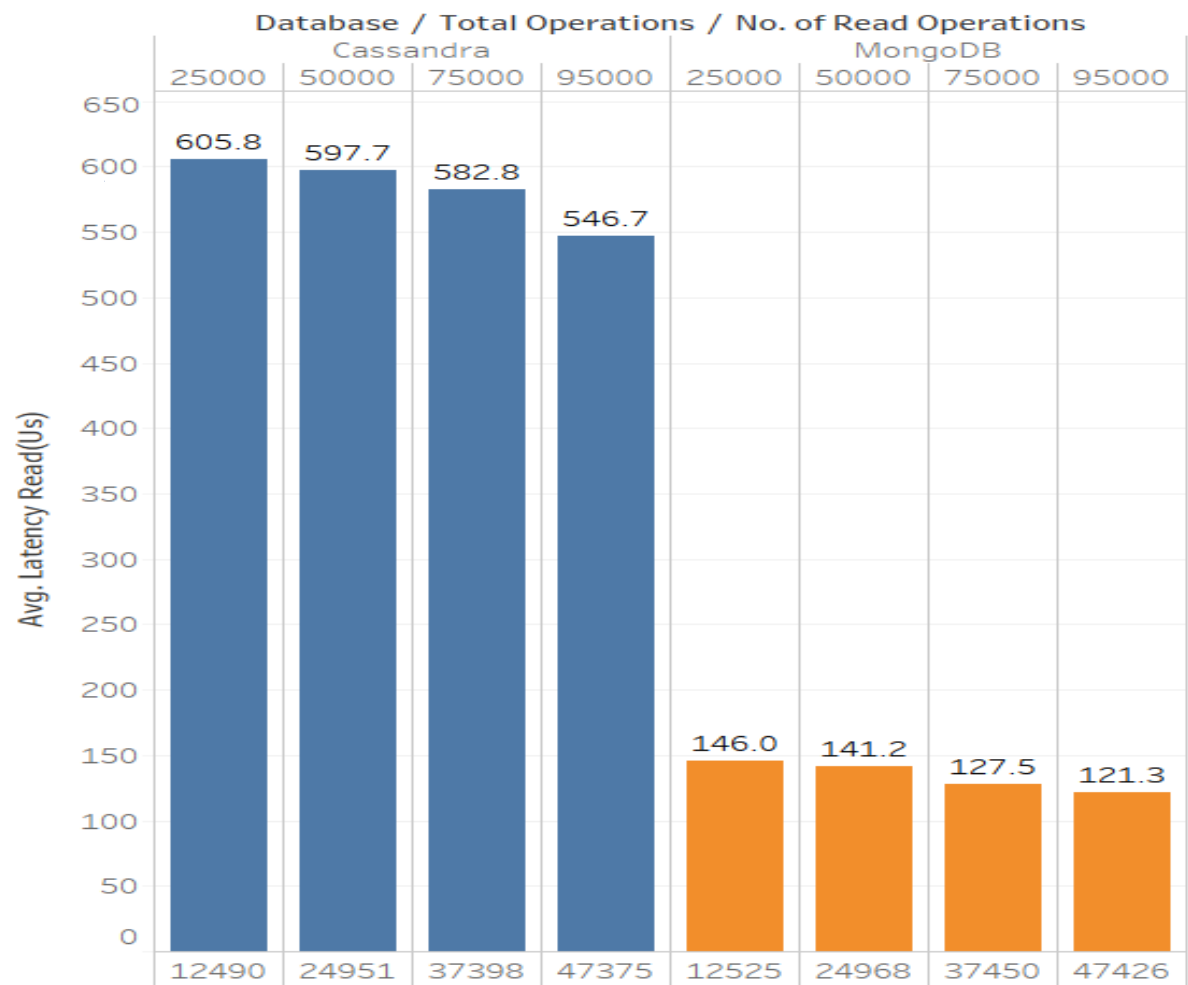


Figure 8: Workload A: Average Latency Vs Read Operations

- b. **Average Latency Vs Update Operations:** MongoDB shows better performance than Cassandra with update operation but up to some threshold limit. From below graph we can observe that average latency update suddenly decreased and shows almost response at 95000 number of operations.

