**Comparison between HBASE and CASSANDRA**

**Using YCSB**

**MSC in DATA ANALYTICS (BATCH A)**

**STUDENT NAME - SHREEYA NAMBOORI**

**ABSTRACT**

Industries are switching to NOSQL databases because of the limited capabilities offered by SQL databases when it comes to handling the data that is growing at a fast pace. There are different types of NOSQL databases available in the market like MongoDB, Voldemort, Cassandra and HBase. The performance of NOSQL databases differs from each other based on the type of workload and the type of operation they are used for and that should be kept in mind when selecting an appropriate NOSQL database. In this paper YCSB (Yahoo Cloud Serving Benchmark) tool has been used for performance comparison of NOSQL databases Cassandra and HBase. The results shows the comparison between the two databases for read, update and insert operation for Workload A and Workload D.

**INTRODUCTION**

Internet is considered to be the best technological advancement so far. It seems impossible to imagine our world without internet as all our important tasks like sharing work related documents in email or Moodle or mundane tasks like ordering groceries online have become intertwined with Internet. This increasing use of Internet has given rise to copious amount of data that needs to be maintained. According to a Forbes article we are generating [2.5 quintillion bytes of data](https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1&sf100871281=1) , but this data is only going to increase with the growth of the Internet(Forbes, 2018).To maintain all this data a need of having a database that was less rigid than the relational database (RDBMS) was realized and that’s how NOSQL database came into focus.

Relational database have fixed schema (changing a schema is both problematic and time consuming), insufficient performance, high latency and limited ability to scale cost effectively which makes them unsuitable for working with the big data. NOSQL has advantages over relational database that it can be integrated with cloud, can store large amount of data and they don’t have a schema unlike relational database. In terms of scalability and performance NOSQL is more superior than the relational database. Whereas RDBMS(Relational Database) can handle only structured data, NOSQL can handle structured, semi-structured and unstructured data. All these features of NOSQL makes it ideal for working with big data. There are four categories of NOSQL databases: Graph databases (Neo4j), Wide column stores (Cassandra and HBase), Document databases (MongoDB) and Key-value stores (Voldemort).In this paper performance of Cassandra and HBase are being compared.

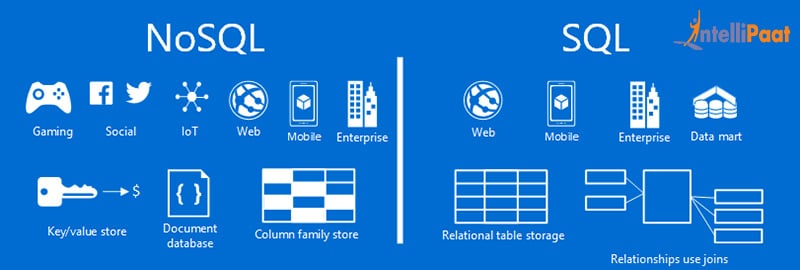


Image path-https://cdn.intellipaat.com/blog/wp-content/uploads/2016/02/NoSQL- vs.-SQL%E2%80%93-What-is-better.jpg

KEY CHARACTERSTICS

**HBASE**

HBase is a column oriented NOSQL database that runs on HDFS (Hadoop Distributed File System).  HBase provides Bigtable (Google’s database) capabilities like [MapR XD](https://mapr.com/products/mapr-xd/) on top of Hadoop-compatible file systems and is written in JAVA. It can work on large amount of data and provide real-time, random read/write access to Hadoop data. It is well suited for sparse datasets and store the data in a fault-tolerant way.

**Key features:**

* **JAVA API-** Availability of easy to use JAVA API’s makes it easier to work with HBase.
* **Replication-**HBase supports the data replication across the clusters.
* **Scalability-**HBase is linearly scalable.
* **Low Latency-** low latency access is provided to rows from a large dataset.
* **Failover Support**- automatic failure support in provided.
* **Consistency-**There is a consistency of write and read operations.
* **Automatic Sharding-** Tables can be sharded automatically.
* **Real time processing-** it easier to process a real time query with the support of block cache and bloom filters.
* **Faster Lookups**- HBase makes use of Hash tables and stores data in indexed HDFS files which makes the lookups faster.
* **Atomic read and Write-** It is offered on a row level. This means that during a read/write process all the other processes are halted from performing any read/write operation.
* **High Availability-** HBase supports deployment of multiple HBase Masters in a cluster to ensure the availability of the resources.
* **Distributed Storage-** distributed storage like HDFS is supported.
* **HBase is Schema-less- concept of fixed columns is not present in HBase.**
* **MapReduce Support- MapReduce is supported for parallel processing of copious amount of data.**
* **Data type- semi-structured and structured data are supported,**
* **Back up Support-** HBase supports the back-up of Hadoop[MapReduce jobs](https://data-flair.training/blogs/mapreduce-job-optimization-performance-tuning-techniques/) in HBase tables.

**CASSANDRA**

The Apache Cassandra is a NOSQL database that provides scalability and high availability without compromising with the performance. It is a open source distributed database system that efficiently handles large volumes of data that is spread across multiple commodity servers. It provides high availability with no [single point of failure](https://en.wikipedia.org/wiki/Single_point_of_failure). Cassandra was built by Facebook and it is based on [DynamoDB (Amazon) and BigTable](http://www.datastax.com/documentation/articles/cassandra/cassandrathenandnow.html) (Google).

**Key Features:**

* **Scalability is Elastic**-Cassandra’s scalability is elastic as more hardware can be added to accommodate more data and more customers depending upon the requirement.
* **Always on architecture-** Cassandra can be used for business-critical applications as it has no single point of failure.
* **Fast Performance-** In Cassandra throughput increases by increasing the number of nodes in the cluster which makes response time quicker.
* **Language-** Cassandra supports large number of languages like Python, C#/.NET, C++, Ruby, Java, Go, and many more.
* **Fast Writes-**Cassandra supports fast writes and stores data without affecting the efficiency of read.
* **ACID (Atomicity, Consistency, Isolation, and Durability) -** Cassandra provides supports to properties like ACID.
* **Easy data distribution** − Cassandra replicates the data across multiple data centers therefore it is flexible in distributing the data.
* **Data type- Cassandra supports structure, un-structured and semi-structured data.**
* **Simplicity-CQL (Cassandra Query Language)** act and looks just like SQL which makes it easier to transition to Cassandra from any RDBMS (relational database).
* **MapReduce support-** Cassandra supports [MapReduce](https://en.wikipedia.org/wiki/MapReduce) along with [Apache Pig](https://en.wikipedia.org/wiki/Pig_(programming_tool)) and [Apache Hive](https://en.wikipedia.org/wiki/Apache_Hive).
* **Fault tolerance and High Availability-** Data is automatically replicated to multiple nodes in Cassandra for providing both high availability and [fault-tolerance](https://en.wikipedia.org/wiki/Fault-tolerance). If any node fails, the data becomes available on different nodes for retrieval.
* **High Performance-** Cassandra has high performance compared to other NOSQL databases.
* **Schema-free-** Cassandra can create columns within the rows therefore it is schema optional.
* **Tunable Consistency-** Two consistencies are supported by Cassandra Eventual consistency and Strong Consistency. Any of the two consistencies can be chosen depending upon the requirement.

