Benchmark test for NoSQL databases- HBase and MongoDB at different workloads using Yahoo Cloud Serving Benchmarking (YCSB)

Tools used: HBase, MongoDB, YCSB, Apache Hadoop

Result are visualized using Tableau.

Varun Sundaram

19th December 2018

Email @ varunsundaram@live.in

Contents

Abstract

The current digital world which is growing fast in terms of data. It is difficult to maintain the volume variety and the velocity. In general, this is referred to Big-Data. All these data are generally maintained by the traditional Relational database management system and other sources. Now currently, they are accompanied by the alternative Database Management system such as NoSql. This particular project provides the classification, benchmark test and the characteristics of the NoSQL databases. The outcome helps the users to know about the independent advantage and disadvantage of the NoSql database in terms of supporting the application which processes the more amount of data.

***Keywords**: Bigdata, Nosql, Volume, performance, use case.*

Introduction

In this project, two databases were taken **Apache HBase & MongoDB**. For these databases, benchmark test was compared using the Yahoo! Cloud Serving Benchmark (Y.C.S.B). Hbase was installed on top of hdfs, since Hadoop can handle the variety of data such as structured, unstructured, and semi-structured data. The choice of a rational database and NoSQL are generally dependent on the business need. If the organization has structured data with fixed schema then the best fit would be the relational database. However, when the data is unstructured without any schema then NoSQL would be the better choice. Nosql databases generally won't have any particular schema they are highly suited for handling more volume and variety etc. Apache hadoop has been installed in pseudo distributed mode on top of it hbase was configured.

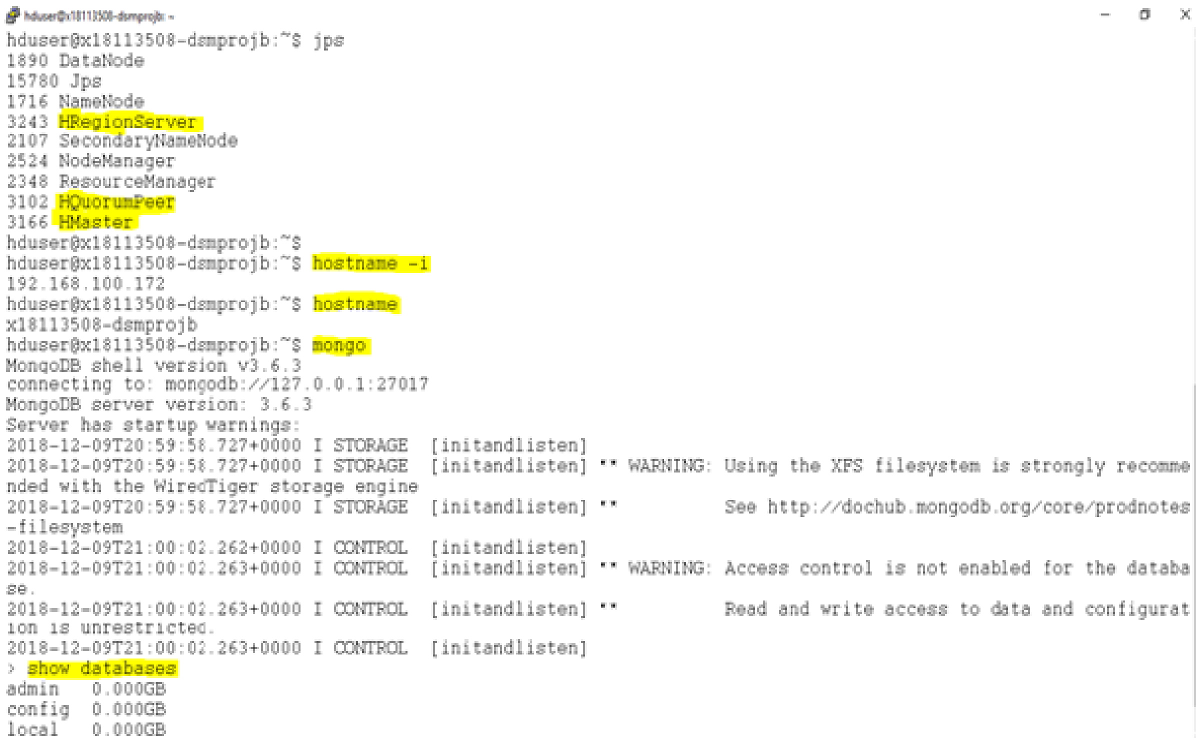
*The YCSB benchmark test ran to compare the performance of HBase / Mongo. Totally **5** opscounts were given: **100k, 200k, 300k, 400k, and 500k** with **2** different workloads (Workload**A** & Workload**F**) ran **thrice** the average result has been compared.*

In the YCSB benchmark test, read update and the insert operation were compared with corresponding Average latency. The overall throughput of each workload was also compared. The final performance test result has been discussed based on comparing the average result.

Instance setup:

This entire project has been worked on OpenStack cloud platform. Chosen hbase on top of hdfs (*hadoop pseudo distributed mode)* and along with the mongodb which was installed with help of Ubuntu repository. (# sudo apt-get install mongodb)

Below the working instance screen has been captured for the reference. All the hadoop, hbase services with mongoDB are up and running as a part of project requirement.



```
hduser@x18113508-dsmprojb: ~
hduser@x18113508-dsmprojb:~$ jps
1890 DataNode
15780 Jps
1716 NameNode
3243 HRegionServer
2107 SecondaryNameNode
2524 NodeManager
2348 ResourceManager
3102 HQuorumPeer
3166 HMaster
hduser@x18113508-dsmprojb:~$
hduser@x18113508-dsmprojb:~$ hostname -i
192.168.100.172
hduser@x18113508-dsmprojb:~$ hostname
x18113508-dsmprojb
hduser@x18113508-dsmprojb:~$ mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
Server has startup warnings:
2018-12-09T20:59:58.727+0000 I STORAGE  [initandlisten]
2018-12-09T20:59:58.727+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recomme
nded with the WiredTiger storage engine
2018-12-09T20:59:58.727+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes
-filesystem
2018-12-09T21:00:02.262+0000 I CONTROL  [initandlisten]
2018-12-09T21:00:02.263+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the databa
se.
2018-12-09T21:00:02.263+0000 I CONTROL  [initandlisten] **          Read and write access to data and configurat
ion is unrestricted.
2018-12-09T21:00:02.263+0000 I CONTROL  [initandlisten]
> show databases
admin   0.000GB
config  0.000GB
local   0.000GB
```

Key Characteristics of MongoDB /Hbase:

Apache HBase is an open source column-based, key-value NoSQL database which runs at top of hdfs. This database has been modelled after google big-table. It is generally used more as Data Ware House solution. Hbase, follow same master/slave architecture like hdfs. Hbase integrated with hdfs and the data access engine are done through YARN. It generally provides the result for map-reduce job which run in the hdfs. Hbase has high throughput with the low input and output latency. SQL is generally supported in the HBase with the help of Apache phoenix it acts as a layer for it. Some of the top companies like Facebook, Yahoo, and Adobe etc. uses the HBase for their business. [1]

*Hbase generally satisfies **C**onsistency and **P**artition tolerance the Availability part has been compromised in Hbase as per **CAP** theorem.*

Hbase Key Characteristics:

- Fault tolerant : It can be replicated across location / data centers and operated during failures.
- Consistency : Provide high speed with continues read and write.
- API Support : Clients can access easily since Java API supported here.
- Mapreduce : Data parallel processing in Map-reduce are supported here.
- Data type : Both semi structured and unstructured data supported.
- Thrift gateway : Non-Java front end supported through thrift gateway.
- Replication : It supports replication across cluster.
- Distributed storage: Distributed storage such as hdfs are supported here.
- Fail over : Automatic failover supported in the Hbase [2]