Workload A : **Average Latency Vs Update Operations**

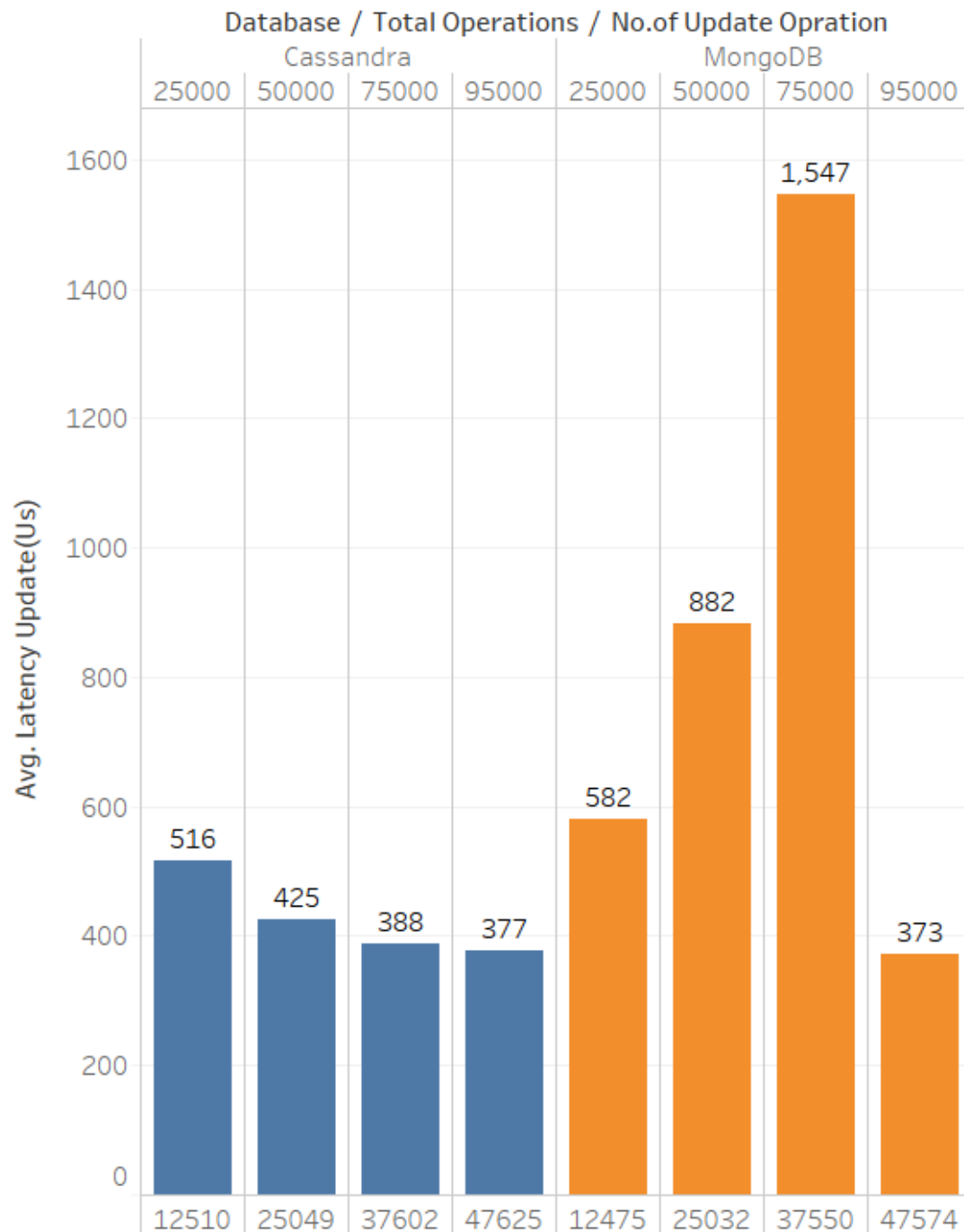


Figure 9: Workload A: Average Latency Vs Update Operations

- c. **Overall throughput vs Record operations:** If we compare the overall throughput with total operations at workload A, the graph indicates the MongoDB has high overall throughput except at 75000 number of operations. The MongoDB shows almost 30% raise in overall throughput from 25000 workloads to 95000 workloads.