**DATABASE ARCHITECTURE**

**HBASE**

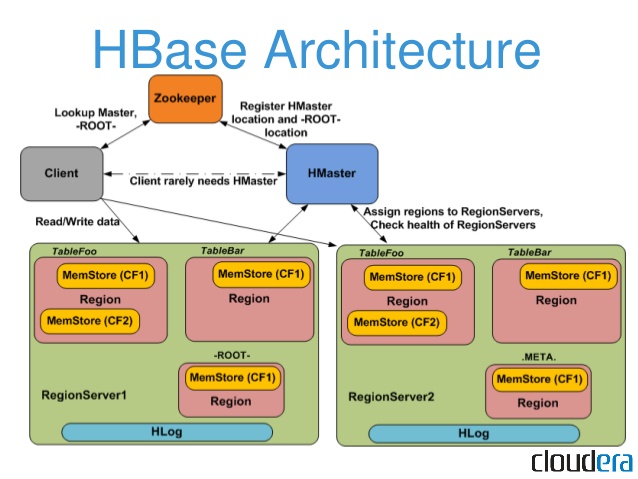


Image source- <https://s3.amazonaws.com/files.dezyre.com/images/blog/Overview+of+HBase+Architecture+and+its+Components/HBase+Architecture.jpg>

HBase Database has a Mater-Slave architecture which mean slave nodes performs the tasks that is approved by the master nodes. In case a slave node fails , the master is able to figure out the node that has failed and fix that failed node but when the master node fails recovery becomes difficult. There are three main components in HBase architecture: HMaster, Regional Server and Zookeeper.

HMaster-Region servers in the Hadoop cluster are assigned regions by HMaster with the help of Zookeeper for load balancing. Responsiblities of HMaster are –

* Monitoring of the Hadoop Cluster is done by HMaster.
* Administartive tasks like creating, updating and deleting tables are performed by HMaster.
* HMaster controls failover.
* DDL operations are handled by HMaster.

Regional Server-It has nodes that handles update, write, delete, and read operations requested by clients.This process is run on each node in the hadoop cluster. Components of Region Server are–

* Block Cache – Most frequently read data is cached by Block Cache. When it is full the least recently used data is evicted.
* MemStore- Caches the data that is to be written on the disk permanently. It is the write cache. MemStore is present in every column family in a region.
* WAL(Write Ahead Log) - This file stores the new data that hasn’t been committed to the permanent storage files.
* HFile- rows are stored as sorted key values on a disk. This file is stored on HDFS and it stores the actual cells on the disk.

**Zookeeper-** It is a centralized monitoring server that behaves like a coordinator inside HBase distributed environment. The client has to approach Zookeeper first if they want to communicate with Region servers or HMaster. Some services that Zookeeper provides are –

* Track the server failure and network partitions.
* Maintains the information about configuration.

**CASSANDRA**

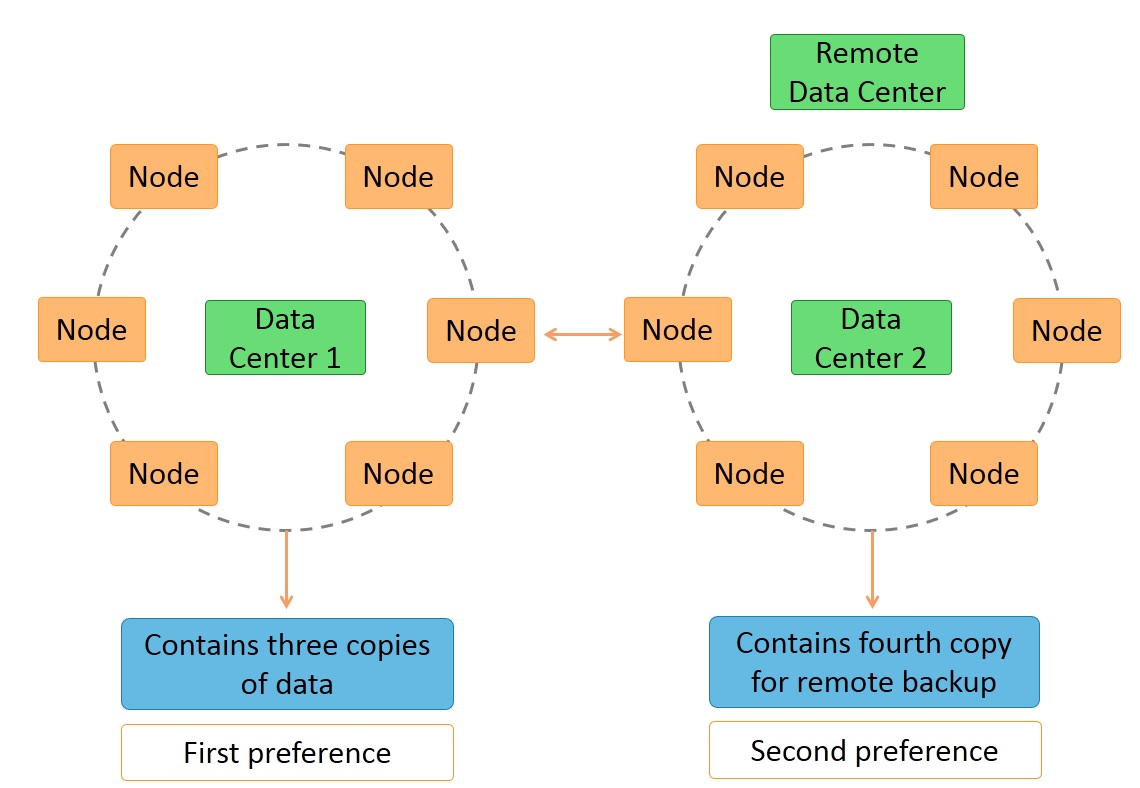


Image source- <https://www.simplilearn.com/ice9/free_resources_article_thumb/multiple-data-centers-and-data-replication-in-cassandra.jpg>

Cassandra Database follows Peer to Peer database architecture. In the Peer to Peer architecture every node is considered to be a peer and failure of a node doesn’t affect the overall performance of a system. Therefore this type of architecture provides good fault tolerance. Cassandra has the following components-

* **Node**-It is the basic component where all the data is stored.
* **Data Center**-It is a collection of related nodes.
* **Cluster**-A collection of many data centers is known as cluster.
* **Commit Log**-Write operations are written to Commit Log so that they can be used for crash recovery.
* **Mem-table**-After writing data in Commit log it is written temporarily in Mem-table.
* **SSTable**- Data is flushed to an SSTable disk file when Mem-table crosses a threshold value..

