ISSN 2349-0292
Impact Factor 2.675

GLOBAL JOURNAL OF ADVANCED ENGINEERING TECHNOLOGIES AND SCIENCES

PERFORMANCE ANALYSIS OF STORAGE PROBLEMS IN BIG DATA USING COLUMN ORIENTED NOSQL DATABASES APPROACH

Zaharaddeen Karami Lawal*
deenklawal13@gmail.com
* Department of Computer Science and Engineering, Jodhpur National University, Jodhpur Rajasthan

ABSTRACT

A column-oriented database management system stores data tables as segments of columns of data instead of rows, similar to relational database management systems. There has been a significant amount of excitement and recent work on column-oriented database systems. These databases show good performance of an order of magnitude better than traditional row-oriented database systems on analytical workloads such as those found in data warehouses, decision support, and business intelligence applications.

The aim of this research work is to evaluate the two most commonly used column oriented databases in order to show their capability in tackling Big Data storage challenges. These databases are Cassandra and HBase. The databases are also compared in terms of insert, latency, reads and writes performance taking into consideration the typical workloads. The comparison allows users to choose the most appropriate database according to the specific mechanisms and application need, but in some features one outperforms other.

KEYWORDS: Availability, Big Data, Consistency, Database, NoSQL, Storage, Partition-Tolerance.

1. Introduction

In earliest days of computing. "Big Data" originally meant the volume of data that could not be processed (efficiently) by traditional database methods and tools. Each time a new storage medium was invented, the amount of data accessible exploded because it could be easily accessed. The original definition focused on structured data, but most researchers and practitioners have come to realize that most of the world's information resides in massive, unstructured information, largely in the form of text and imagery. The explosion of data has not been accompanied by a corresponding new storage medium [1] [2].

Big Data can be defined as the amount of data just beyond technology's capability to store, manage and process efficiently. As little as 8 years ago, we were only thinking of tens to hundreds of gigabytes of storage for our personal computers. Today, we are thinking in tens to hundreds of terabytes. Thus, big data is a moving target. Put another way, it is that amount of data that is just beyond our immediate grasp, e.g., we have to work hard to store it, access it, manage it, and process it. The current growth rate in the amount of data collected is staggering [1].

Enormous amount of data is generated every minute. A recent study estimated that every minute, Google receives over 4 million queries, e-mail users send over 200 million messages, YouTube users upload 72 hours of video, Facebook users share over 2 million pieces of content, and Twitter users generate 277,000 tweets [3].
The Big Data size ranges from terabytes (10^{12}). Around 2.5 quintillion (10^{18}) bytes of new data is created. Almost 90% of world's data has been created in just last 3 to 4 years [1].

2. Characteristics of Big Data

One view, espoused by Gartner's Doug Laney describes Big Data as having three dimensions: volume, variety, and velocity. Thus, IDC defined it: "Big data technologies describe a new generation of technologies and architectures designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis" [1] [4].

Although IBM defines Big Data with four characteristics by adding veracity to the above 3 characteristics [1]. Other Researchers defined it with five Vs with the addition of Value.

i. Volume

The first characteristic of Big Data, which is "Volume", refers to the quantity of data that is being manipulated and analyzed in order to obtain the desired results. It represents a challenge because in order to manipulate and analyze a big volume of data requires a lot of resources that will eventually materialize in displaying the requested results [5]. For instance, a computer system is limited by current technology regarding the speed of processing operations. The size of the data that is being processed can be unlimited, but the speed of processing operations is constant.

ii. Velocity

Data velocity measures the speed of data creation, streaming, and aggregation. Ecommerce has rapidly increased the speed and richness of data used for different business transactions (for example, web-site clicks). Data Variety: Data variety is a measure of the richness of the data representation – text, images video, audio, etc [1] [5].

iii. Variety

Data variety is a measure of the richness of the data representation – text, images video, audio, etc. From an analytic perspective, it is probably the biggest obstacle to effectively using large volumes of data. Incompatible data formats, non-aligned data structures, and inconsistent data semantics represents significant challenges that can lead to analytic sprawl [2].

iv. Veracity

The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data.

v. Value

Data value measures the usefulness of data in making decisions. It has been noted that "the purpose of computing is insight, not numbers". Data science is exploratory and useful in getting to know the data, but "analytic science" encompasses the predictive power of big data [2].

3. Big Data Storage Challenges

Now a day there are many challenges that big data is facing at storage level. These challenges are availability, consistency, flexibility, heterogeneity, partition-tolerance, reliability and scalability. There is a need to provide a data store that can handle the challenges of Big Data.

4. NoSQL Database

The term NoSQL was first introduced in 1998 for a relational database that restricted the use of SQL. The term was emerged again in 2009 and used for conferences of advocates on non-relational databases. These conferences state that especially Web 2.0 start-ups have begun their business without Oracle [7] and MySQL [8]. Instead, they built their own data stores influenced by Amazon's Dynamo and Google's Bigtable in order to store and process huge amounts of data. Most of these data stores are open source software. For example, Cassandra which is a column-oriented database originally developed for a new search feature by Facebook is now part of the Apache Software Project.

Different types of data stores satisfy different needs and hence various non-relational databases are classified into key-value, columnar, document-oriented and graph-based databases. Document oriented databases gain an immense ease-of-use, while the key-value and column oriented databases make it easier to distribute data over clusters of computers. Graph-bases databases are preferred in applications where relationships amongst data are as important as data.

4.1 Replication

Replication in the case of distributed databases means that a data item is stored on more than one node. This is very useful to increase read performance of the database, because it allows a load balancer to distribute all read operations over many machines. It is also very advantageous that it makes the cluster robust against failures of single nodes. If one machine fails, then there is at least another one with the same data which can replace the lost node.

Sometimes it is even useful to replicate the data to different data centers, which makes the database immune against catastrophic events in one area. This is also done to get the data closer to its users, which decreases the latency.

But the downside of data replication are the write operations. A write operation on a replicated database has to be done on each node that is supposed to store the respective data item. A database has basically two choices for doing this: Either a write operation has to be committed to all replication nodes before the database can return an acknowledgment. Or a write operation is first only performed on one or a limited number of nodes and then later send asynchronously to all the other nodes.

ISSN 2349-0292
Impact Factor 2.675

The choice of one of this two options decides the availability and consistency properties of the database, which will be explained later in this work with Brewer's CAP Theorem.

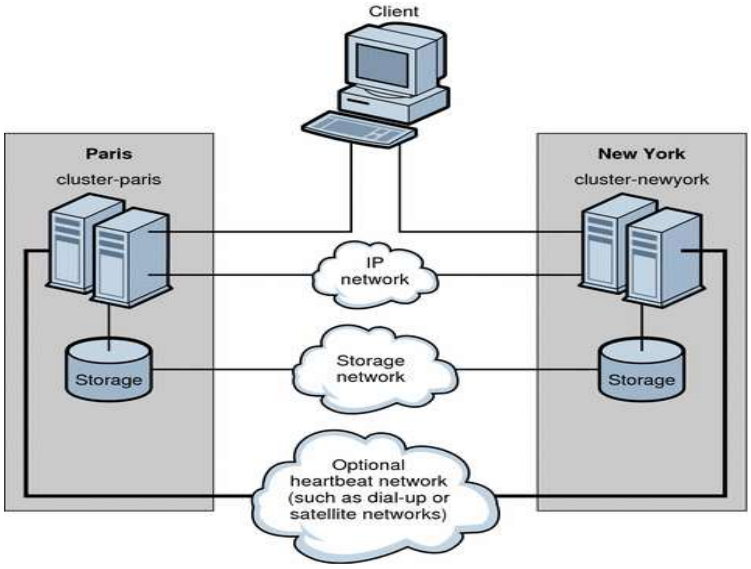


Figure 1 Replication

4.2 Sharding

The term Sharding derives from the noun 'shard' as it means that the data inside a database is splitted into many shards, which can be distributed over many nodes. The data partitioning can, for example, be done with a consistent hash function that is applied to the primary key of the data items to determine the associated shard.