Workload A : Overall Throughput Vs Record Operations

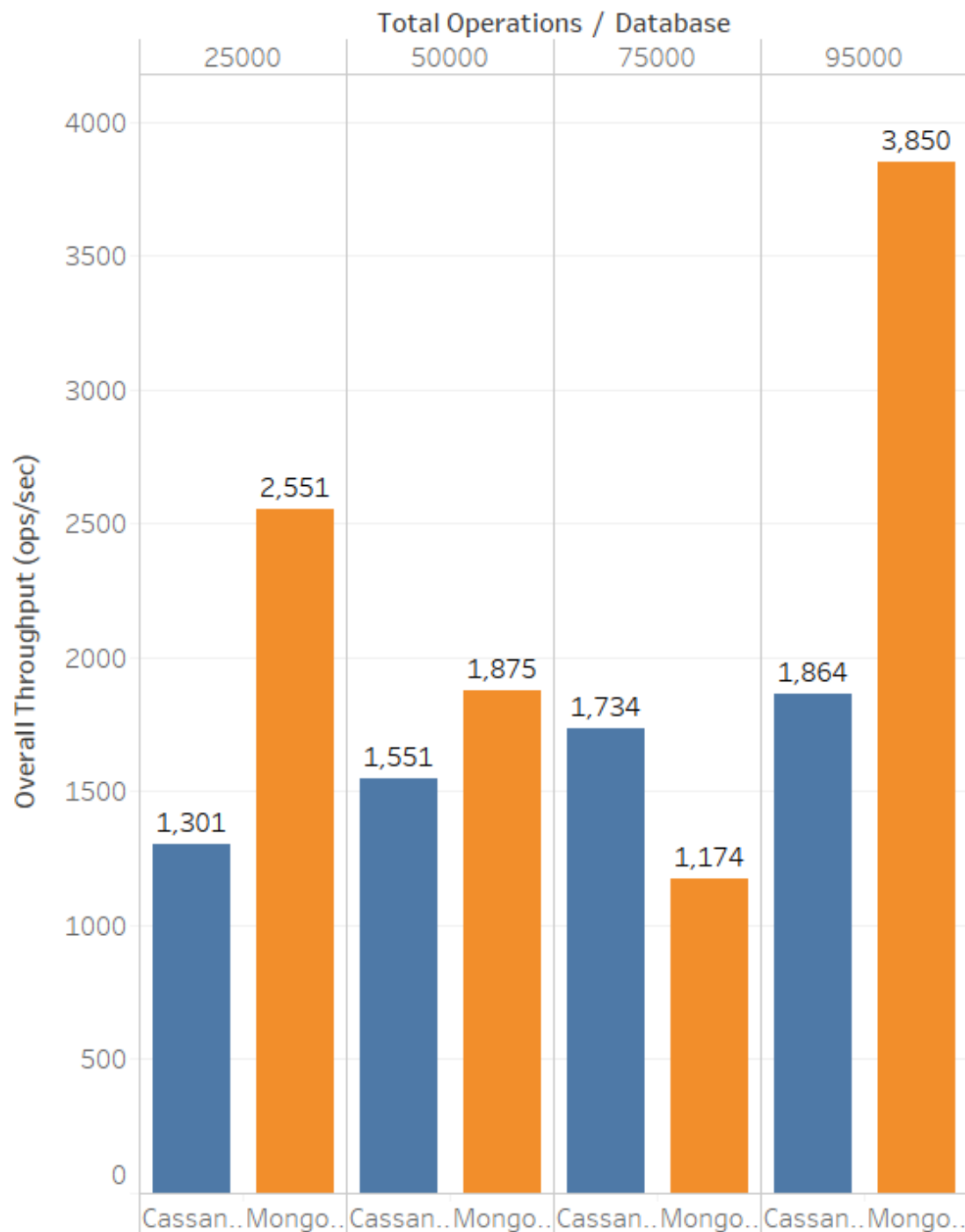


Figure 10: Workload A: Overall Throughput Vs Total Record Operations

2) Workload B:

- a. **Average Latency Vs Read Operations:** If we compare the below graph with workload A figure 7 we could find that the Cassandra shows similar behaviour at both the workload by showing more average latency than MongoDB at both the workloads.