**COMAPRISON BETWEEN HBASE and CASSANDRA ON:**

**CAP** **Theorem** is a concept that states that a distributed database system can only have 2 of the 3 properties which are consistency, availability and partition Tolerance.

1. **Scalability, Availability and Reliability:**

**HBase-**HBase supports linear and modular scalability. It scales horizontally which means new regional servers can be added to the cluster to increase the processing capability and storage.  It has high availability in a standard configuration. The cluster usually consists of 1 Master and 3 or more Region Servers. The data is stored in HDFS making copies of the data that are stored in different machines and this data becomes available to every node in the cluster. If a node fails then data from failed node is redistributed to the nodes that are still running so that data is available at all times. **CAP** theorem states that HBase provides “Partition Tolerance” and “Consistency” but is not always “Available” as availability will not be there if zookeeper or all master servers fail at once, To ensure that every component is highly available we can configure one or more backup Masters. The backup Masters run on other hosts than the active Master. HBase has a high reliability. If any of the Region Server fails then the availability of data might be delayed if the failed Region Server was managing the key that was being queried at the time of failure but with proper configurations (Zookeeper and MasterServer) the data will be available again. HBase is tolerant of network failures and is consistent but it is not highly available like traditional HA database systems.

**Cassandra**-Cassandra supports linear scalability. It can scale either horizontally by adding more datacenters or vertically by adding more nodes. When a new node joins the cluster, it takes responsibility from other nodes .If a node fails then the load is spread out evenly across other nodes in the cluster. Cassandra has high availability. The clusters are in the form of a ring and each node in the ring has the same role and responsibility. Therefore, there is no single point of failure. There could be multiple data centers so that if one data center goes down the data could still be made available from other data center. Cassandra also provides replication and stores redundant copies of data across nodes in a Cassandra ring.  **CAP** theorem states that Cassandra has availability and partition tolerance but does not have consistency. Cassandra is fully distributed *,* every node in the ring shares the proportion of activity in a cluster. Since Cassandra has masterless architecture it helps in delivering lower latency in read and write operations. Cassandra supports automated backup of data/cluster using snapshots and for that it uses tools like nodetool or [opscenter](http://www.malinga.me/building-a-cassandra-cluster-with-datastax-opscenter-centos/). Data in Cassandra can be restored using those snapshots.

1. **Security:**

**HBase**-HBase provides the security features under three categories that are Authentication, Role based security and database security and logging.

1Authentication- The the options for authentication in HBase are:

1. [**Client Authentication**](https://quabase.sei.cmu.edu/mediawiki/index.php/Client_Authentication)**:** Different protocols like Kerberos and SSL are used to enable the authentication of clients.
2. [**Server authentication**](https://quabase.sei.cmu.edu/mediawiki/index.php/Server_authentication)**:** In this authentication database servers are authenticate with each other to ensure a secure operating environment. Server authentication can be done using shared keyfile.
3. [**Credential Store**](https://quabase.sei.cmu.edu/mediawiki/index.php/Credential_Store)**:** Authentication is required to keep account credentials secure. External file can be used for this.

Role based security- This section describes how roles are given authorization access to database contents.

1. [**Role Based Security**](https://quabase.sei.cmu.edu/mediawiki/index.php/Role_Based_Security)**:** Security can be defined based on the roles and it simplifies the administration.
2. [**Security Role Options**](https://quabase.sei.cmu.edu/mediawiki/index.php/Security_Role_Options)**:** Features available from role based security are default roles and custom roles.
3. [**Scope of Roles**](https://quabase.sei.cmu.edu/mediawiki/index.php/Scope_of_Roles)**:** Scope of a role is defined here. The scopes for roles provided by HBase are cluster, database, collection, and field.

Database security & logging-It describes the features for security and logging available in an HBase database server.

1. [**Database Encryption**](https://quabase.sei.cmu.edu/mediawiki/index.php/Database_Encryption)**:** Database is encrypted (partitioned) for security purpose.
2. [**Logging**](https://quabase.sei.cmu.edu/mediawiki/index.php/Logging)**:**  Event logging and fixed event logging are supported for enabling an administrator to choose the events related to security to log, and define a log flush condition so that the log does not take too much space and can be flushed out when required.

**Cassandra-**Cassandra provides the following security features-

1. [Authentication for internally controlled rolename/passwords](https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureInternalAuthenticationTOC.html)

In Cassandra authentication is given on the basis of role which is stored internally in Cassandra tables. Administrators after successfully logging in can alter, create, list, or drop roles. Superuser, non-superuser, and login privileges are used for creating roles.

2. [Authorization based on object permission management](https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureInternalAuthorizationTOC.html)

GRANT and REVOKE are two CQL commands used to give authorization permission to access Cassandra cluster operations based on role authentication.

3. [Authentication and authorization based on JMX username/passwords](https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureJMXAuthenticationTOC.html)

JMX (Java Management Extensions) authentication is done for monitoring and managing the resources related to JVM (Java Virtual Machine) instance.This authentication stores username and password in two files, one file is for passwords and one is for access. It is used by [jconsole](http://docs.oracle.com/javase/1.5.0/docs/guide/management/jconsole.html) and [nodetool](https://docs.datastax.com/en/cassandra/3.0/cassandra/tools/toolsNodetool.html).

4. [SSL encryption](https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureSSLEncryptTOC.html)

This encryption ensures that data is transmitted securely without being compromised at any point of time. Cassandra tools (cqlsh, nodetool, DevCenter) can be configured to use SSL encryption.

5. [General security measures](https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureFireWall.html)

Some ports in the firewall must be opened so that the nodes can communicate in the cluster.