Mongo-db is an open source document oriented NoSql database. It generally stores data in JavaScript Object Notation – JSON. Mongodb's collection are generally equal to table in Relational database management system. Mongodb was developed in C++. The retrieval of quires in Mongodb are fast. The document fields are indexed with the primary & secondary. Mongodb supports the data replication. Mogodb has GridFS it's a file system it generally store and retrieve the data such as video, audio etc. Mongodb is single master machine the read operation go to primary one by default. Mongodb uses more system memory compared to other Nosql databases. In Mongodb use the dynamic schema which remove the need of pre-define structure, such as value or field types. Some of the top companies like Ebay, Bosh, Adobe, and Expedia use Mongodb for their business. [3]

*MongoDB falls under **C**onsistency & **P**artition tolerance and it is compromised with Availability as per CAP theorem.*

Mongodb Key Characteristics:

- MongoDB supports various storage engines.
- MongoDB has Auto-sharding for horizontal scalability.
- Any field in a document can be indexed.
- Built-In replication for high-availability.
- It can handle a large amount of data load.
- Multiple users can access data at a time with locking feature to ensure the concurrency.
- MongoDB support Aggregation.
- It supports master/slave replication like hdfs.
- MongoDB has load balancing facility. [4]

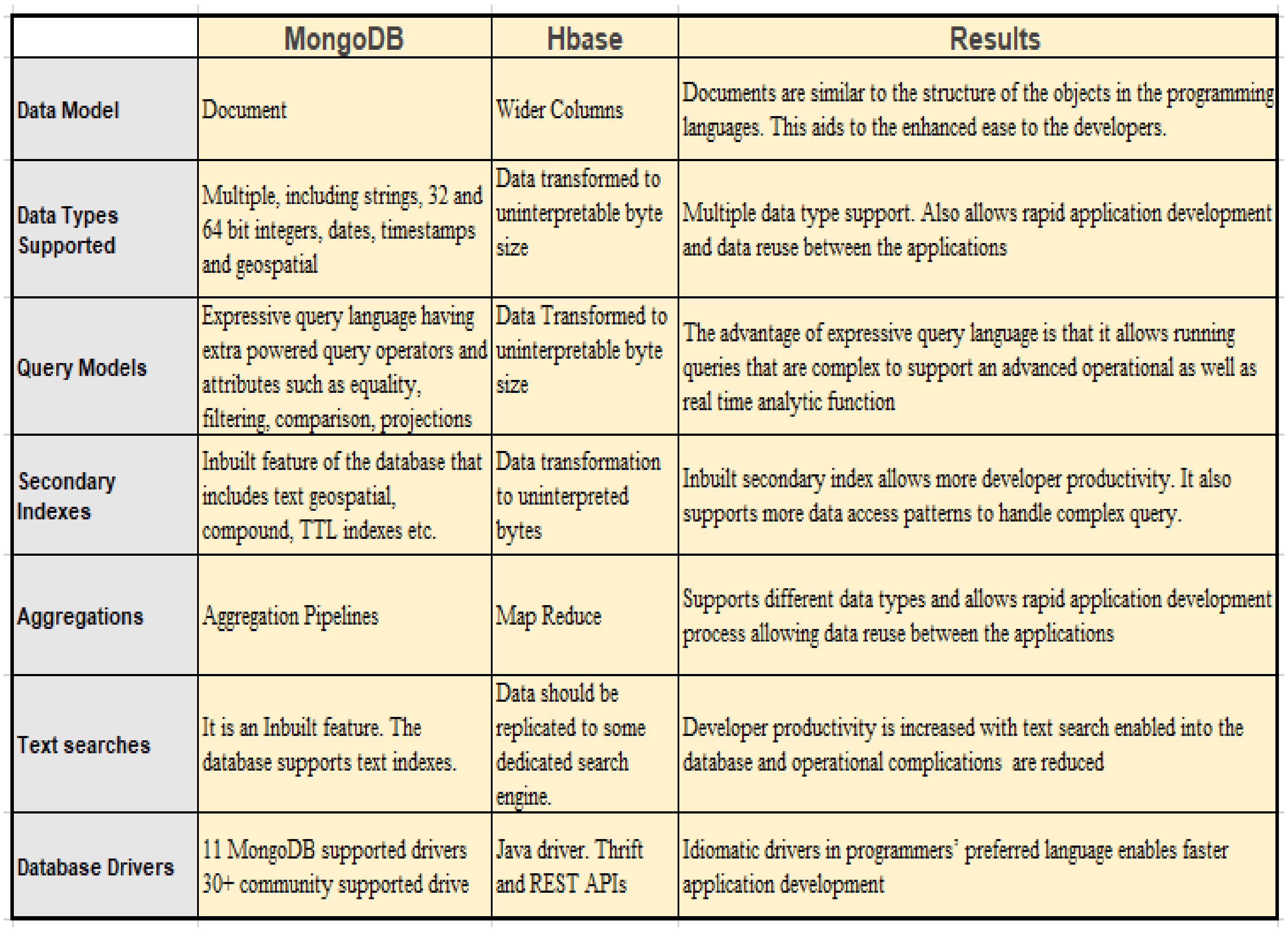
CAP Theorem:

It is a concept of distributed database system – Consistency(C) Availability (A) Partition tolerance (P). In this theory, it can satisfy only any two out of three. Particularly in current Bigdata world CAP has been used widely. For RDBMS it has ACID properties likewise NoSQL has CAP theorem. However, CAP is quite different from the ACID properties. [5]

Consistency	–	All the replicas are in sync and have the same data at any point in time.
Availability	–	All the client request which is received for read/write will be fulfilled.
Partition tolerance	–	Nodes operate and interact with each other even during any failures.

*As discussed above both **HBase** and **MongoDB** satisfy **C**onsistency & **P**artition tolerance and they're compromised with Availability.*

Major comparison between Mongodb and Hbase.



	MongoDB	Hbase	Results
Data Model	Document	Wider Columns	Documents are similar to the structure of the objects in the programming languages. This aids to the enhanced ease to the developers.
Data Types Supported	Multiple, including strings, 32 and 64 bit integers, dates, timestamps and geospatial	Data transformed to uninterpretable byte size	Multiple data type support. Also allows rapid application development and data reuse between the applications
Query Models	Expressive query language having extra powered query operators and attributes such as equality, filtering, comparison, projections	Data Transformed to uninterpretable byte size	The advantage of expressive query language is that it allows running queries that are complex to support an advanced operational as well as real time analytic function
Secondary Indexes	Inbuilt feature of the database that includes text geospatial, compound, TTL indexes etc.	Data transformation to uninterpreted bytes	Inbuilt secondary index allows more developer productivity. It also supports more data access patterns to handle complex query.
Aggregations	Aggregation Pipelines	Map Reduce	Supports different data types and allows rapid application development process allowing data reuse between the applications
Text searches	It is an Inbuilt feature. The database supports text indexes.	Data should be replicated to some dedicated search engine.	Developer productivity is increased with text search enabled into the database and operational complications are reduced
Database Drivers	11 MongoDB supported drivers 30+ community supported drive	Java driver. Thrift and REST APIs	Idiomatic drivers in programmers' preferred language enables faster application development

Architecture

Hbase:

Hbase has similar master and slave architecture like hdfs. Hbase master node is called as 'HMaster' and the Slave nodes are 'Region server'. All the request which are received from the Hbase is generally redirected to Region servers by the HMaster. HBase works in a distributed environment and handles many region servers hence it required Zookeeper.The Zookeeper acts as a failover controller it generally coordinate and communicates the current status between the HMaster and the Region server. [6]

Few important components of Hbase are:

- HMaster
- Region server
- Zookeeper

Hbase default port numbers:

- HMaster – 60000, HMaster web UI – 60010, Region Server – 60020.

HMaster:

- It monitors the region server and redirect the request.
- Any schema or metadata changes those operation taken care by HMaster.
- It maintains all the metadata of the hbase.
- It also take care of load balancing and distribute the work to region servers.