Workload B : Average Latency Vs Read Operations

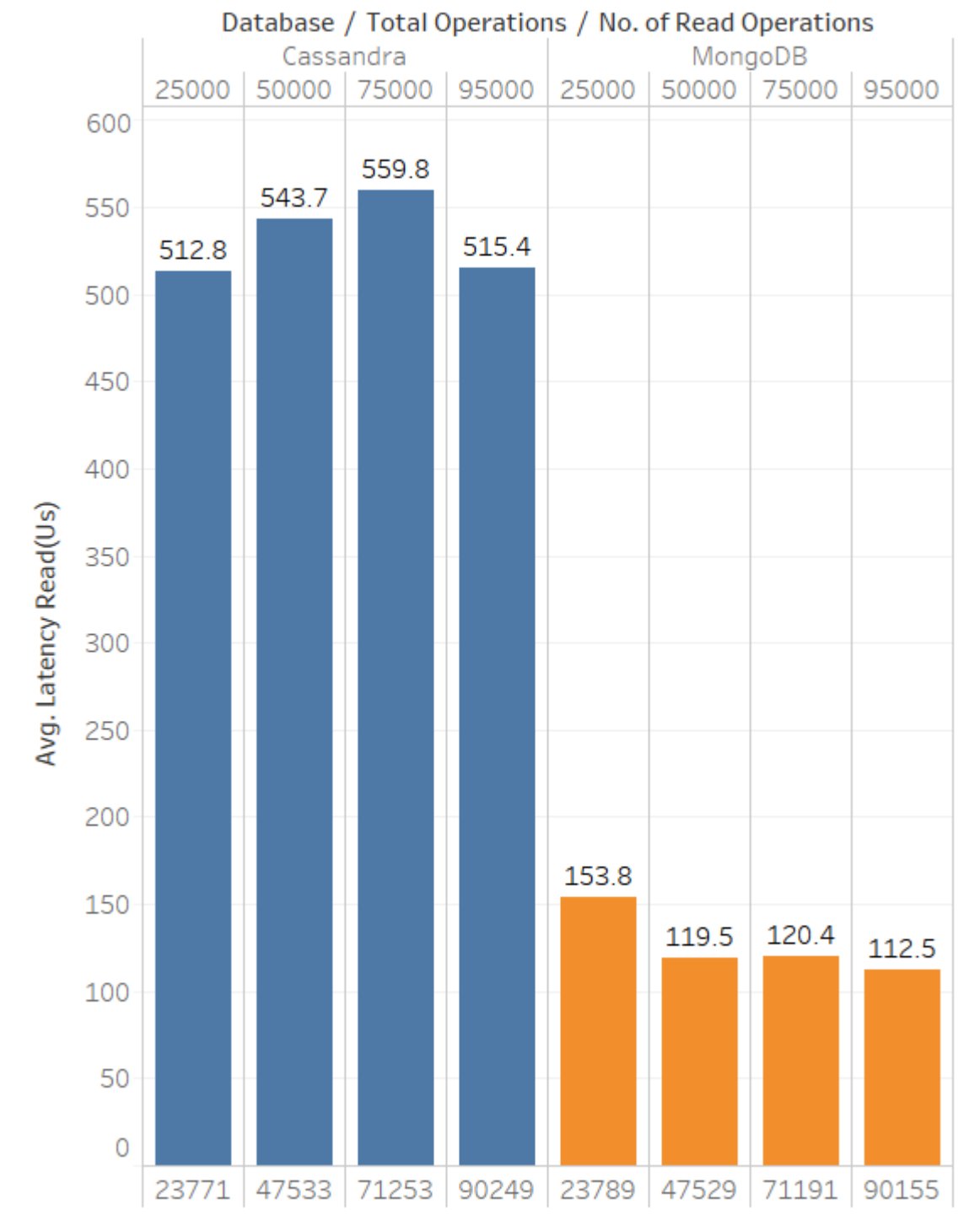


Figure 11: Workload B: Average Latency Vs Read Operations

- b. **Average Latency Vs Update Operations:** The figure 11 shows that as the number of workloads increases then MongoDB reduces the update latency but still Cassandra shows better performance at every number of workload.