**Related Work**

A study has made comparison between the traditional Databases and NOSQL Databases. The relational databases forms tables for the data that is present while NOSQL build bookshelves of elements and allows access per bookshelf. The data in those bookshelf are stored in a demoralized way i.e. in large chunks. The RDBMS can only store structured data while NOSQL can store structured, unstructured and semi-structured data. The NOSQL databases can scale automatically. All these features are making NOSQL more popular.

Comparison and Analysis of Big-Table, DynamoDB, and Cassandra has been done by a research paper. BigTable is used by Google, DynamoDB is used by Amazon and Cassandra is used by Facebook. Cassandra is the first choice of many companies unlike Big-Table and DynamoDB. Cassandra is called the daughter of Big-Table and DynamoDB as it contains some features of both the database. Big-Table can only operate on structured data while both Cassandra and DynamoDB can manage structured and unstructured data sets. Data portioning technique in Big-Table is dynamically partitioned hash tables while in both DynamoDB and Cassandra it is Consistent Hashing. Big-Table and Cassandra both have fault tolerance while DynamoDB does not have it. Scalability of Big-Table is defined as high, for DynamoDB it’s defined as incremental and for Cassandra it is linear. CAP theorem states that each of the modern database can exhibit only two out of three properties that are: Consistency (C), Availability (A) and Partition Tolerance (P).Big-Table has Consistency and Partition Tolerance while both DynamoDB and Cassandra has Availability and Partition. Big-Table has master slave architecture while Cassandra and DynamoDB has peer to peer database architecture. The results of this paper had highlighted the features and differences of all the three database.

A document from facebook employees lists the capabilities of Cassandra database and the reason it was born. Cassandra was born for resolving Facebook’s Inbox search problem. Facebook needed a scalable and reliable database for its operational needs and that is why Cassandra was made. Cassandra partitions data across the cluster using consistent hashing and the advantage of this is that the departure and arrival of a node only affects the neighbors and the other nodes remain unaffected. Cassandra uses replication for high availability. There are various replication policies like Rack Unaware, Rack Aware and Datacenter aware and any one of them can be chosen based on the requirements. Cassandra uses Zookeeper to elect a leader among its nodes. All nodes contact the leader that tells them the ranges they are replica for. Metadata about ranges of nodes responsible is stored locally at each node and in zookeeper so that if a node crashes and comes back up it will know the range it is responsible for. Cassandra uses failure detection to avoid attempts to communicate with unreachable nodes during various operations. Cassandra uses modified version of Accural failure detector that emits a value which represents the suspicion level for each monitored nodes. When a node starts for the ﬁrst time it chooses a random token for its position in the ring and this mapping is stored in disk locally and in Zookeeper. The token information is then gossiped around the cluster and that’s how we about all nodes and their respective positions in the ring. This paper has highlighted all the features that Cassandra provides.

One research paper results show that HBase performance is affected if a security strategy is implemented for securing HBase. It was found that the maximum throughput of the 12 node cluster that was using HBase native encryption (for security of HBase) was similar to the throughput produced by the unsecured 6 node cluster. Deploying HBase in the cloud would require additional security which means HBase cluster must be encrypted but such security measures affects performance or add upto the cost for using resources to improve performance. Apache Cassandra uses a much simpler secure communication architecture based on standard TLS implementations and the researchers believe that HBase security can be improved by using highly optimized TLS implementation instead of custom RPC protocols. It is realized that the security implementations of HBase should be properly examined and an efficient security implementation should be made that does not affect the HBase performance.

In one of the papers comparison between MongoDB, HBase and Cassandra was tested using YCSB The performances of these database were tested based on the effect of number of nodes, number of cores and replication factor. The results of the tests show that HBase takes advantage of the number of cores better than the other databases. MongoDB scales well initially but then it shows a plateau at around 6 cores while Cassandra is slower that HBase in absolute terms but it scales quite linearly. HBase performs well than any other database for the same number of nodes. As the replication factor becomes more than 1 the performance of MongoDB starts decreasing but there is no decrease in performance (throughput) of HBase and Cassandra.

Another paper has made comparisons between MongoDB, HBase and Cassandra using for different types of workload. The result for Read mostly operation (Workload B having ratio of 95% read and 5 % update) showed that MongoDB performs best, having lowest latency which increases with throughput because due to its support of memory mapped caching. Cassandra also performed well in this workload because of its key-row caching. HBase had high throughput in this workload but its latency was highest among the databases. For Read only operation (100% read operation) Cassandra and MongoDB performed better than HBase for latency. In Insert mostly operation (90% insert and 10% read) HBase and Cassandra both performed well (higher throughput) compared to MongoDB.

**Performance Test Plan**

YCSB (Yahoo! Cloud Serving Benchmark) tool has been used for drawing out comparison between HBase and Cassandra. It is an open source framework that evaluates and compares the performance of different data based systems. YCSB provides different types of workloads for comparison. The workloads used for Performance comparison of both the NOSQL databases are **Workload A** & **Workload D**. Test harness script is used for running the workloads that gives the throughput and latency output of the databases for making comparisons.

**Workload A: It is called an Update heavy workload** as this workload has a ratio of 50 reads and 50 writes(50/50). A session store that is recording recent actions is an example for this type of workload.

**Workload D: It is called a Read latest workload** as new records are inserted in this workload and the most recently inserted records are read(or showed). It has 5% of insert and 95% of read operations. A user updating his status in facebook and people viewing it as appears as the latest update in their home page is an example for this type of workload.

The test results would be compared by making use of latency (read, insert and update) and throughput values. Latency is the time interval between a request and response or to be generally speaking time taken to retrieve, insert or update a data. A lower latency is desirable in database. Throughput is the rate at which data request is being processed and it is usually in ops/sec. A higher throughput is desirable as it would mean larger number of requests are being processed and there are very few requests in queue that are waiting to be processed.

The opcounts value used for performing testing are: 2000, 10000, 50000, 100000 and 150000.Tableau software is used for the visualization of results received after running the testharness script. Tableau is a tool that helps in [data visualization](https://en.wikipedia.org/wiki/Visual_analytics) for making business decisions.