Meta table:

Holds the location information of region servers consist of host id and path details.

Region server:

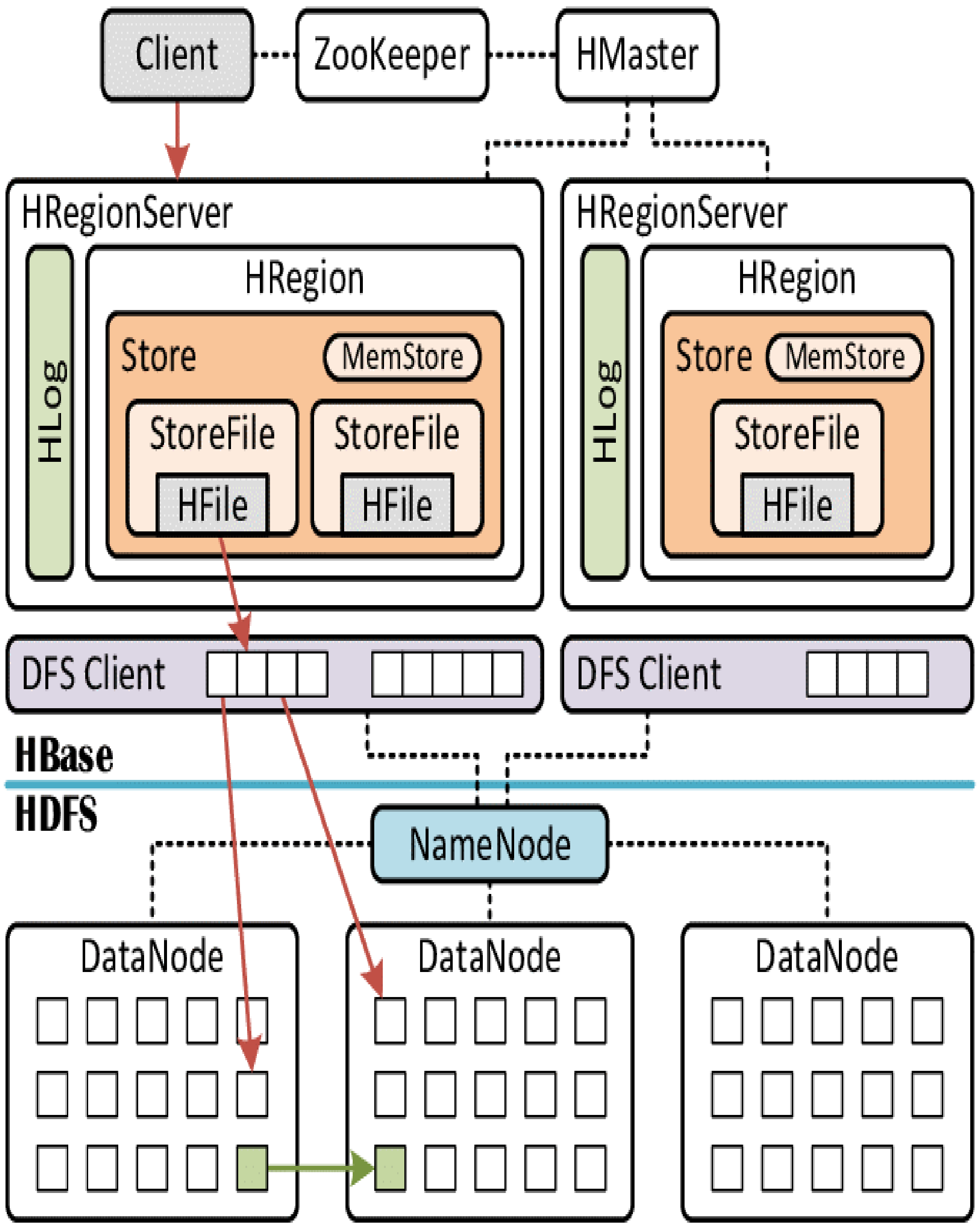
They are generally worker node for HBase read/write/update/delete operation is performed here as redirected by the HMaster.

Region server internally has 4 different components:

- *Write-Ahead-Log (WAL):* Used during failures, it stores new data that haven't persisted to storage.
- *Block Cache*: It generally stores frequently read-data in memory.
- *Memstore*: It writes the cache and stores new data which haven't written in a disk.
- *H-files*: Stores the row assorted key value on the disk.

Zookeeper: (HQuorumPeer)

- Client's interaction are usually done to region server through Zookeper.
- It tracks all the region server availability and ensure there status through heartbeat signal.
- Zookeeper works in distributed way and its each file system called **Z-node.**
- Zookeeper generally configured in Odd numbers (1,3,5,7 etc)
- Zookeper will keep HMaster synchronized and it is important component for hbase.


Client
ZooKeeper
HMaster
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
StoreFile
HFile
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
DFS Client
DFS Client
HBase
HDFS
NameNode
DataNode
DataNode
DataNode

Source: [7]

MongoDb:

Mongodb design focused on combination of Relation database with NoSQL technologies. It has flexible storage architecture. It allows multiple storage engines to store the data. They generally store the data as document in Binary representation known as BSON – Binary JSON. The document contain fields and each field contain specific values such as data types, arrays etc. [8]

Mongod: Database instance.

Mongos: Sharding process.

Mongo: Interactive shell.

Core of MongoDB works in BSON. *Online* data are managed with RDBMS and *offline* data are managed with hadoop / OLAP or EDW etc.

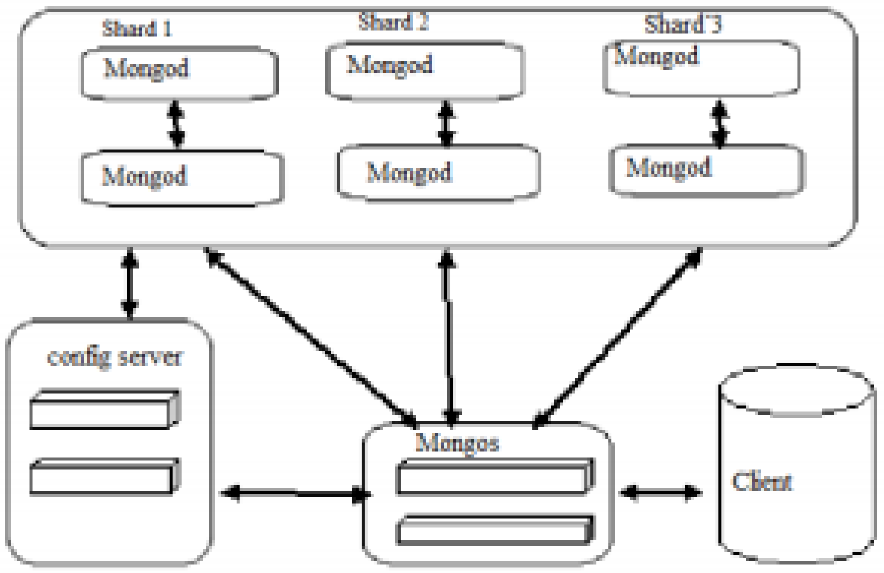
Query Routers: One or more mongos instance for the cluster. It will be deployed each application server.

Shards: Two or more replica set.

Arbiter: Is a mongod instance and part of replica set, it won't contain data.

Mongodb cluster built based on 3 main component Shard node, configuration Server, Mongos.

Shard node contain one or more shards ad each stores data into the database and each stores data into the database. It will be in replicated mode as primary and secondary. If the primary goes down secondary will run. All the operation which runs in primary will be routed to secondary without any downtime.



Source: [9]

Security Comparison between Hbase and Mongo:

Hbase:

Hbase generally allows the users to perform the read/write the available tables. However, in the enterprise setup, it is not done. Admins usually set up the firewall and allow the client to communicate with the hbase. Hbase super user allowed to perform all the operation and those details will be configured in hbase-site.xml.