This implicates that a table (if the database uses a concept comparable to tables) is not stored on one single machine, but in a cluster of nodes. Its advantage is that nodes can be added to the cluster to increase the capacity and the performance of write and read operations without the need to modify the application. It is even possible to reduce the size of a sharded database cluster when the demands decreases.

The downside of sharding is that it makes some typical database operations very complex and inefficient. One of the most important operations in relational databases is the join operator, which is used to materialize the relations of data items. A join operation works on two sets of data items, a left one and a right one, which are connected with a pair of attributes. A distributed join in a sharded database would require that the database would have to search in the right set for all items that are associated to each item in the left set. This would require many requests to all machines that store data items from one of the two sets, which would cause a lot of network traffic. Because of this, most sharded databases do not support join operations.

The more nodes are used inside a sharded database cluster then the more the probability that increase in one of the machines or one of the network connections failure may not lead to the entire system failure. Therefore, sharding is often combined with replication, which makes the cluster more robust against hardware failures.

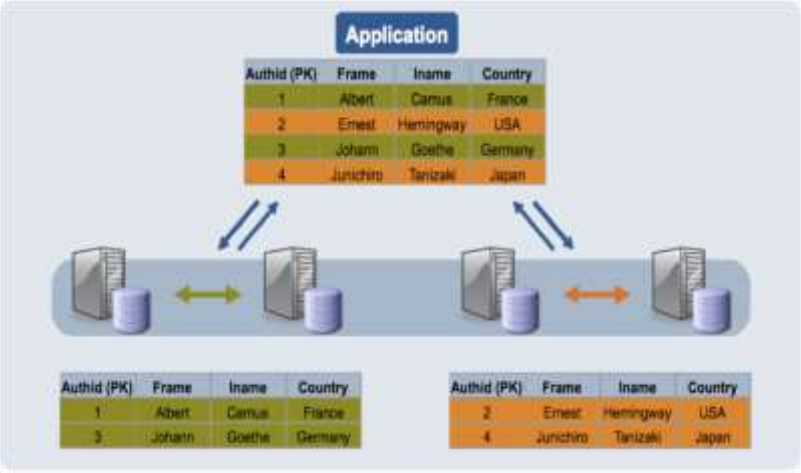


Figure 2 Sharding

4.3 Characteristics of NoSQL Databases

In order to guarantee the integrity of data, most of the classical database systems are based on transactions. This ensures consistency of data in all situations of data management. These transactional characteristics are also known as ACID (Atomicity, Consistency, Isolation, and Durability).

- **Atomicity**: Either all parts of a transaction must be completed or none.
- **Consistency**: The integrity of the database is preserved by all transactions. The database is not left in an invalid state after a transaction.
- **Isolation**: A transaction must be run isolated in order to guarantee that any inconsistency in the data involved does not affect other transactions.
- **Durability**: The changes made by a completed transaction must be preserved or in other words be durable.

However, scaling out of ACID-compliant systems has shown to be a problem. Conflicts are arising between the different aspects of high availability in distributed systems that are not fully solvable - known as the CAP-theorem.

4.4 CAP Theorem

Brewer's CAP theorem explains various possible trade-offs. The theorem states that it is difficult for a web service to provide guarantee for consistency, availability and partition tolerance at the same time.

- **Consistency:** means all the nodes of a system should have the same data.
- **Availability:** means that the data should be available all the time even if some nodes are failed.
- **Partition tolerance:** makes the system continues to operate despite arbitrary message loss.

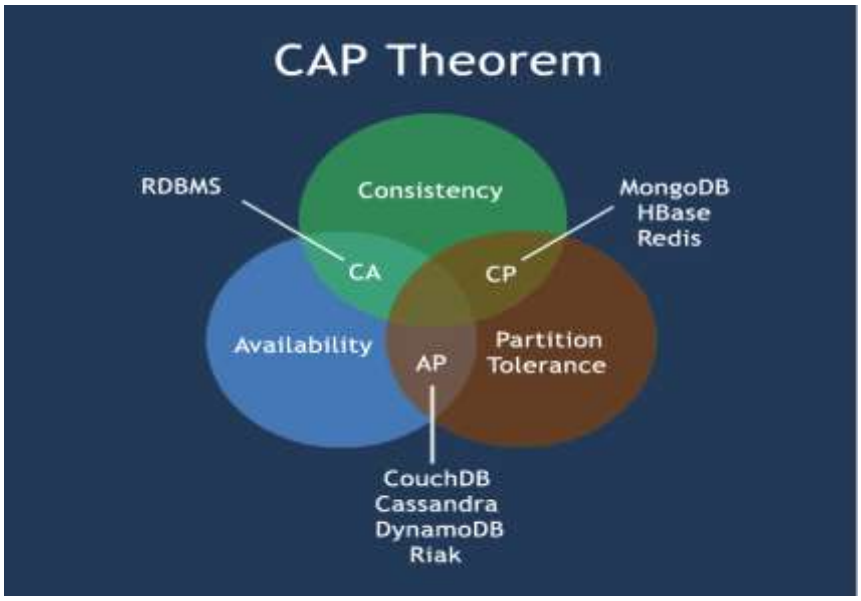


Figure 3 CAP Theorem

In web applications based on horizontal scaling strategy, it is necessary to decide between C and A that is Consistency and Availability. Usually DBMSs prefer Cover A and P. There are two directions in deciding whether C or A. One of them requires strong consistency as a core property and tries to maximize availability.

The advantage of strong consistency, that is ACID transactions, means to develop applications and to manage data services in more simple way. On the other hand, complex application logic has to be implemented, which detects and resolves inconsistency.

The second direction prioritizes availability and tries to maximize consistency. Priority of availability has rather economic justification. Unavailability of a service can imply financial losses. Existence of 2 phase commit protocol ensures consistency and atomicity from ACID. Then, based on the CAP theorem, availability cannot be always guaranteed and for its increasing it is necessary to relax consistency, that is when the data is written, not everyone, who reads something from the database, will see correct data. This is called eventual consistency or weak consistency. Such transactional model uses BASE (Basically Available, Soft state, Eventually Consistent) model.