A virtual machine was created in Openstack and all the softwares (Hadoop, HBase, Cassandra and YCSB) that were required for performing the test were downloaded in the virtual machine. The specifications of the virtual machine created are below:

**Operating System:** Debian

**Flavour:** m1.large

**RAM:** 8GB

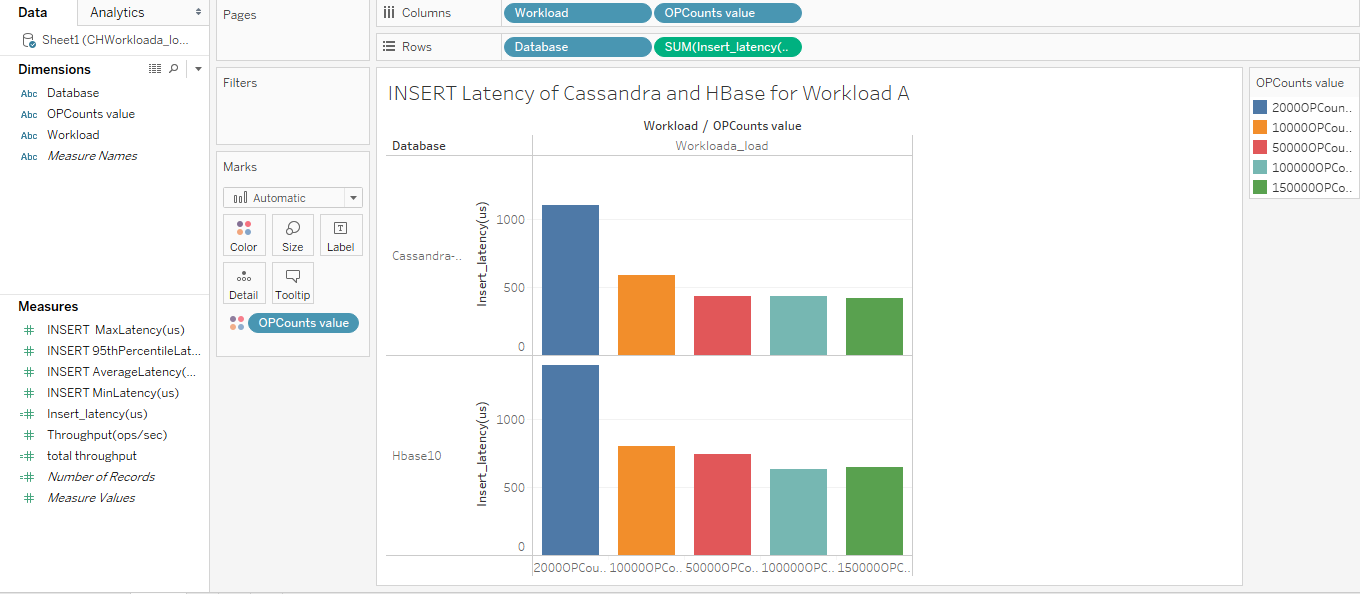
**VCPUs:** 4

**Disk:** 80 GB

**Mscdata-Net**: 192.168.100.234

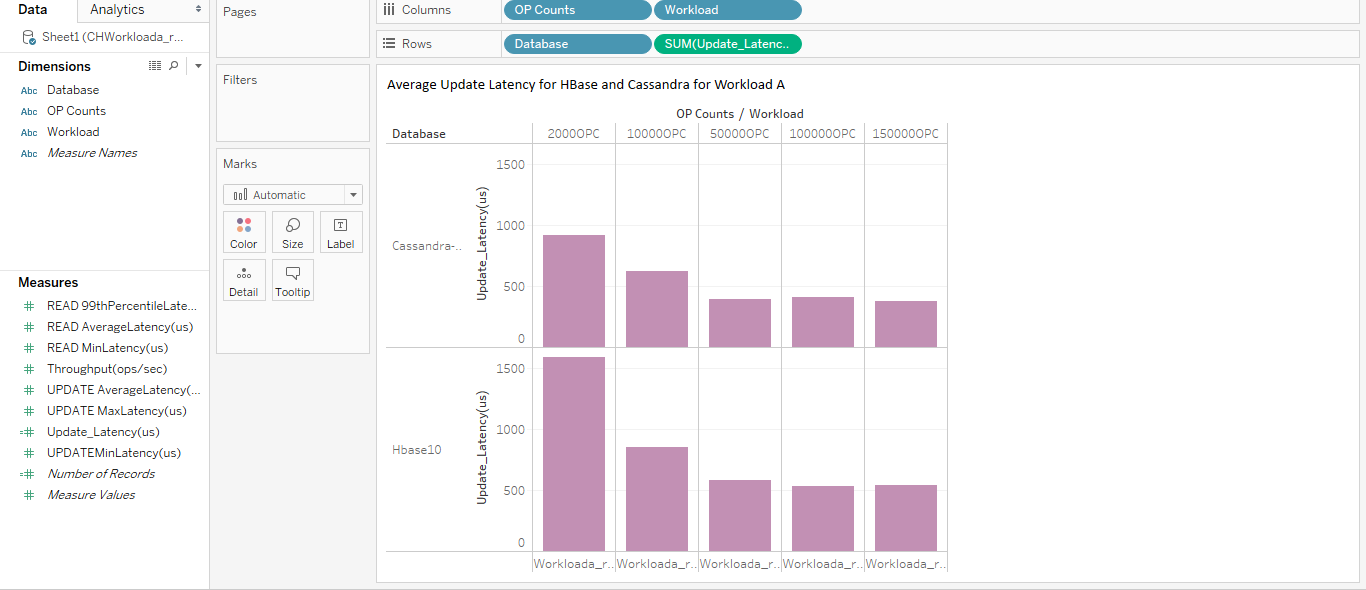
**Evaluation and Results**

**1. HBase and Cassandra performance for Average Insert Latency in Workload A**



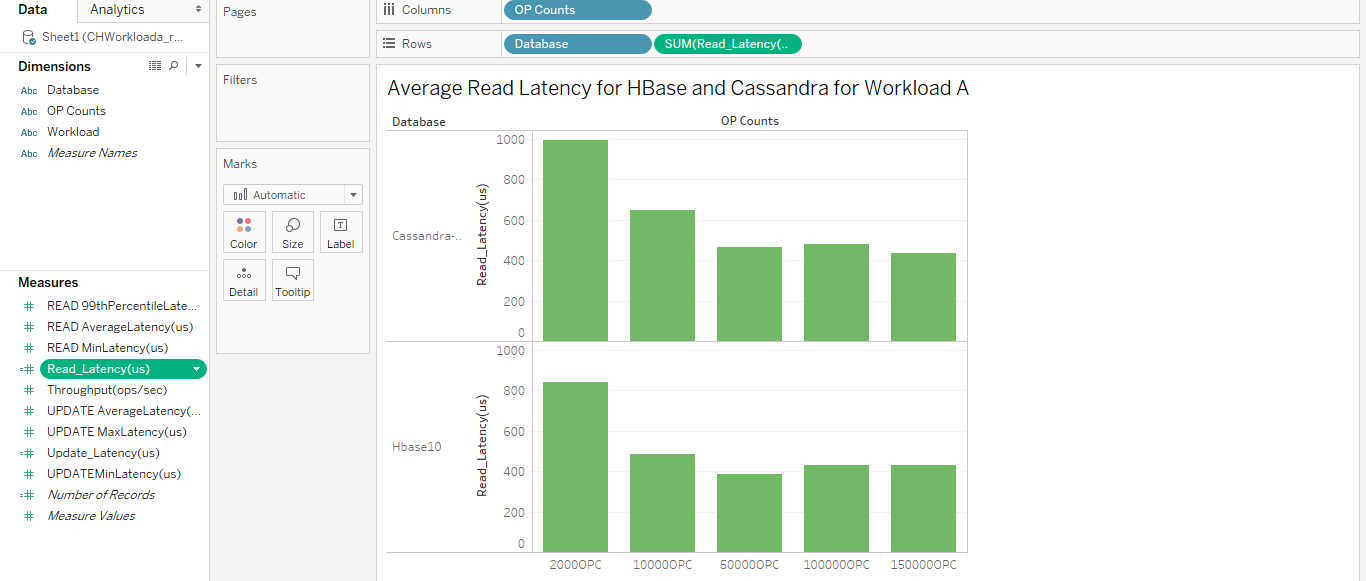
From the above visualization graph we can see that insert latency of Cassandra is lower than HBase for the same no. of operational counts. For 150k opcounts Cassandra has an Insert latency of 417.8 us while HBase has an Insert latency of 646.1 us This means that data insertion is faster in Cassandra than HBase.

**2. HBase and Cassandra performance for Average Update Latency in Workload A**



From the above visualization graph we can see that update latency of Cassandra is lower