Kerberos:
Authentication → Authorization → Service request

Kerberos can be implemented in Hadoop / HBase they provide the user authentication. Kerberos is a network authentication provided for client & server using a key concept. Once the client proves their identity to a server (vice-versa) the communication will be established. [10] [11]

Hbase has been configured with the secure client which run at top of hdfs. Hbase generally needs Kerberos credentials to authenticate and interact with a Kerberos-enabled cluster. Hbase performs its authentication using the key tab file. The method for generating the key tab file will be the same as creating for hdfs.

Kerberos principal will be provide to Hbase client. Password will be assigned in keytab file. Each client principal has ticket validity, it will be renewed to avoid the interruption in the session. Ticket validity can be set using the K-admin example: `addprinc -maxrenewlife 2days`. Usually the long running jobs or scripts will be assigned with maximum 3 days ticket validity based on their requirement .

In Hbase generally 3 table level security related commands used: Revoke, Grant & User permission.

Grant: Permit user Read/Write/Execute on table.

R- Read, W- write, X-execute, C-create, A- admin privilege

```
Hbase (main):010:0> grant 'usertable', 'RWXCA'
```

Revoke: Access for the table has been removed.

```
hbase> revoke <james>
```

User permission: Provide permission for table.

```
hbase> user_permission ‘usertable
```

MongoDB:

Mongodb has default inbuilt authentication also it has external authentication. It generally check the clients try to connect the mongodb.

Mongo supports multiple authentication method for verifying the user identity. Here are the list of following authentication methods.

- Salted Challenge Response Authentication Mechanism – SCRAM (***default)***
- Kerberos
- LDAP
- X.509 Certificate

Shard cluster authentication:

In this cluster the clients who try to authenticate will be redirected to ***mongos*** instances directly.

SCRAM:

It is the default authentication method used by the MongoDB. It generally verify the user credentials in the `admin.system.users.db` the authentication database was created with combination of username/server for the identification.

Kerberos:

It is used widely in the current industry for authentication purpose between the client and the servers.Kerberos user and service principals will be configured in Mongodb. For client authentication the ticket will be generated and distributed from the Key distribution centre (KDC). Once the client and server key matches the user can able to login. All this entire process happen in backend.

LDAP:

The enterprise edition of MongoDB LDAP authentication will be used. User management required for LDAP and MongoDB. Generally in Linux based system it will be used. LDAP authentication requires that MongoDB forward the user’s password in plain text. Specify digestPassword set to false during authentication.

X.509 Certificate:

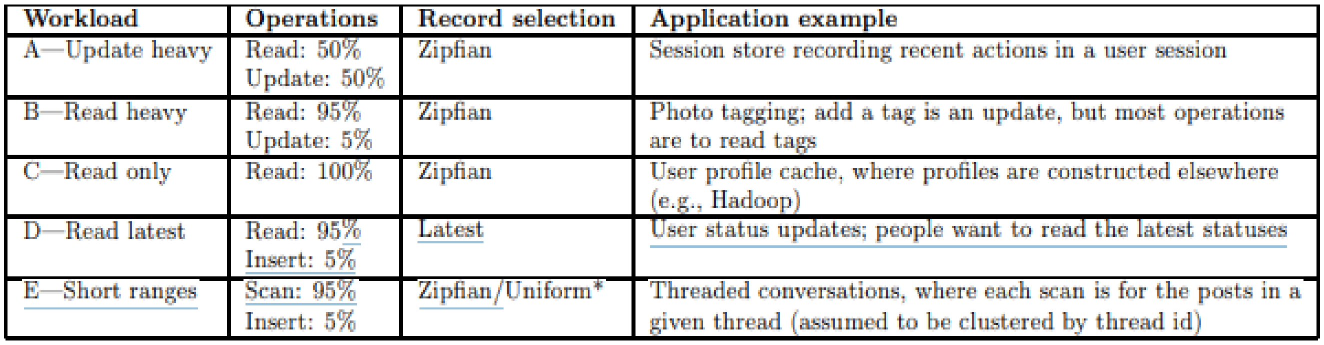
In Mongodb the X.509 certificate used for authenticating with the secure SSL & TLS connection. In this client authenticate will be allowed using the certificate instead user name / password. In the enterprise edition certificate will be generated and signed by single-certificate-authority and it will be provided to both client / server. [12]

In general the security parameter will be configured in Hbase/ MongoDB based on the enterprise and it can be customized according to their need. Usually they won’t relay with single option.

Literature Survey:

Benchmark test used to the find the performance of the existing work efficiency of their database. There are several benchmarking tool available to perform this. However Yahoo provided open source YCSB. This particular tool support the benchmark mark test for DB's like MongoDB, Hbase, Cassandra, Mysql, CouchDb etc. The YCSB has pre-in-built script it generally invokes the test using the YCSB client and it will load each dataset and workload according to the user requirement and their needs. Based on the successful benchmark outcome we compare the result and decide which database can be used. [13]

In this particular survey the benchmark test for Hbase, MongoDB and other Nosql db was performed using the YCSB benchmark tool. In this performance test they compared Throughput with workload, and average latency with workload. Totally five different workload was selected to compare the mentioned databases. Each workload there are different operation counts are given and based on that the benchmark test has been ran. [14]



Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Insert: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Scan: 95% Insert: 5%	Zipfian/Uniform*	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

Source: [14]

The YCSB provided fair comparison. For each and every workload overall throughput is compared with the Operation/sec and along with the operation in Average latency. Based on the comparison the graphs were plotted and along with the workloads were explained.

Here they have compared the performance test for the databases ran on node count with ***three different load on the three different days****. The final average has been calculated based on the database, node, and workload.*

The final result of Hbase was mentioned as they encounter with problem if it have the node count as 8 and above and run out of disk on the shard during the load process. Turing on compression resolved this.

MongoDB unable to handle the range scans in sharded config, when the load run in the single node it looks comparatively acceptable however the multi-node results to run in long time.

Based on the above Literature learning for this current project used MongoDB / Hbase made them to run in single node machine the performance has been captured using YCSB. Totally 2 workloads A and F has been used with 5 different Ops count and ran thrice. The final result has been discussed based on the average output interpretation which is discussed below.

Performance Test Plan:

To run the YCSB benchmarking tools there are few prerequisite done as mentioned below.

Given Machine Specification:

OpenStack Computer
Product: Intel Core Processor (Haswell, no TSX, IBRS)
Memory: 4 Gig
Storage: 46 Gig
OS: Ubuntu 18.04.1 LTS

Hadoop / Hbase installation:

Installation of Hadoop, on top of the hdfs need to install the hbase. Make the required changes in the configuration xml files after the successful installation make sure all the hadoop and hbase service up and running (following service should be up: DataNode, NameNode, HMaster, NodeManager, ResourceManager, HRegionServer, SecondaryNameNode, HQuorumPeer).

MongoDB installation:

Installation of the mongodb done through using the Ubuntu repository apt-get. Post installation make sure the mongo services are started and up and running.

YCSB installation:

After the successful installation of the databases download the *YCSB 0.15.0* latest version from the github and untar the file in machine. Inside the *testharness* directory give the required operation count and the workload.

Test plan:

For this test plan: 100k, 200k, 300k, 400k, 500k totally 5 different operation counts given with 2 different workloads A and F. The benchmarking test ran 3 times and the average output result has been compared.

The other process like loading and running the work load has been internally taken care by YCSB by invoking it client process. It generally loads the given operation count corresponding to the given workload. Before running the benchmark test 'usetable' should be created in the hbase. However YCSB will automatically create the tables and database in the Mongodb. After completion on each workload the YCSB will atomically delete the data.

By using the final result we can evaluate and consider which database is more efficient in terms of their need and requirement.

The above mentioned installation all the putty session logs and the installation steps are attached in the below appendices.