Workload B : **Average Latency Vs Update Operations**

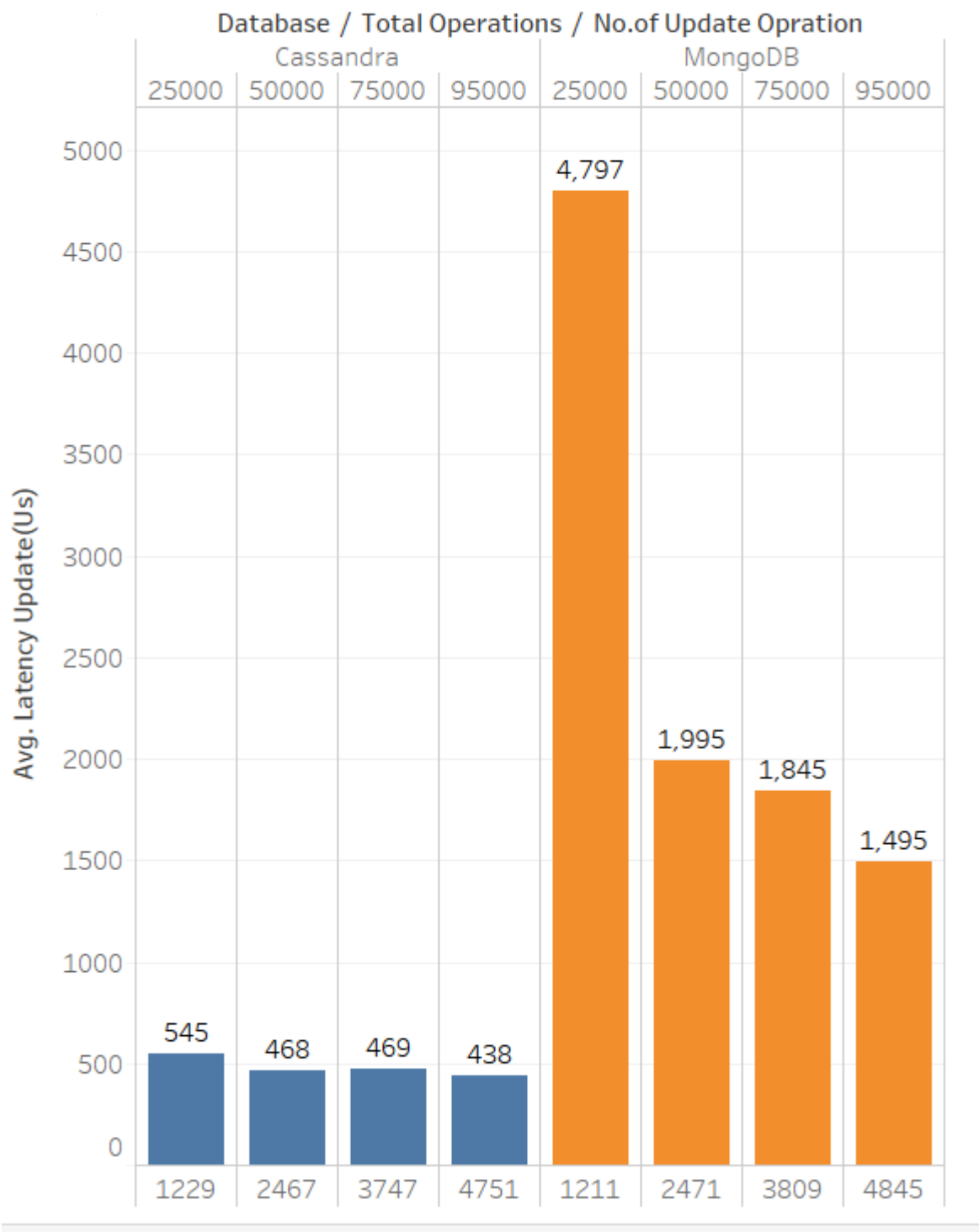


Figure 12: Workload B: Average Latency Vs Update Operations

- c. **Overall throughput vs Record operations:** The MongoDB shows better performance at overall throughput than Cassandra. Both databases show increase in throughput as number of workloads increases. The MongoDB overall throughput increase by 50% from 2500 workloads to 95000 workloads.