ISSN 2349-0292
Impact Factor 2.675

4.5 BASE

BASE is an acronym for Basically Available Soft-state services with Eventual-consistency. BASE is associated with No SQL data stores that focuses on Partition tolerance and availability and literally chucks consistency out in order to achieve better partitioning and availability. Although relational databases are very mature, but upcoming NoSQL databases have few advantages over traditional relational databases which are listed below:

1. Relational databases follow strict data consistency. But the ACID properties implemented by RDBMSs might be more than necessary for particular applications and use cases. As an example, Adobe's Connect Now holds three copies of user session data; these replicas do not neither has to undergo all consistency checks of a relational database management systems nor do they have to be persisted. Hence, it is fully sufficient to hold them in memory.
2. NoSQL databases provide significantly higher performance than traditional RDBMSs. For example, the column-store Hypertable which follows Google's Bigtable approach allows the local search engine to store one billion data cells per day. Take another example, Google' MapReduce approach is able to process 20 peta byte a day stored in Bigtable.
3. NoSQL databases are designed in a way that PC clusters can be easily and cheaply expanded without the complexity and cost of sharding which involves cutting up databases into multiple tables to run on large clusters or grids.
4. In contrast to relational database management systems most NoSQL databases are

designed to scale well in the both directions and not rely on highly available hardware. Machines can be added and removed without causing the same operational overhead involved in RDBMS cluster-solutions.

4.6 Classes of NoSQL Databases

NoSQL databases can be categorized into four classes: Key-value databases, which implement a key to value persistent map for data retrieval, Column-family databases, which is inspired by BigTable model of Google, Document oriented databases, which are oriented to store semi-structured data and Graph Databases, which are oriented to store data in graph-like structures.

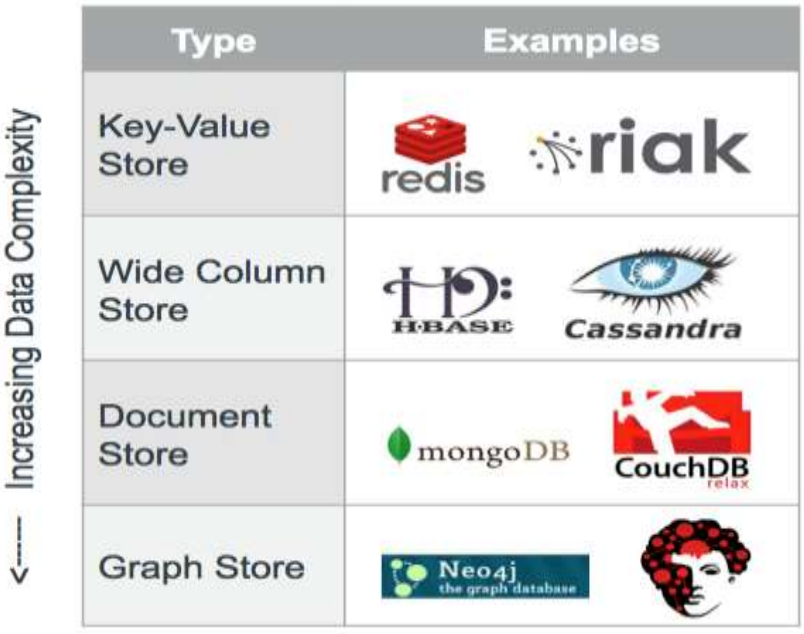


Figure 4 NoSQL Database Classes

i. Key-Value Databases

Key-value databases have very simple data model. Rather than tables or semi-structured documents, data is just organized as an associative array of entries. A unique key is used to identify a single entry and all of the three available operations i.e. delete the given key, change the entry associated with the key and insert a new key, uses this key. While the key-value data store looks and acts like an associative array, it may rely on tables, indexes and other artifacts of relational systems to be efficient in practice. Key-value data stores use a data model similar to the popular memcached (distributed in-memory cache), with a single key-value index for all the data. Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering. Key value data stores allow the application to store its data in a schema-less way [7]. The data could be stored in a data-type of a programming language or an object.) ([9], [7]).

Popular Key-value databases are Dynamo [10], SimpleDB [11] Redis [12]. Redis is a new project on Key-value memory database. Redis load entire data Into memory for faster access and save the data asynchronously to the hard disk.

ii. Graph Databases

Leading the innovation in Web 3.0 are websites like Facebook, Twitter and LinkedIn. The amount of data generated by these websites in a month is several terabytes. To traverse this huge data various scientific communities have come together to give a common solution. Graph databases like Neo4j [13]and FlockDB are the end results which provides traversal of millions of nodes in milliseconds which otherwise would have taken hours in traditional RDBMS. In graph based data stores, instead of tables of rows and columns and the rigid structure of SQL, a flexible graph model is used which can scale across multiple machines. These kinds of databases are for data whose relations are well represented with a graph-style that is where elements are interconnected with an undetermined number of relations between them. In graph based data store, every element contains a direct pointer to its adjacent element and no index look-ups are necessary [7]. General graph databases that can store any graph are distinct from specialized graph databases such as triple stores and network databases. Most of the important information is stored in the edges [14].

iii. Document Databases

Document store databases refers to databases that store their data in the form of documents. Document stores encapsulate key value pairs within documents, keys have to be unique. Every document contains a special key "ID", which is also unique within a collection of documents and therefore identifies a document explicitly. In contrast to key value stores, values are not opaque to the system and can be queried as well. Documents inside a document-oriented database are somewhat similar to records in relational databases, but they are much more flexible since they are schema less. The documents are of standard formats such as XML, PDF, JSON etc. In relational databases, a record inside the same database will have same data fields and the unused data fields are kept empty, but in case of document stores, each document may have similar as well as dissimilar data. Documents in the database are addressed using a unique *key* that represents that document. Storing new documents containing any kind of attributes can as easily be done as adding new attributes to existing documents at runtime. The most prominent document stores are CouchDB, MongoDB, RavenDB. CouchDB and RavenDB do in fact store their data in JSON. MongoDB uses a twist on JSON called Binary JSON (BSON) that's able to perform binary serialization. Document oriented databases should be used for applications in which data need not be stored in a table with uniform sized fields, but instead the data has to be stored as a document having special characteristics document stores should be avoided if the database will have a lot of relations and normalization.

iv. Column-oriented Database

Column-oriented data stores were created to store and process very large amounts of data distributed over many machines. There are still keys but they point to multiple columns. The columns are arranged by column family, where a column family tells how the data is stored on the disk. All the data in a single column family will reside in the same file. A column family can contain columns or super columns, whereas super column is a dictionary; it is a column that contains other columns. These extensible record stores have been motivated by Google's success with BigTable, where BigTable is a NoSQL data store introduced by Google in 2006 [15]. Basic data model is rows and columns, and basic scalability model is splitting both rows and columns over multiple nodes.

A column family is a container for rows, analogous to the table in a relational system. Each row in a column family can reference by its key. Columns and super columns in a column database are spare, meaning that they take exactly 0 bytes if they do not have a value in them. Rows are split across nodes through sharding on the primary key [16].

A column-oriented database serializes all of the values of a column together, then the values of the next column, and so on. In this layout, any one of the columns more closely matches the structure of an index in a row-based system. This causes confusion about how a column-oriented store is really just a row-store with an index on every column. However, it is the mapping of the data that differs dramatically. In a row-oriented indexed system, the primary key is the row id that is mapped to indexed data. In the column-oriented system primary key is the data, mapping back to row ids [17].