Evaluation and Results:

Databases used	: HBase / MongoDb
Versions	: HBase 1.4.8 / MongoDb 3.6.3 (Nosql)
Benchmarking tool	: YCSB
Operation count	: 100k, 200k, 300k, 400k, 500k
Workload count	: 2 (Workload-A & Workload-F)
Run count	: Ran 3 times and average result has been taken.

Hbase:

Installed Apache Hbase and ran at top of hdfs in pseudo mode with one zookeeper / Region server.

With default memory.

Default Heap size of Hbase has been enabled *hbase-env.sh*

# The maximum amount of heap to use. Default is left to JVM default.

```
 export HBASE_HEAPSIZE=1G
```

MongoDb:

Installed and ran mongo in single node machine.

YCSB evaluation:

Bench mark test ran in openstack and workload ran in background using **nohup** and the log in the below directory.

```
hduser@x18113508-dsmprojb:/var/log/mongodb$ cd /usr/local/testharness/
hduser@x18113508-dsmprojb:/usr/local/testharness$ ls - ltr
drwxr-xr-x  2 hduser hadoop   4096 Dec 19 15:43 ./
drwxr-xr-x 14 root   root     4096 Dec 10 01:38 ../
-rw-r--r--  1 hduser hadoop 239283 Dec 13 22:29 run1.log
-rw-r--r--  1 hduser hadoop 247575 Dec 14 01:10 run2.log
-rw-r--r--  1 hduser hadoop 237004 Dec 14 03:21 run3.log
```

The YCSB output files generated in the below directory

```
drwxr-xr-x  3 hduser hadoop 4096 Dec 17 23:28 run1/
drwxr-xr-x  3 hduser hadoop 4096 Dec 17 23:47 run2/
drwxr-xr-x  3 hduser hadoop 4096 Dec 18 00:18 run3/
hduser@x18113508-dsmprojb:~/ycsb-0.15.0/output$ pwd
/home/hduser/ycsb-0.15.0/output
```

No failures has been observed in the YCSB benchmark test.

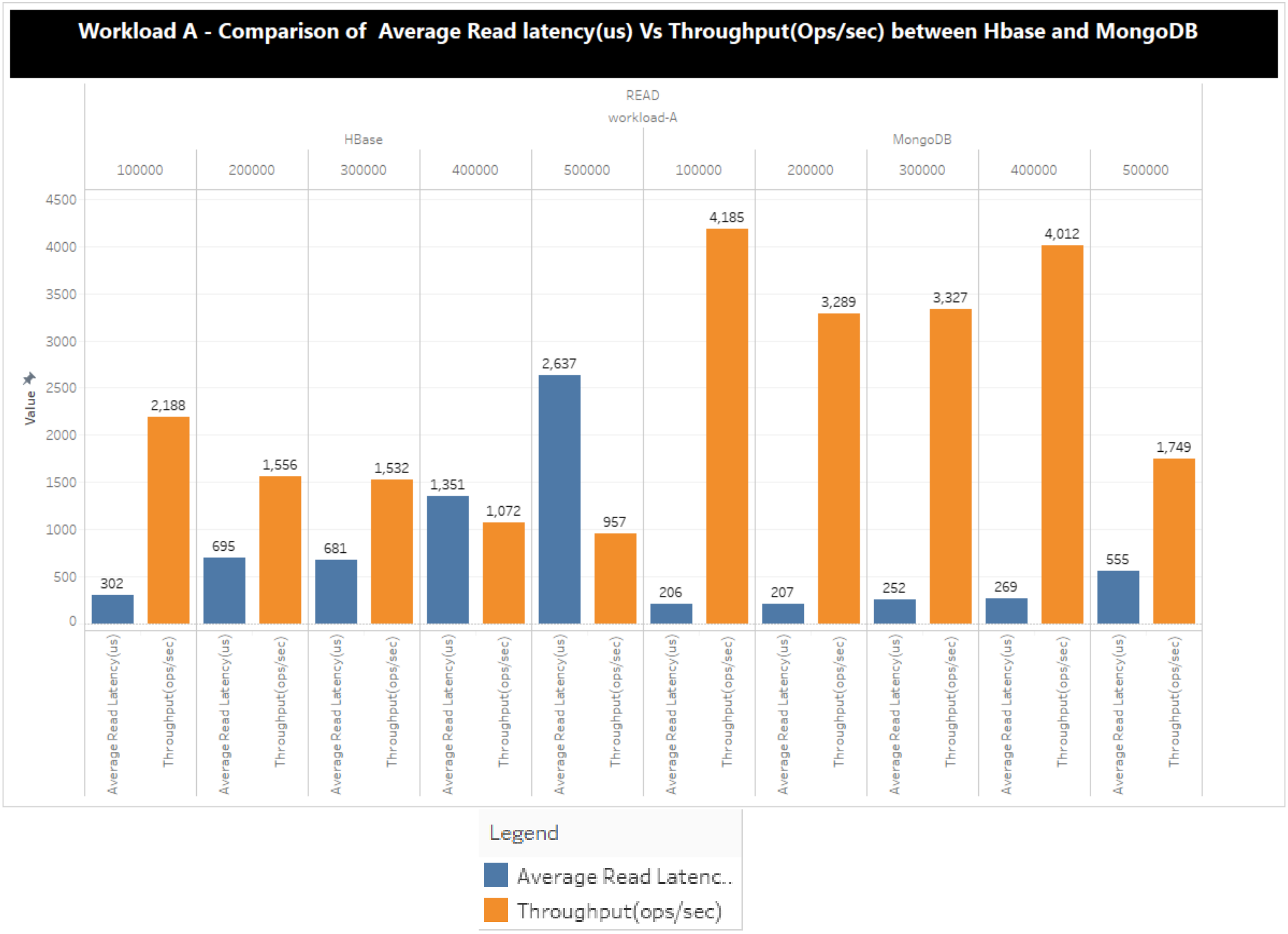
YCSB Benchmark result:

Workload A:

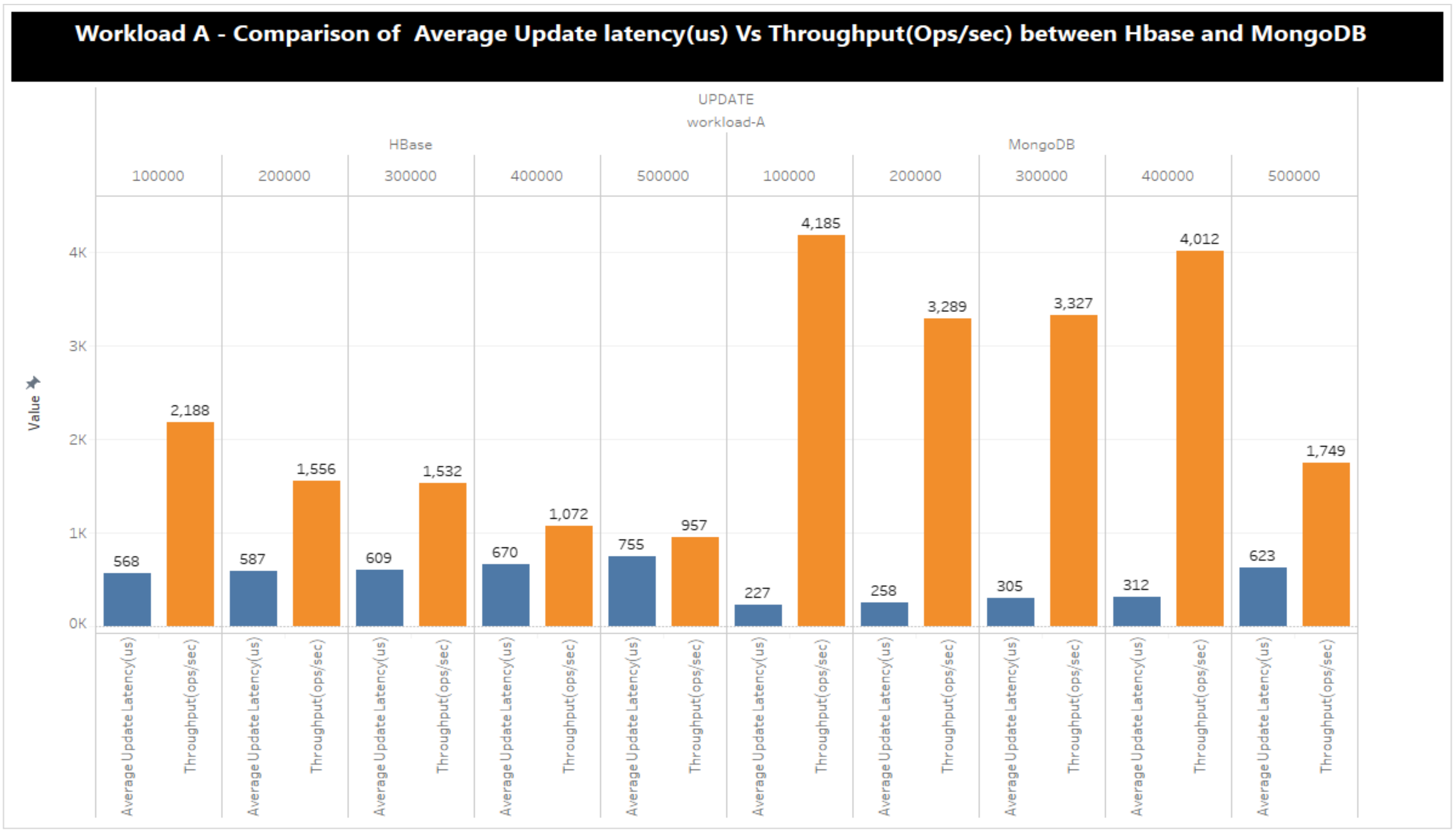
Distribution of workload: **Read 50 % and Update 50%**