Workload B : Overall Throughput Vs Record Operations

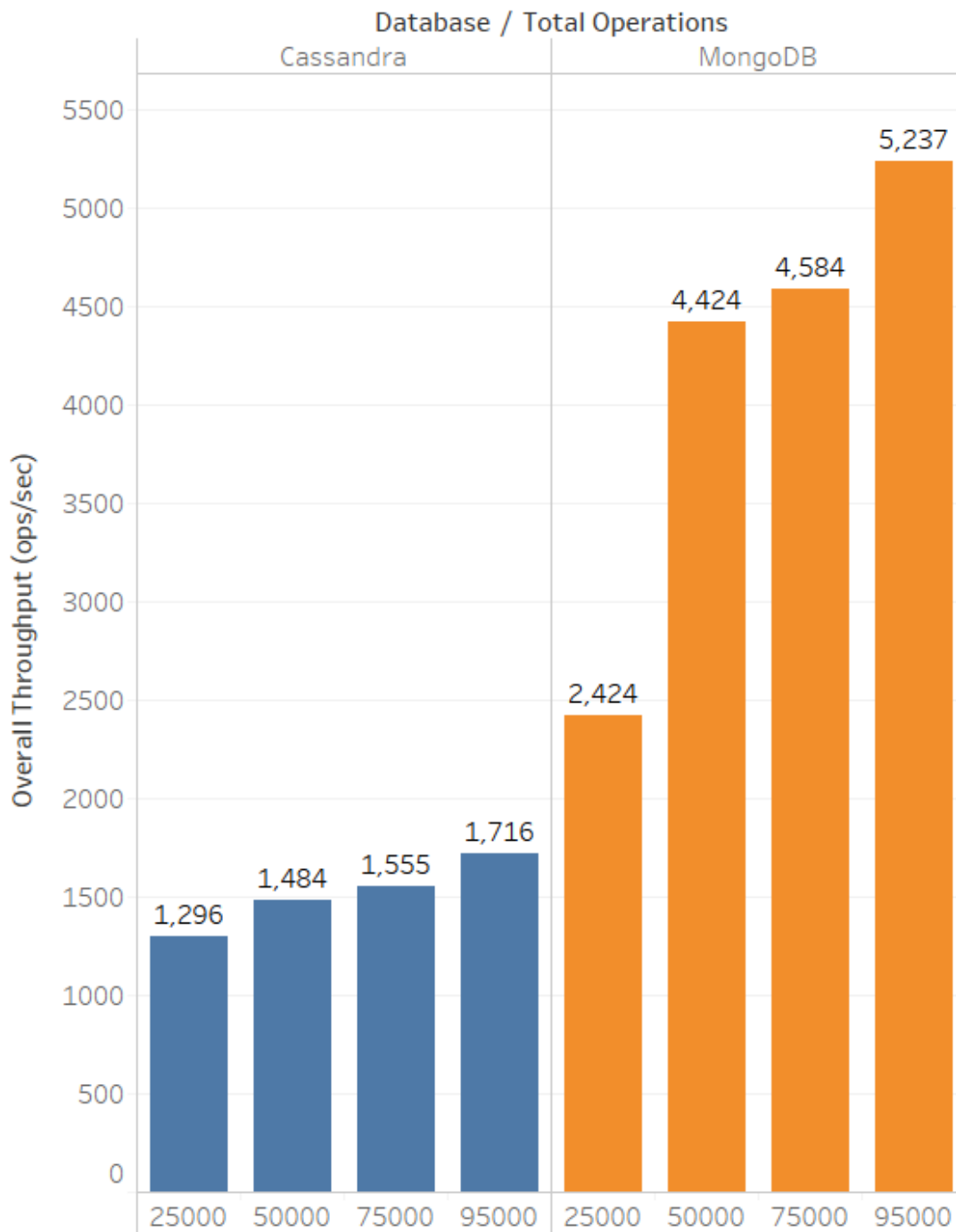


Figure 13: Workload B: Overall Throughput Vs Total Record Operations

CONCLUSION

MongoDB and Cassandra were compared by YCSB with two different types of workloads wherein workload A is 50/50 read-update ratio and workload B is 95/5 read-update ratios. MongoDB has outperformed over Cassandra at both of them workloads. The Cassandra has shown better performance at workload B with low update latency but overall MongoDB has ruled on it. The overall output shows contradiction to some research papers which describes Cassandra as winner (Abramova et al.,2013) but this may differ due to the environmental configurations. The Cassandra shows better performance with operation count more than 500000 and with higher RAM configuration.

REFERENCES

- Băzăr, C. and Iosif, C.S., 2014. The transition from rdbms to nosql. a comparative analysis of three popular non-relational solutions: Cassandra, mongodb and couchbase. *Database Systems Journal*, 5(2), pp.49-59.
- Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J., 2011, November. Security issues in nosql databases. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on* (pp. 541-547). IEEE.
- Klein, J., Gorton, I., Ernst, N., Donohoe, P., Pham, K. and Matser, C., 2015, February. Performance evaluation of nosql databases: A case study. In *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems* (pp. 5-10). ACM