than HBase for the same no. of operational counts. For 150k opcounts Cassandra has an update latency of 374 us while HBase has an Insert latency of 538 us. This means that data is updated faster in Cassandra than HBase.

**3. HBase and Cassandra performance for Average Read Latency in Workload A**



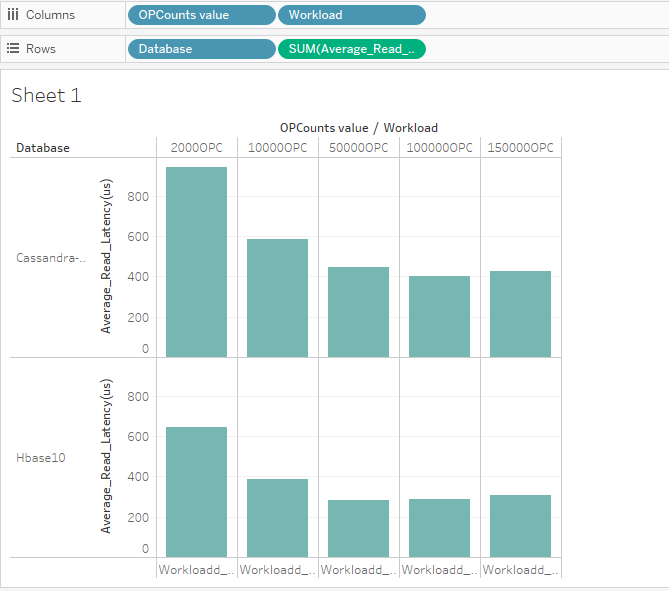
From the above visualization graph we can see that read latency of Cassandra is higher than HBase for the same no. of operational counts. For 150k opcounts Cassandra has a read latency of 436.7 us while HBase has an Insert latency of 431.7 us. This means that HBase is better than Cassandra for read operations

**4. HBase and Cassandra performance for Average Insert Latency in Workload D**



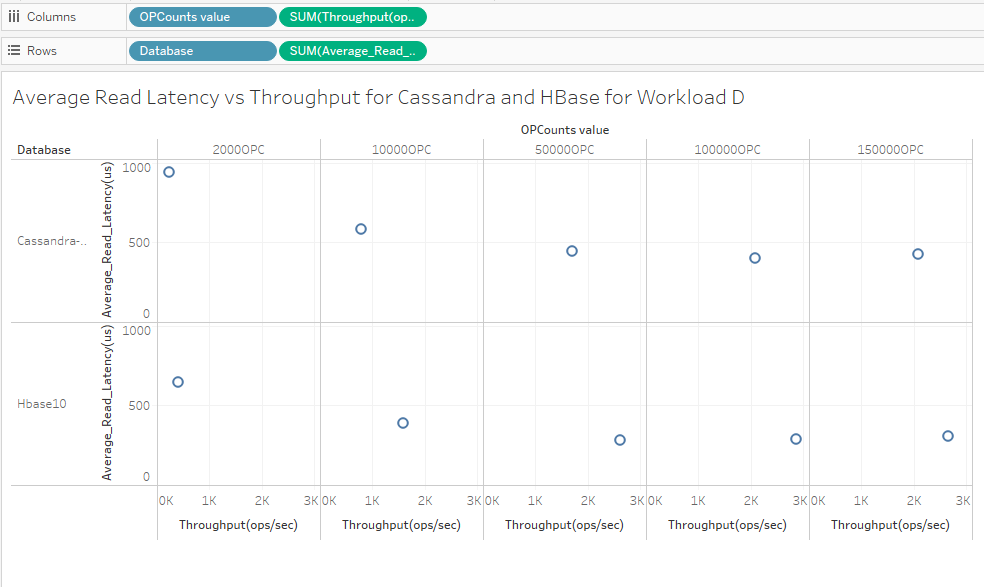
From the above visualization graph we can see that insert latency of Cassandra is lower than HBase for the same no. of operational counts. For 150k opcounts Cassandra has an Insert latency of 429.5 us while HBase has an Insert latency of 626.3 us. This means that data insertion is faster in Cassandra than HBase.