Units: Average latency (us) – Micro seconds, Throughput (Ops/sec) – Operation per second

Read : 50%



Update: 50%



Legend
🟦 Average Update Late..
🟧 Throughput(ops/sec)

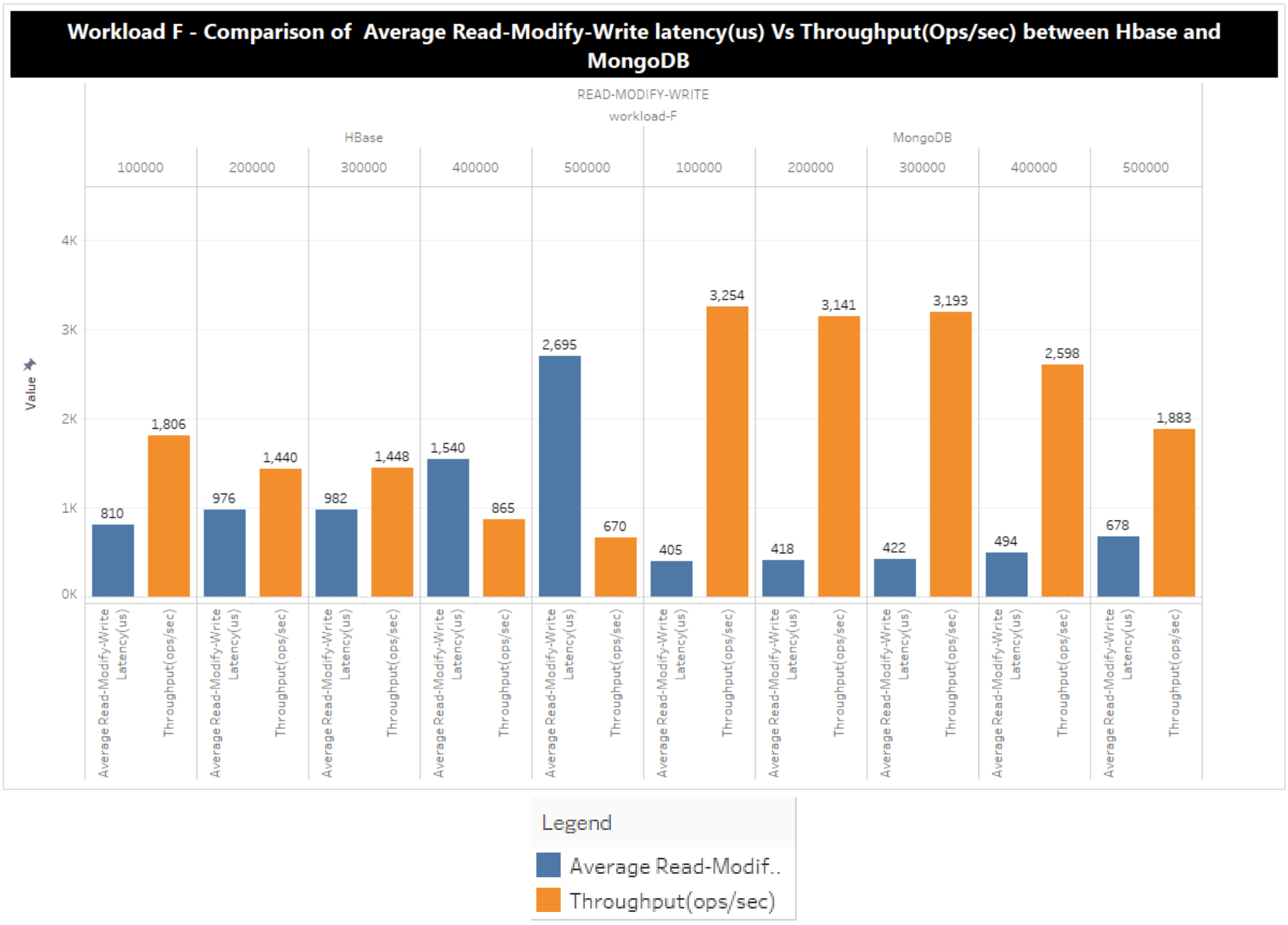
Workload F:

Distribution of workload: **Read-modify-Read 50 % and Read 50%**

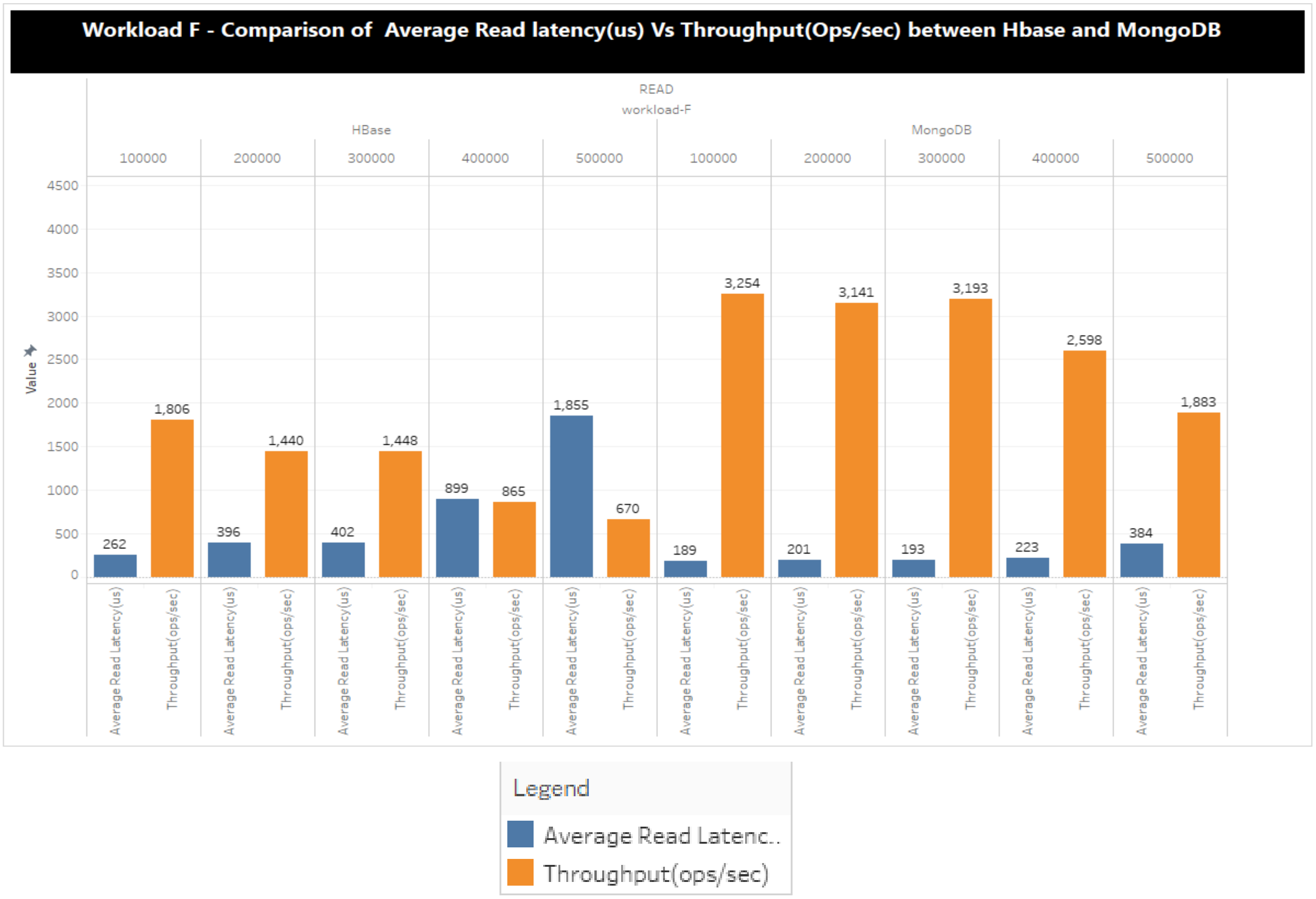
Read-modify-write. In this workload, the client will read a record, modify it, and write back the changes.

Units: Average latency (us) – Micro seconds, Throughput (Ops/sec) – Operation per second

Read-modify-Read 50 %



Read 50%



The above benchmark test ran nearly 7-8 hours, and the compared the output using the tableau and plotted graph.

When compared the overall Average result of the Workload A and Workload B, MongoDB has performed well and hbase result was comparatively low in Read/Write/Update Vs over all Throughput.

Hence the YCSB benchmark result outcome clearly states performance MongoDb better than Hbase.

Conclusion

The database is the traditional storage method which helps to maintain the variety of data such as structured / unstructured / semi-structured. However, in recent times there are many unstructured data has been generated the existing traditional RDBMS facing many challenges in storing and processing those data. Hence there are many NoSQL databases emerged and gained more popularity in the production environment. Each database has their own features and uses cases. Eg: MongoDB / CouchDB – Perform well in document type files, Cassandra – run across the many servers and it has own file system (CFS) Hbase – More Suitable for storing column related data and run top of hdfs, DynamoDb – Key value and document related.

In this current project, MongoDB and HBase has been compared with 5 different types of heavy ops counts were given. The workload performed as stated 50/50 read and update another workload with read-modify-write and write. The test result shows the increase of the data size and HBase performance started to decrease even became poor sometimes when comparing with Hbase the MongoDB performed well on running on different workloads like read/write/update operation and the results are pretty much faster in it.

Upon reviewing the overall YCSB benchmark test analysis Hbase was not adequately performed well and the MongoDB performed well in the overall scenarios. Upon comparing the MongoDB / Hbase both satisfy the ***Consistency & Partition*** however the availability has been compromised as per CAP theorem.

When it comes to production environment the selection of database are purely based on requirement. However, from this current project YCSB benchmarking we couldn't conclude Hbase will perform less in general. There are several parameter which can be tweaked in Hbase in enterprise level (i.e. Cloudera / Hortonworks.) Also several contributing factors which helps the Hbase to work fast and reliable.

Since Hbase running top of Hdfs generally enterprise level there will be more Data node / Region server hence the performance won't be hinder much.

Below listed following parameter which are usually added in Hbase enterprise level for performance tuning.

Few samples parameter tweak: [15]

- Heap_Size for HMaster: Since Master basically responsible for processing Meta data.
- Heap_Size for Regionserver: Since all the load and process happen here more heap memory will help for quick processing
- Total count of Region server: It helps to increase the parallel process.
- File_Block_Cache Size: Here total heap use the cache and read faster.
- Hbase memory region flush size: Memory store will be flushed here in form of hfile bigger flush size will do better performance

Since for educational purpose we are performing benchmark test on Open source edition of both Hbase / MongoDB however in the enterprise edition the scenario will be totally different we can't rely with above result. The overall result is applicable only for the Open source edition of Apache Hbase & MongoDb.

References:

[1] “Apache HBase – Apache HBase^{TM} Home.” [Online]. Available: https://hbase.apache.org/. [Accessed: 15-Dec-2018].

[2] Tutorialspoint.com, “HBase Overview,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/hbase/hbase_overview.htm. [Accessed: 15-Dec-2018].

[3] N. K, “MongoDB for absolute beginners: ACID and CAP,” *MongoDB for absolute beginners*, 07-Jun-2016. .

[4] “MongoDB Features - javatpoint.” [Online]. Available: https://www.javatpoint.com/mongodb-features. [Accessed: 16-Dec-2018].

[5] R. Prasad, “CAP Theorem simplified,” *Ravindra Prasad*, 25-Aug-2017. .

[6] “HBase Architecture | HBase Data Model | HBase Read/Write | Edureka,” *Edureka Blog*, 17-Nov-2016. .

[7]“hbase-architecture.png (584×320).” [Online]. Available: https://bighadoop.files.wordpress.com/2014/05/hbase-architecture.png. [Accessed: 14-Dec-2018].

[8] “MongoDB Architecture | Wideskills.” [Online]. Available: http://www.wideskills.com/mongodb-tutorial/03-mongodb-architecture. [Accessed: 14-Dec-2018].

[9]MongoDB, “MongoDB Architecture Explained.” [Online]. Available: https://www.ibmbpnetwork.com/linux-blog/mongodb-architecture. [Accessed: 16-Dec-2018].

[10] © 2018 Cloudera, I. A. rights reserved A. Hadoop, associated open source project names are trademarks of the A. S. F. F. a complete list of trademarks, and C. Here, “Managing HBase Security | 5.14.x | Cloudera Documentation.” [Online]. Available: https://www.cloudera.com/documentation/enterprise/5-14-x/topics/admin_hbase_security.html#id_v2w_bv3_tw. [Accessed: 14-Dec-2018].

[11] “Chapter 8. Secure Apache HBase (TM).” [Online]. Available: http://hbase.apache.org/0.94/book/security.html. [Accessed: 14-Dec-2018].

[12] “Security — MongoDB Manual,” *https://github.com/mongodb/docs/blob/v4.0/source/security.txt*. [Online]. Available: https://docs.mongodb.com/manual/security. [Accessed: 14-Dec-2018].

[13] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with YCSB,” in *Proceedings of the 1st ACM symposium on Cloud computing - SoCC ’10*, Indianapolis, Indiana, USA, 2010, p. 143.

[14] *Datastax.com*, 2018. [Online]. Available: https://www.datastax.com/wp-content/themes/datastax-2014-08/files/NoSQL_Benchmarks_EndPoint.pdf. [Accessed: 17- Dec- 2018]

[15] “HBASE Master Heap Size Recommendation - Hortonworks.” [Online]. Available: https://community.hortonworks.com/questions/81527/hbase-master-heap-size-recommendation.html. [Accessed: 18-Dec-2018].