Column oriented databases show good performance of an order of magnitude better than traditional row-oriented database systems on analytical workloads such as those found in data warehouses, decision support, and business intelligence applications. The reason behind this performance difference is that the column-stores are more I/O efficient for read-only queries since they only have to read from disk those attributes accessed by a query. By denoting one as column oriented, we are referring to both the ease of expression of a column-oriented structure

and the focus on optimizations for column-oriented workloads. This approach is in contrast to row-oriented or row store databases and with correlation databases, which use a value-based storage structure.

Column-oriented data stores are the preferred method for storing time series data in many applications in capital markets. For example, they are excellent choices for storing tick data from stock exchanges, where tick data refers to market data which shows the price and volume of every print, also often includes information about every change to the best bid and ask. The leading technologies in this area are Google's BigTable [14] [18] and Cassandra [19], originally developed by Facebook.

4.5.1 Available Column-oriented Databases

There are many leading technologies in column-oriented databases but most common are BigTable, HBase and Cassandra. HBase is a smaller version of BigTable. Out of these column-oriented databases, HBase and Cassandra are discussed in following section.

HBase

HBase is an open source, non-relational and distributed database [36]. It is developed as part of Apache Hadoop project [37] and runs on top of Hadoop distributed file system. It is a distributed, persistent, strictly consistent storage system with near-optimal write and provides excellent read performance and makes efficient use of disk space by supporting pluggable compression algorithms that can be selected based on the nature of the data in specific column families.

HBase can run on a cluster of computers instead of a single computer without any major issue. It is possible to scale horizontally by adding more machines to the It is a cluster. Each node in the cluster provides a bit of storage, a bit of cache, and a bit of computation as well. HBase is also incredibly flexible and distributive. All Nodes are same, so it is simply to replace damaged node with another node. Column-family database modelled after Google's BigTable [18] and has BigTable like capabilities. It is often described as a sparse, persistent, multidimensional map, which is indexed by row key, column key, and timestamp. It is also referred as a key value store and sometimes a database storing versioned maps of maps.

HBase can handle both structured and semi-structured data naturally. It can store unstructured data too, as long as it is not too large [22]. It does not care about types and allows for a dynamic and flexible data model that does not constrain the kind of data to be stored. It does not create any problem in storing an integer in one row and a string in another for the same column.

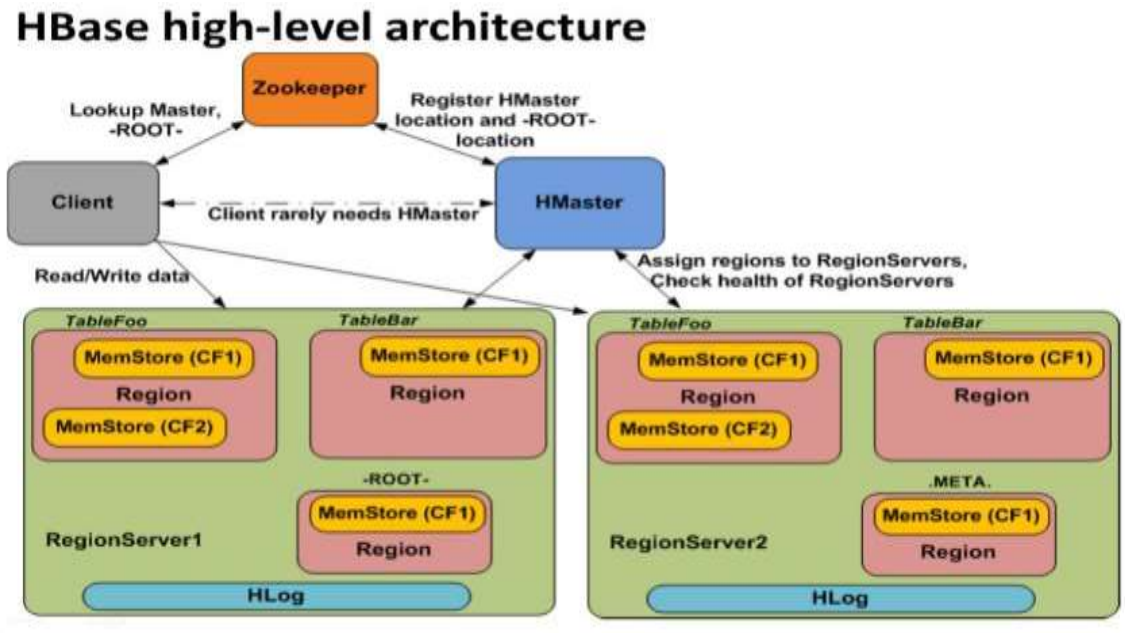


Figure 5 Hbase Architecture

Cassandra

Apache Cassandra is an open source distributed database management system. It is an Apache Software Foundation top-level project designed to handle very large amounts of data spread across many commodity servers

while providing a highly available service with no single point of failure. Cassandra provides a structured key-value store with tunable consistency [23]. Keys map to multiple values, which are grouped into column families. The column families are fixed when a Cassandra database is created, but columns can be added to a family at any time. Furthermore, columns are added only to specified keys, so different keys can have different numbers of columns in any given family. The values from a column family for each key are stored together. Each key in Cassandra corresponds to a value which is an object. Each key has values as columns, and columns are grouped together into sets called super columns. Also, each super column can be grouped in super column families. In figure 2.3, organization of data in Cassandra has been illustrated. In Cassandra, super column family contains one or more super columns. Each super column contains zero or more columns and is organizationally equivalent to an HBase column family [14].

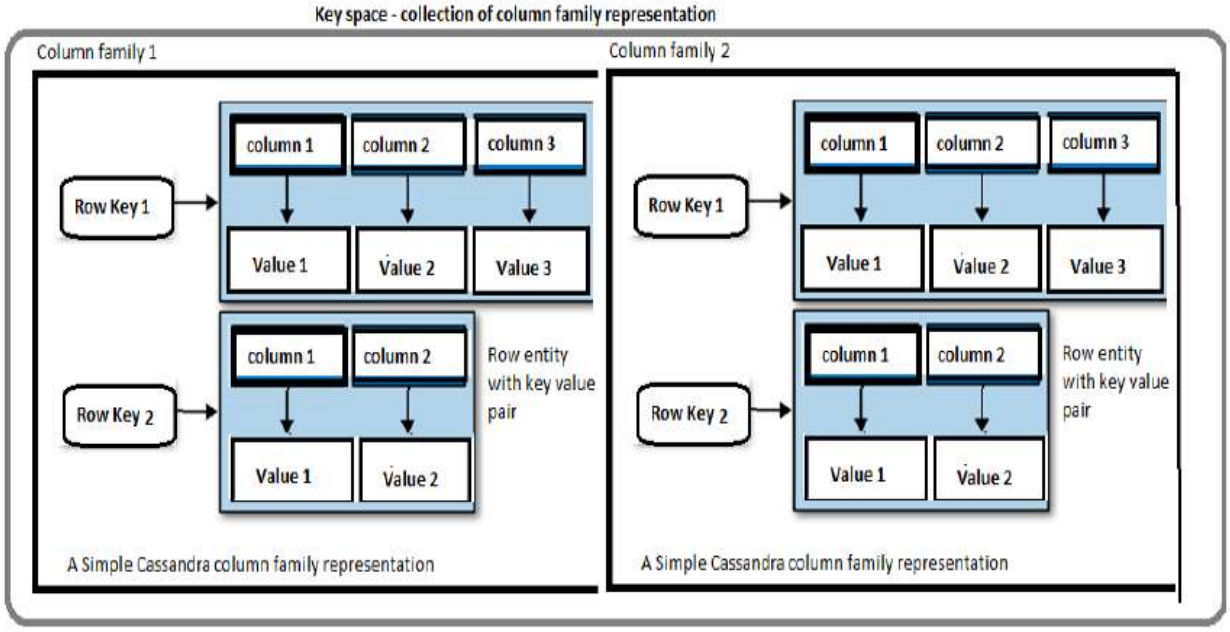


Figure 6 Cassandra data model

5. Comparison between Hbase and Cassandra

HBase and Cassandra have been compared in terms of consistency, performance, availability, etc. Although both are popular and widely used databases but in some features one outperforms other. Also this comparison table will help anyone in difficulty to choose between HBase and Cassandra.