5. **HBase and Cassandra performance for Average Read Latency in Workload D**



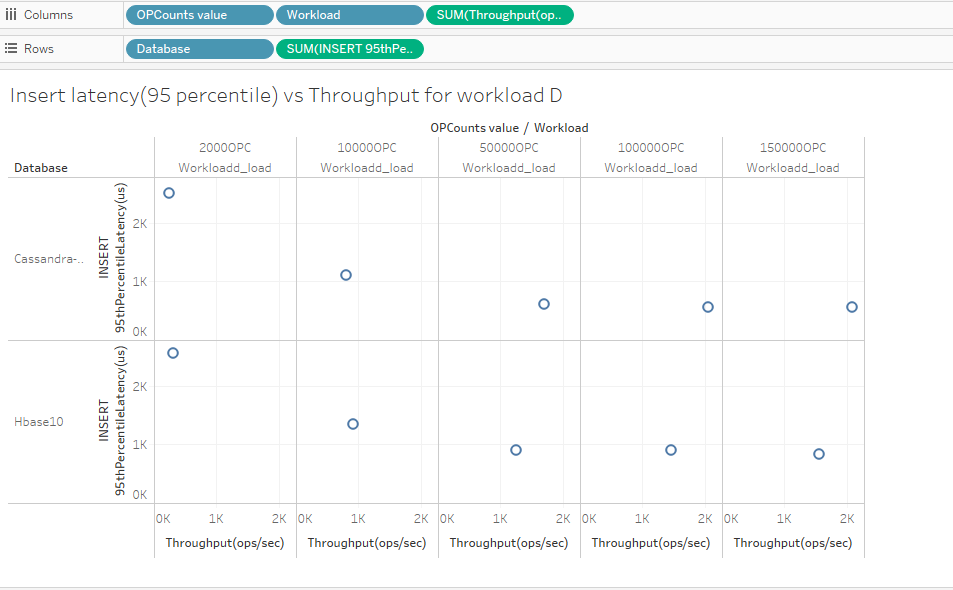
From the above visualization graph we can see that read latency of Cassandra is higher than HBase for the same no. of operational counts. For 150k opcounts Cassandra has a read latency of 425 us while HBase has an Insert latency of 305.9 us. This means that HBase is better than Cassandra for read operations.

**6. HBase and Cassandra Average Read Latency vs Throughput for Workload D**



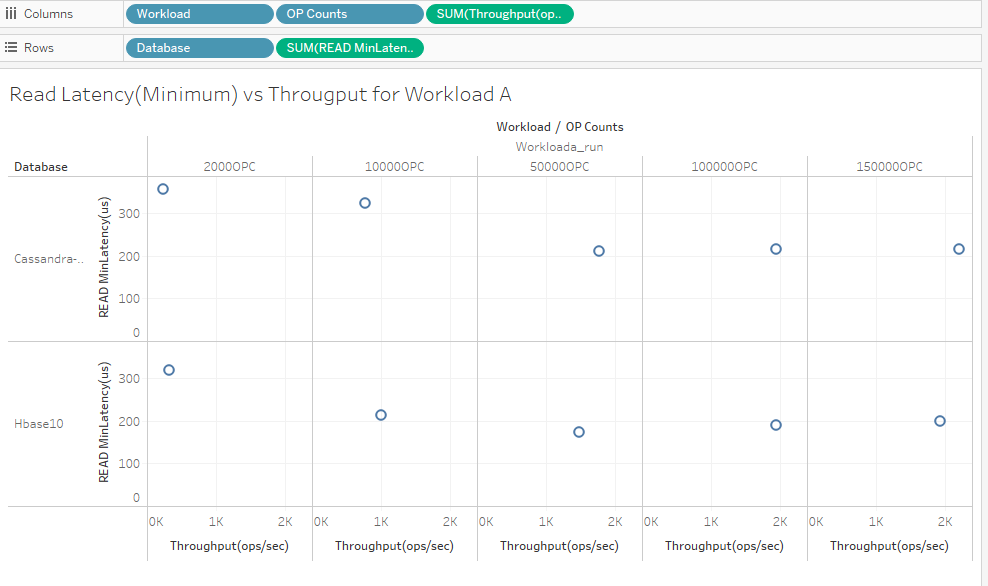
From the above Graphical representation we can see that HBase has higher throughput than Cassandra for each opcount value and the latency of Cassandra is higher than HBase. This means HBase is more efficient and better database compared to Cassandra for read operations in the scenario where there are more read operations and less insert operations.(Workload D- 95% reads and 5% insert).

**7. HBase and Cassandra Insert latency (95 percentile) vs Throughput for workload D**



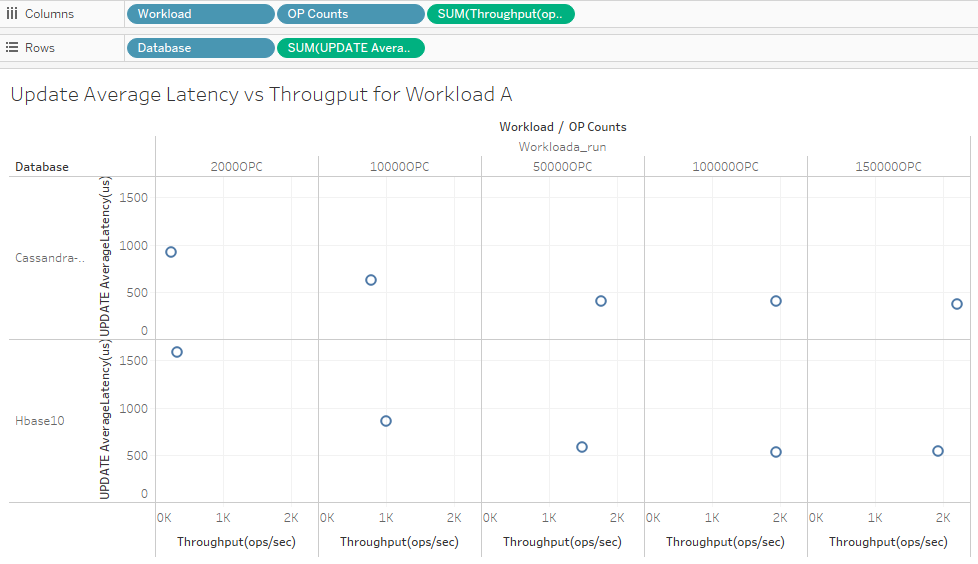
From the above Graphical representation we can see that HBase has higher throughput than Cassandra till 10k opcounts but as the opcounts increases the throughput of Cassandra becomes greater than the throughput of HBase. The latency of Cassandra is high till 10 k opcounts but as opcount increases the latency of Cassandra decreases. The performance of Cassandra for insert operations is better than HBase for higher opcounts.