Appendices

Below are the step-by-step installation commands has been given which was used for my project along with instance installation setup sessions putty logs for reference.

x18113508_putty_log.txt

Hadoop installation

```
sudo apt-get update
sudo apt-get install openjdk-8-jre
sudo apt-get install openjdk-8-jdk
sudo apt-get install ssh
sudo apt-get install rsync
sudo apt-get install vim
sudo apt-get install ifconfig
sudo apt-get install python
sudo apt-get install curl



sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo adduser hduser sudo
su - hduser


ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
ssh localhost



sudo vi /etc/sysctl.conf
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1

cd /usr/local
sudo wget
http://ftp.heanet.ie/mirrors/www.apache.org/dist/hadoop/common/stable/hadoop-
2.9.2.tar.gz
sudo tar xzf hadoop-2.9.2.tar.gz
sudo ln -s hadoop-2.9.2 hadoop
sudo chown -R hduser:hadoop hadoop-2.9.2



cd /usr/local/hadoop/etc/hadoop
cp mapred-site.xml.template mapred-site.xml

vi hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

vi mapred-site.xml
<configuration>
<property>
<name>mapreduce.jobtracker.address</name>
<value>local</value>
</property>
</configuration>

mkdir ~/tmp
mkdir ~/hdfs
chmod 750 ~/hdfs


vi core-site.xml
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hduser/tmp</value>
</property>
<property >
<name>fs.defaultFS</name>
<value>hdfs://localhost:54310</value>
</property>
</configuration>




vi hdfs-site.xml
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hduser/hdfs</value>
</property>
</configuration>
```

```
cd /usr/local/hadoop
bin/hdfs namenode -format
sbin/start-dfs.sh
sbin/start-yarn.sh
jps
```

Mapreduce sample word-count job test

```
cd /tmp
mkdir gutenberg
cd gutenberg
wget http://www.gutenberg.org/files/4300/4300-0.txt

cd /usr/local/hadoop

bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/hduser
bin/hdfs dfs -mkdir /user/hduser/gutenberg

bin/hdfs dfs -copyFromLocal /tmp/gutenberg/* /user/hduser/gutenberg

bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar wordcount
/user/hduser/gutenberg /user/hduser/gutenberg-output

bin/hdfs dfs -ls /user/hduser/gutenberg-output

mkdir /tmp/gutenberg-output

bin/hdfs dfs -get /user/hduser/gutenberg-output/part-r-00000 /tmp/gutenberg-output

cd /tmp/gutenberg-output
```

Hbase installation

```
cd /usr/local
sudo wget http://ftp.heanet.ie/mirrors/www.apache.org/dist/hbase/stable/hbase-1.4.8-
bin.tar.gz
sudo tar xzf hbase-1.4.8-bin.tar.gz
sudo ln -s hbase-1.4.8 hbase


sudo chown -R hduser:hadoop hbase-1.4.8
sudo mkdir /usr/data
sudo mkdir /usr/data/hbase
sudo chown -R hduser:hadoop /usr/data/hbase

cd /usr/local/hbase/conf

sudo vi hbase-site.xml
<configuration>
<property>
<name>hbase.rootdir</name>
<value>file:///usr/data/hbase</value>
</property>
</configuration>


sudo vi hbase-env.sh

#Java home
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

#Native lib
export LD_LIBRARY_PATH=/usr/local/hadoop/lib/native




cd /usr/local/hbase

bin/start-hbase.sh

bin/hbase shell
status
create 'newtable', 'colfam1'
create 'usertable', 'cf1'
put 'newtable', 'row-1', 'colfam1:q1', 'val-1'
put 'newtable', 'row-2', 'colfam1:q2', 'val-2'
scan 'newtable'
```

HBase on Hdfs :

```
cd /usr/local/hadoop
sbin/stop-dfs.sh
sbin/stop-yarn.sh

wait 30sec

cd /usr/local/hbase
```

```
bin/stop-hbase.sh

jps

**Wait 30sec**

again put #jps and make sure all process closed if else kill it manually (kill -9
give pid)


vi core-site.xml --- No changes

cd /usr/local/hadoop/etc/hadoop
vi hdfs-site.xml
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hduser/hdfs/data</value>
</property>
<property>
<name>dfs.namenode.data.dir</name>
<value>/home/hduser/hdfs/name</value>
</property>
<property>
<name>dfs.datanode.data.dir.perm</name>
<value>755</value>
</property>
<property>
<name>dfs.datanode.max.xcievers</name>
<value>8192</value>
</property>
</configuration>


cd /usr/local/hbase/conf
vi hbase-site.xml

<configuration>
        <property>
                <name>hbase.rootdir</name>
                <value>hdfs://localhost:54310/hbase</value>
        </property>
             <property>
                      <name>hbase.cluster.distributed</name>
                           <value>true</value>
             </property>
             <property>
```

```
                            <name>hbase.zookeeper.quorum</name>
                            <value>localhost</value>
              </property>
              <property>
                           <name>dfs.replication</name>
                           <value>1</value>
              </property>
              <property>
                           <name>hbase.zookeeper.property.clientPort</name>
                           <value>2181</value>
              </property>
              <property>
                       <name>hbase.zookeeper.property.dataDir</name>
                            <value>/home/hduser/hbase/zookeeper</value>
              </property>
</configuration>



hostname -i
<<Note down ip address starts with 192.168*****>>

hostname
<<Note down name of machine>>


sudo vi /etc/hosts
<<add in last line  ip address and name>>
eg: 192.168.*** x***-dsmprojb

<<After editing this file just keep this putty session>>

--------------------------
<<Now open new putty session>>
su - hduser
cd /home/hduser
sudo cp .bashrc .bashrc.bkp
sudo vi .bashrc
export HBASE_HOME=/usr/local/hbase
export PATH=$PATH:$HBASE_HOME/bin

<<After saving this file close this putty session>>
-------------

<<Continue below steps in old putty session>>
cd /usr/local/hadoop
sbin/start-dfs.sh
sbin/start-yarn.sh
jps

wait 30sec
```

```
cd /usr/local/hbase
bin/start-hbase.sh

bin/hbase shell
status
create 'test', 'colfam1'

Then, exit
```

MongoDB installation

```
sudo apt-get update
sudo apt-get install mongodb
sudo service mongodb start
sudo service mongodb status
```

YCSB setup:

```
su - hduser
cd /home/hduser
wget https://github.com/brianfrankcooper/YCSB/releases/download/0.15.0/ycsb-
0.15.0.tar.gz


sudo tar xzf ycsb-0.15.0.tar.gz
sudo chown -R hduser:hadoop ycsb-0.15.0
cd /home/hduser/ycsb-0.15.0


mkdir output

sudo -i
cd /usr/local/
curl -o testharness.tgz --location
"https://drive.google.com/uc?export=download&id=1hp53oAcDeC2M1mJ8eMPaL_hDrcdDRMo0"
tar xzf testharness.tgz

sudo chown -R hduser:hadoop testharness


su - hduser
cd /usr/local/testharness
vi opcounts.txt (#here by default two ops count will be there 1000 & 2000 edit
according to your requirement)

vi runtest.sh
```

```
<<In vi mode type :-  esc+: set number>>

( Edit line no:14 YCSB_HOME=/home/hduser/ycsb-0.8.0
  -> YCSB_HOME=/home/hduser/ycsb-0.15.0 (changed version)

 Edit line no:73 columnfamily=family
  -> columnfamily=cf1 (changed columnname) )


  vi testdbs.txt
  mongodb (add mongodb by default hbase will  be there)

  vi usertableClear.js
 change YCSB -> as ycsb (from caps to small)


=================================================================
```

Initiating benchmark test

```
 su - hduser
  cd /usr/local/testharness
  ./runtest.sh your_file_name


  output file can be seen here.
   cd /home/hduser/ycsb-0.15.0/output/

 ======================================================================
```