Table 1 Comparison between HBase and Cassandra

FEATURE	CASSANDRA	HBASE
Architecture	Peer to peer architecture model	Master Slave architecture model
Consistency	Tunable Consistency. Read and write consistency levels can be set	Strict consistency (focuses mainly on consistency according to cap theorem).
Availability	Very high availability (focuses mainly on availability according to cap theorem),because N-Replicas are available across nodes	Failover clustering to provide availability in case of master node failure.
Partitioning Strategy	Supports partitioning (random partitioner, byte order partitioner)	Hbase regions provides range partitioning.
Data Model	Keyspace – column family	Regions-column family
Replication	Replication strategy can be defined by setting Replication Factor	Hbase has Replication scope (0-disabled 1- enabled). HDFS has replication factor
Fault Tolerance	No single point of failure with peer to peer architecture	Single point of failure in master slave approach. Can be overcome by failover

		clustering.
Cluster Communication	Cassandra uses gossip protocol for inter node Communication	Apache Zookeeper is responsible for Hbase node co-ordination.
Writes	Very fast writes	Writes slower than
Performance	because of peer to peer architecture and cassandra data model	cassandra if it uses pipelined writes (synchronous). Asynchronous writes are configurable
Reads performance	Performance based on consistency level (decreases in performance with increase in consistency level) and replication factor.	Follows strict consistency model and are optimized for reads. Very fast reads in Hbase with Hadoop support.
Durability	Achieved using a commit log	Achieved using Write Ahead Log (WAL)
Concurrency	Row level locking.	Row level locking.
Aggregate Functions	No support for aggregate and group by functions	Supports aggregate functions via hive.
Indexing technique	Hash indexes	LSM trees that are similar to b trees
Map Reduce	Can support map reduce integration with Hadoop.	Very good support for map reduce because of HDFS.
CAP Theorem Focus	Consistency, Availability	Availability, Partition-Tolerance
Optimized For	Reads	Writes
Main Data Structure	CF, RowKey, Name value pair set	CF, RowKey, Name value pair set
Dynamic Columns	Possible	Possible
Column Names as Data	Allowed	Allowed
Static Columns	Not allowed	Allowed
RowKey Slices	Possible	Not Possible
Cell Versioning	Supported	Not Supported
Rebalancing	Automatic	Not needed with random Partitioning
Data Node Failure	Graceful degradation	Graceful degradation
Data Node Restoration	Same as node addition	Requires node repair admin-action
Data Node Addition	Rebalancing automatic	Rebalancing requires token assignment adjustment
Data Node Management	Simple (Roll In, Role Out)	Human admin action required
Cluster Admin Nodes	Zookeeper, Name Node, HMaster	All Nodes are Equal
Compression	Support	Support

In most cases, HBase is the most preparable choice over Cassandra as it is relatively more consistent and mature platform. HBase is not restricted to only traditional HDFS but can change underlying storage depending on the needs. Native map reduce is a concept of simpler schema that can be modified without a cluster restart in HBase that is not possible in Cassandra. Cassandra can be used for applications requiring faster writes and high availability.

ISSN 2349-0292
Impact Factor 2.675

From the above comparisons table user can select a database of choice based on the features or functionalities required.

6. Performance Analysis of Hbase and Cassandra

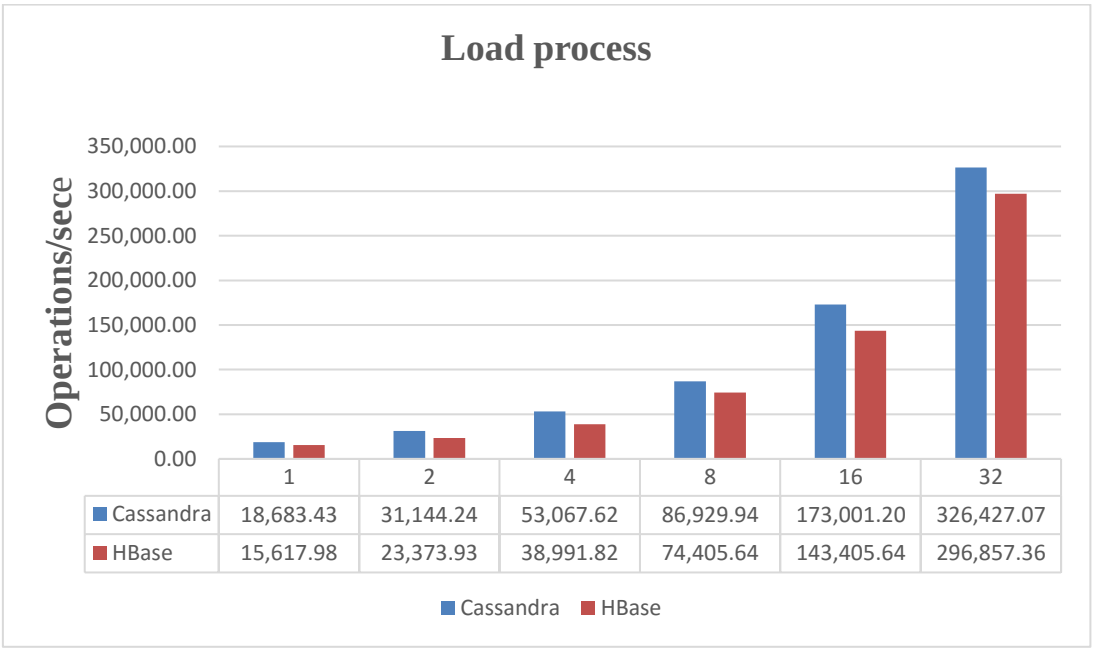
YCSB is used as a benchmark tool. It provides its results in a fairly structured text format for each workload. For each workload the overall throughput in operations/second and the total test runtime were gathered, along with the operation specifics in average latency.

Throughput by Workload .

Each workload appears below with the throughput/operations-per-second (**more is better**) graphed vertically, the number of nodes used for the workload displayed horizontally, and a table with the result numbers following each graph.