- Abramova, V. and Bernardino, J., 2013, July. NoSQL databases: MongoDB vs cassandra. In *Proceedings of the international C* conference on computer science and software engineering* (pp. 14-22). ACM.
- Gao, X., Investigation and Comparison of Distributed NoSQL Database Systems.
- MongoDB v s Cassandra , <https://scalegrid.io/blog/cassandra-vs-mongodb/> [Date accessed: 28-04-2018]
- Hossen, A.K.M., 2017. "Prototype Implementation of Virtual Datacenter using Hadoop Framework for Distributed and Parallel Big Data Processing (Doctoral dissertation).
- Manoj, V., 2014. Comparative study of nosql document, column store databases and evaluation of cassandra. *International Journal of Database Management Systems*, 6(4), p.11.
- Srinivas, S. and Nair, A., 2015, August. Security maturity in NoSQL databases-are they secure enough to haul the modern it applications?. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on* (pp. 739-744). IEEE.
- Zahid, A., Masood, R. and Shibli, M.A., 2014, June. Security of sharded NoSQL databases: A comparative analysis. In *Information Assurance and Cyber Security (CIACS), 2014 Conference on*(pp. 1-8). IEEE.
- MongoDB , available at : <https://www.mongodb.com/blog/post/atrest-encryption-in-mongodb-3-2-features-and-performance> [Date accessed: 28-04-2018]