**8. HBase and Cassandra Read Latency (Minimum) vs Throughput for Workload A**



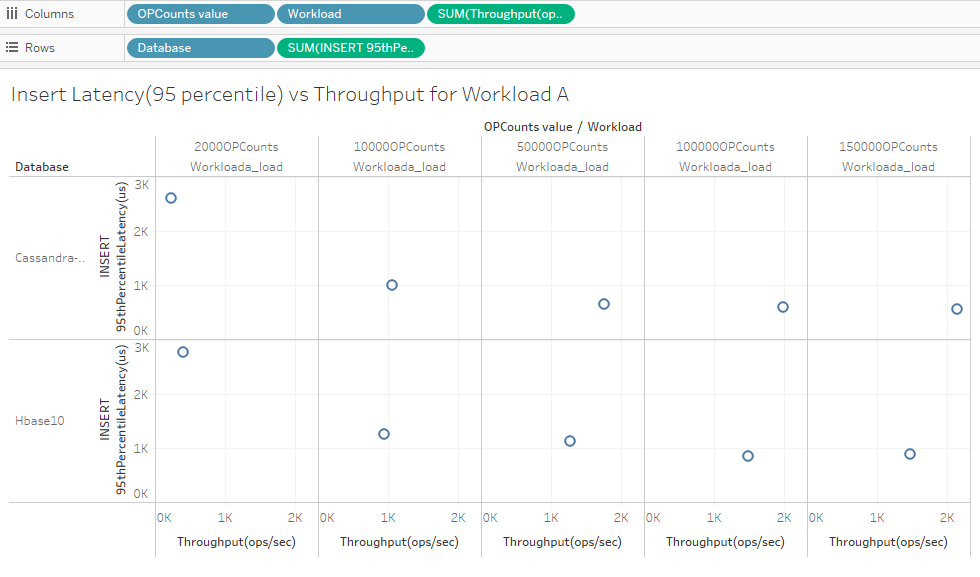
From the above Graphical representation we can see that HBase has higher throughput than Cassandra till 10k opcounts but as the opcounts increases the throughput of Cassandra becomes greater than the throughput of HBase. The latency of HBase is low compared to Cassandra for all opcounts value.

**9. HBase and Cassandra Update Average Latency vs Throughput for Workload A**



From the above Graphical representation we can see that HBase has higher throughput than Cassandra till 10k opcounts but as the opcounts increases the throughput of Cassandra becomes greater than the throughput of HBase. The latency of Cassandra is low compared to HBase for all the opcount values. Cassandra has a much better performance than HBase for update operations.

**10. HBase and Cassandra Insert Latency (95 percentile) vs Throughput for Workload A**



From the above Graphical representation we can see that HBase has higher throughput than Cassandra for 2000 opcount value but as the opcounts increases the throughput of Cassandra becomes greater than the throughput of HBase. The latency of Cassandra is low compared to HBase for all the opcount values. The performance of Cassandra for insert operations is better than HBase.

**CONCLUSION**

NOSQL databases are taking the industry by storm because of the rapidly growing data. Choosing a NOSQL database for a company can be a daunting task. While choosing a NOSQL database one has to identify the type of operations performed by their company and then choose a relatively best NOSQL database option that is suited for that types of operation. The difference between NOSQL databases can be found by their performance in different types of workloads or operations using the YCSBB tool.

In this paper HBase and Cassandra were compared for Workload A and Workload D for opcounts 2000, 10000, 50000, 100000 and 150000.For update and insert operation Cassandra’s performance was better than HBase for both workload A and Workload D while HBase performed better than Cassandra for Workload D read operation .For Workload A HBase had higher throughput than Cassandra in read operation till 10k opcounts but as the opcounts increases the throughput of Cassandra becomes greater than the throughput of HBase while latency of HBase was low compared to Cassandra for all opcounts value.

**REFERENCES**

Forbes.com. (2019). *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read*. [online] Available at: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#5b717d6e60ba [Accessed 7 May 2019].

Quabase.sei.cmu.edu. (2019). *HBase Security Features - QuABaseBD - Quality Architecture at Scale for Big Data*. [online] Available at: https://quabase.sei.cmu.edu/mediawiki/index.php/HBase\_Security\_Features [Accessed 8 May 2019].

Docs.datastax.com. (2019). *Securing Cassandra*. [online] Available at: https://docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureIntro.html [Accessed 8 May 2019].

DeZyre. (2019). *Overview of HBase Architecture and its Components*. [online] Available at: https://www.dezyre.com/article/overview-of-hbase-architecture-and-its-components/295 [Accessed 8 May 2019].

Eduonix Blog. (2019). *Learn about HBase NoSQL DB and it features*. [online] Available at: https://blog.eduonix.com/bigdata-and-hadoop/use-hbase-nosql-db/?share=google-plus-1 [Accessed 8 May 2019].

Guru99.com. (2019). [online] Available at: https://www.guru99.com/cassandra-architecture.html [Accessed 8 May 2019].

Bertozzi, M. (2019). *How Scaling Really Works in Apache HBase - Cloudera Engineering Blog*. [online] Cloudera Engineering Blog. Available at: https://blog.cloudera.com/blog/2013/04/how-scaling-really-works-in-apache-hbase/ [Accessed 8 May 2019].

RapidValue. (2019). *Cassandra – The Right Data Store for Scalability, Performance, Availability and Maintainability - RapidValue*. [online] Available at: https://www.rapidvaluesolutions.com/tech\_blog/cassandra-the-right-data-store-for-scalability-performance-availability-and-maintainability/ [Accessed 7 May 2019].

Hammood, Ali and Saran, Murat. (2016). ‘A Comparison of NoSQL Database Systems: A Study on MongoDB, Apache Hbase, and Apache Cassandra’, 2016 International Conference on Computer Science and Engineering.

Trumboo, Noor Owais and Nisar, Jasra. (2018). ‘Traditional Databases vs NOSQL’, 2018 International Journal of Trend in Scientific Research and Development (ijtsrd).

Khalid, S., Syed, A., Mohammad, A. and Halgamuge, M. (2019).’Big-data NoSQL databases: A comparison and analysis of Big-Table, DynamoDB, and Cassandra’, 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), pp. 89-93.

Lakshman, Avinash and Malik, Prashant. (2010). ‘Cassandra - A Decentralized Structured Storage System’, 2010 ACM SIGOPS Operating Systems Review, pp. 35-40.

Pallas, Frank, Gunther, Johannes and Bermbach, David. (2016).’ Pick your choice in HBase: Security or performance’, 2016 IEEE International Conference on Big Data (Big Data).

Kumar S, Rajith and Mary, Rosaline R. (2017).’ Comparative Performance Analysis of various NoSQL Databases: MongoDB, Cassandra and HBase on Yahoo Cloud Server’, 2017 Imperial Journal of Interdisciplinary Research (IJIR).