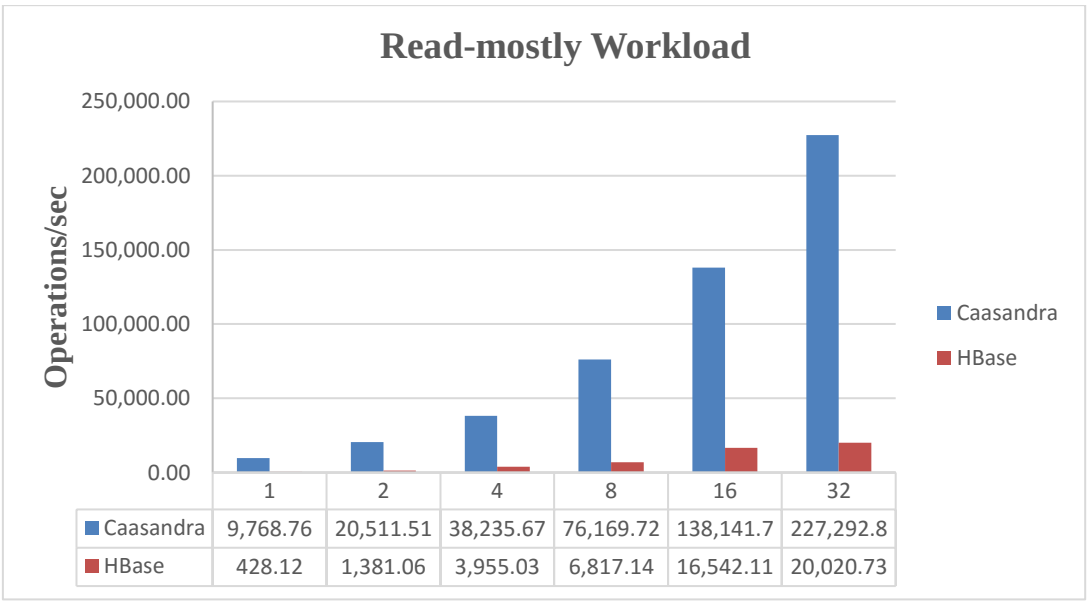
Load process

Bulk load was done ahead of each workload. Each database allowed to perform non-durable writes for this stage only to ingest as fast as possible.



	1	2	4	8	16	32
Cassandra	18,683.43	31,144.24	53,067.62	86,929.94	173,001.20	326,427.07
HBase	15,617.98	23,373.93	38,991.82	74,405.64	143,405.64	296,857.36

Figure 7 Load process



	1	2	4	8	16	32
Caasandra	9,768.76	20,511.51	38,235.67	76,169.72	138,141.7	227,292.8
HBase	428.12	1,381.06	3,955.03	6,817.14	16,542.11	20,020.73

Figure 8 Read-mostly Workload

ISSN 2349-0292
Impact Factor 2.675

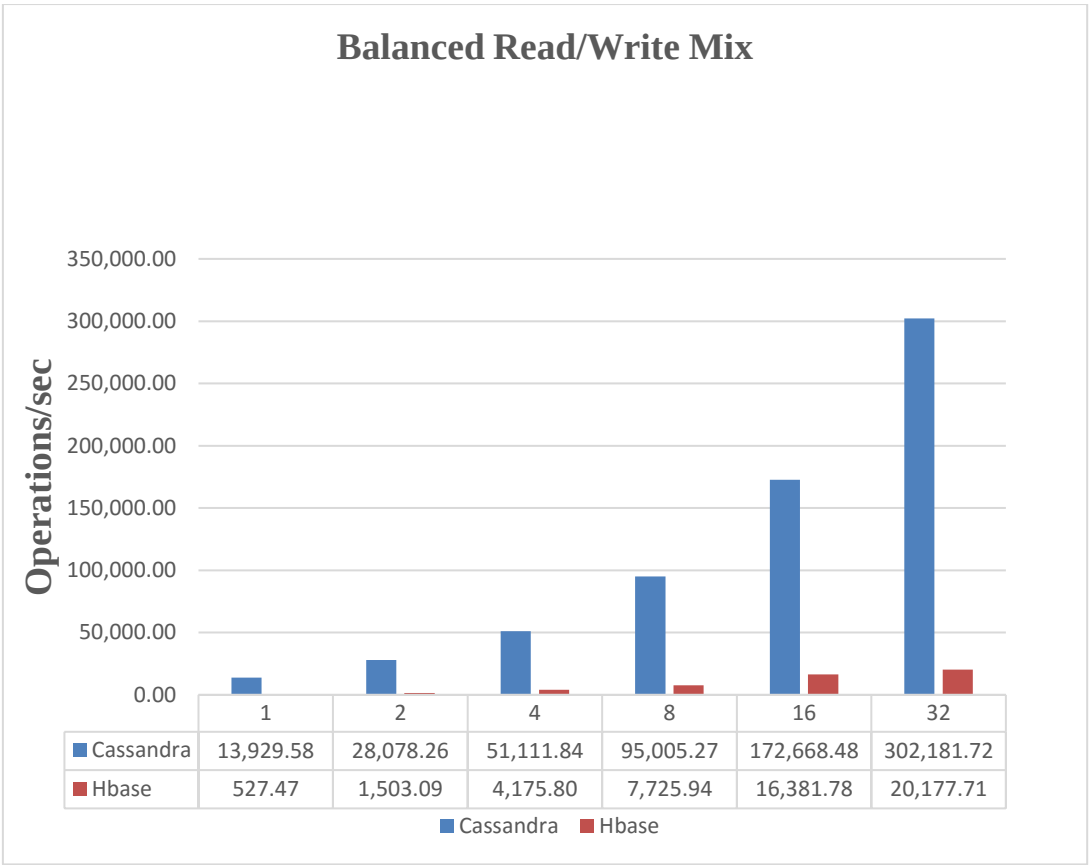


Figure 9 Balanced Read/Write Mix

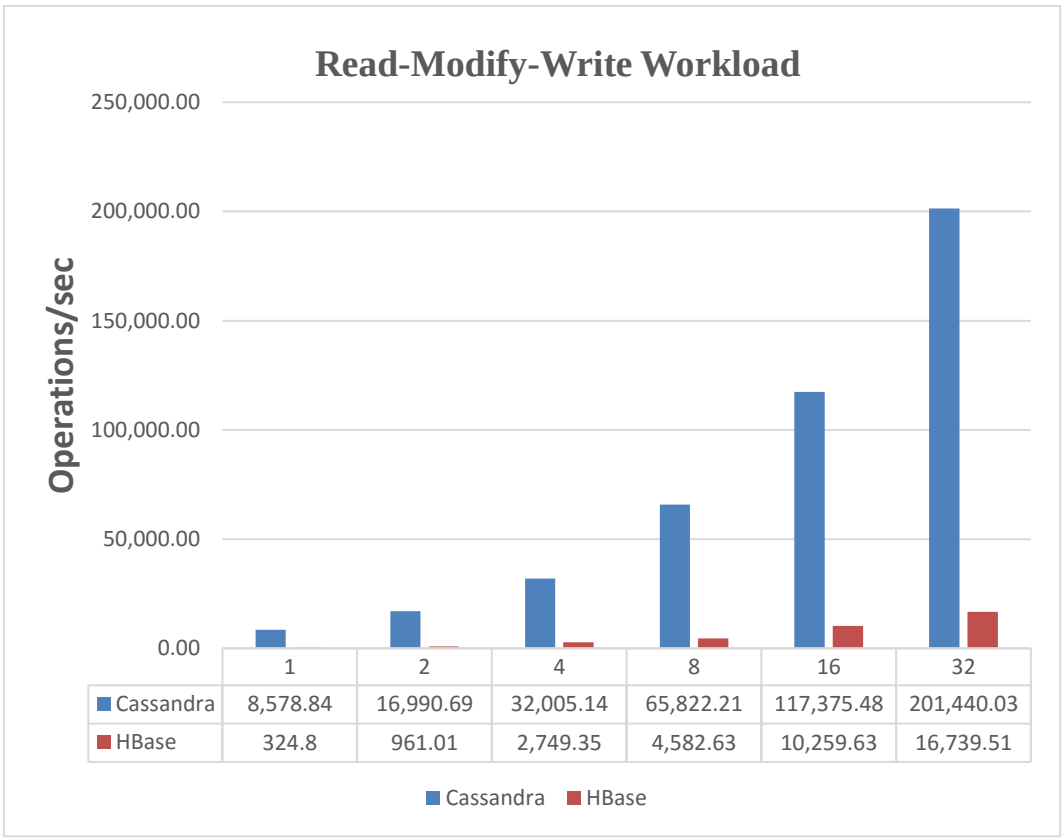


Figure 10 Read-Modify-Write Workload

ISSN 2349-0292
Impact Factor 2.675

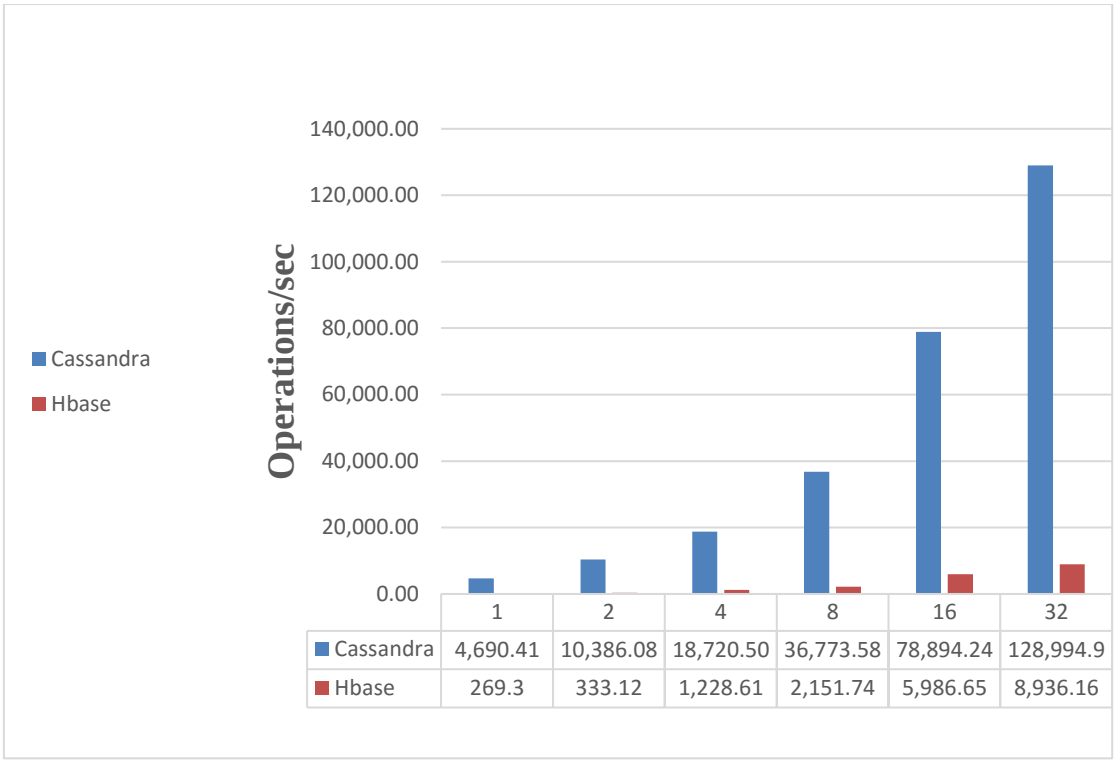


Figure 11 Mixed Operational and Analytical Workload

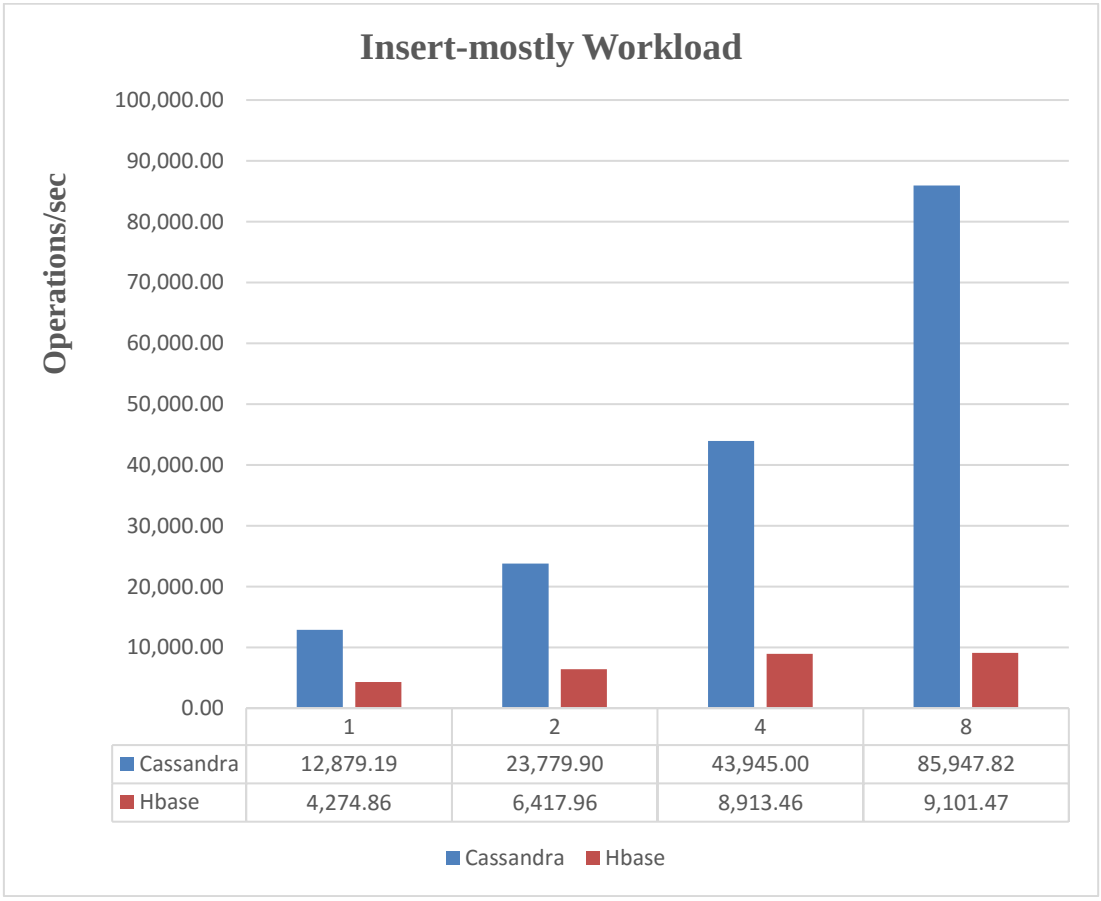


Figure 12 Insert-mostly Workload

The following latency results (less is better) are displayed below in microseconds. In general, Cassandra exhibited the lowest and ŕ consistent latency numbers for each test.

ISSN 2349-0292
Impact Factor 2.675

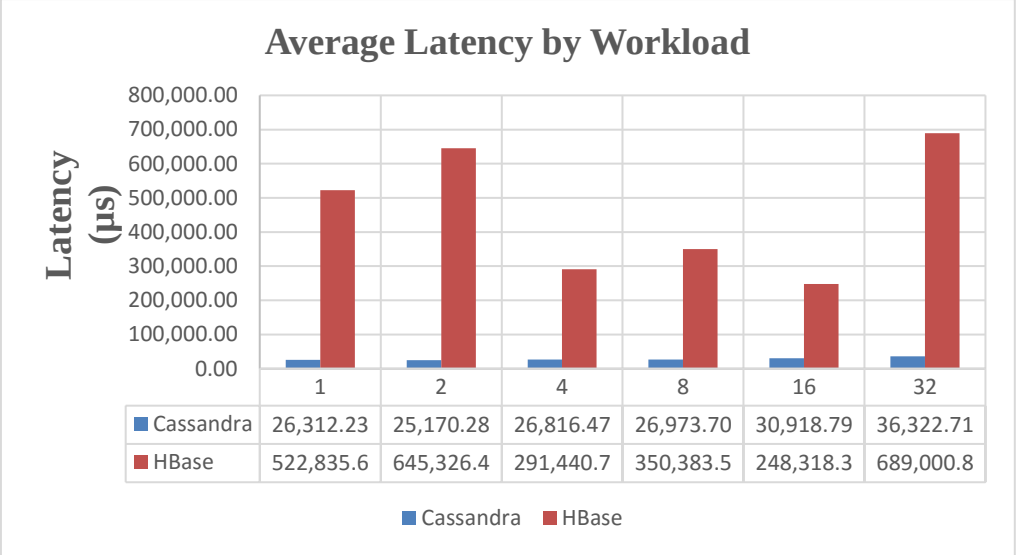


Figure 13 Average Latency ***by Workload***

7. Discussion

YSCB is used as a benchmark tool for testing the performance of the two databases. YCSB provides its results in a fairly structured text format for each workload. For each workload the overall throughput in operations/second and the total test runtime were gathered, along with the operation specifics in average latency.

The results were sum up as each client instance produced its own output. For the overall throughput, as each client tracked its own operations independently, the operations/sec across each client were summed in order to get an overall throughput per workload. Latency was aggregated as an average value of all client instances.

Result of the analysis shows that cassandra faster than Hbase in terms read, writes, insert and low latency.

Conclusion

NoSQL databases are the best choice for applications that store and process very large set of data. Among the classifications of NoSQL column-oriented databases has been selected as a case study. In which Cassandra and Hbase were chosen for this research, because they are the commonly used column store databases. BigTable was not included due to several features they have in common with Hbase.

The comparisons between Cassandra and Hbase shows their capability in handling Big Data storage challenges. They process very large amounts of data distributed over many machines.

Cassandra provides Availability and Partition-Tolerance while Hbase provides Consistency, and Partition-Tolerance.

Therefore, column oriented databases solve the biggest storage problems of Big Data.

REFERENCES

[1] Z. K. Lawal, R. Y. Zakari, M. Z. Shuaibu and A. Bala, "A review: Issues and Challenges in Big Data from Analytic and Storage perspectives," International Journal Of Engineering And Computer Science, vol. Volume – 5, no. Issue -03 , pp. No.15947-15961, March, 2016.
[2] S. Kaisler, F. Armour and J. A. Espinosa, "Big Data: Issues and Challenges Moving Forward," in 46th Hawaii International Conference on System Sciences, 2013.
[3] J. . K. U. and J. . M. David, "ISSUES, CHALLENGES, AND SOLUTIONS: BIG DATA MINING," Computer Science & Information Technology (CS & IT).
[4] G. J. and . E. Reinsel, ""Extracting Value from Chaos", IDC's Digital Universse Study, sponsored by EMC," 2011.
[5] A. A. TOLE, "Big Data Challenges," Database Systems Journal, vol. vol. IV, no. no. 3, pp. 31-40, 2013.
[6] K. S. Dr. Jangala., M. Sravanthi, K. Preethi and M. Anusha, "Recent Issues and Challenges on Big Data in Cloud computing," IJCST , , vol. Vol. 6, no. Issue 2, April - June 2015.
[7] "Oracle," Oracle , [Online]. Available: http://www.oracle.com/in/index.html.
[8] "MySQL," [Online]. Available: http://www.mysql.com/.

[9] "R. Rees, Comparison of NoSQL databases," [Online]. Available: http: //www.thoughtworks.com/articles/nosql-comparison .. [Accessed 13 March 2016].
[10] "CouchDB," [Online]. Available: http://couchdb.apache.org/.
[11] "SimpleDB," [Online]. Available: http://aws.amazon.com/simpledb/.
[12] "Redis," [Online]. Available: http://redis.io/.
[13] "The neo database. [Online]. Available: http://dist.neo4j.org/ Neotechnology-technology," Nov 2006. [Online]. [Accessed Jan 2016].
[14] R. Kaur, "A novel algorithm for transforming row-oriented databases into column-oriented databases," COMPUTER SCIENCE AND ENGINEERING DEPARTMENT THAPAR UNIVERSITY, PATIALA, PATIALA, June 2013.
[15] F. Changetal, ""Bigtable: a distributed storage system for structured data," in Proc. of the 7th symposium on Operating systems design and implementation OSDI '06,," Berkeley , CA, 2006, pp. 205-218..
[16] R. Rees, ""Comparison of NoSQL databases"," [Online]. Available: http: //www.thoughtworks.com/articles/nosql-comparison . [Accessed 2015 Nov].
[17] D. Abadi and S. Madden,, ""Debunking another Myth: Column-Stores vs. Vertical Partitioning",," The Database Column, 31 July 2008..
[18] F. Chang, J. Dean, S. Ghemawat and . W. C. Hsieh, ""Bigtable: A Distributed Storage System for Structured Data" in Seventh Symposium on Operating System Design and Implementation,," 2006.
[19] "Cassandra," [Online]. Available: http://cassandra.apache.org/.
[20] "HBase," [Online]. Available: http://HBase.apache.org/. [Accessed 05 March 2016].
[21] T. white, ""Hadoop: The Definitive Guide"," O'Reilly Media, 2009, pp. 524,.
[22] P. Rigaux, ""Understanding HBase and BigTable", [Online]. Available:," [Online]. Available: http://jimbojw.com/wiki/index.php?title=Understanding_HBase_and_BigTable. [Accessed Feb 2016].
[23] A. Gupta, ""HBase vs Cassandra"," [Online]. Available: http://bigdatanoob.blogspot.in/2012/11/HBase-vs-cassandra.html. [Accessed March 2016].
[24] R. Pointer, ""Introducing FlockDB"," 2010. [Online]. Available: https://blog.twitter.com/2010/introducing-flockdb. [Accessed Jan 2016].