International Journal of Technology (2015) 4: 640-649
ISSN 2086-9614

PERFORMANCE EVALUATION OF NOSQL-CASSANDRA OVER RELATIONAL DATA STORE-MYSQL FOR BIGDATA

Sangeeta Gupta[1*], Narsimha[2]

[1] *Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh 533003, India*
[2] *Jawaharlal Nehru Technological Universiy Hyderabad College of Engineering Jagtial, Kondagattu, Karimnagar, Telangana 505501, India*

(Received: August 2015 / Revised: September 2015 / Accepted: September 2015)

ABSTRACT

The massive amounts of data collected from numerous sources like social media, e-commerce websites are a challenging aspect for analysis using the available storage technologies. Relational databases are a traditional approach of data storage more suitable for structured data formats and are constrained by Atomicity, Consistency, Isolation, and Durability (ACID) properties. In the modern world, data in the form of word documents, pdf files, audio and video formats are unstructured. Therefore, tables and schema definition are not a major concern, Relational databases, such as Mysql, may not be suitable to serve such Bigdata. An alternate approach is to use the emerging Nosql databases.

In this work, a comprehensive performance and scalability evaluation of large web collection data in data stores, such as Nosql-Cassandra and relational-Mysql, is presented. These systems are evaluated with data and workloads that can be found related to Bigdata, yielding scalability of applications. The insights presented in this work serve not only for performance and scalability, but also as lessons learned and experiences relating to the configuration complexity and evaluation in sorting out the complex queries of what data storage can be used on which usage cases for large data sets. The results show how the Bigdata collected across the Web with billions of records generating continuously are poorly evaluated with Mysql in terms of 'write' operations, but how these perform well with Nosql-Cassandra. This paper yields a new approach which is unique in representing Nosql-Cassandra's poor performance in retrieval of records and disk utilisation with ever-increasing loads. The results presented in this paper show an improvement in 'read' performance with the proposed architecture and configuration over Mysql, achieving cost saving benefits to any organisation willing to use Nosql-Cassandra for managing Bigdata for heavy loads.

Keywords: Bigdata; Cassandra; Crawler; Mysql; Nosql

1. INTRODUCTION

Cloud computing has evolved as a new computing paradigm, allowing end users to utilise the resources on a demand-driven basis, unlike grid and cluster computing which are the traditional approaches to access resources. Enormous amounts of data flooded across the Internet and the storage capacities of the relational technologies have experienced inadequacy to access the huge amounts of data. To store petabytes (One quadrillion bytes) of data, most of the organisations, particularly social networking sites and e-commerce sites are moving towards the Cloud to

* Corresponding author's email: ss4gupta13@gmail.com, Tel. +99-89-404702, Fax. +99-89-404702
Permalink/DOI: http://dx.doi.org/10.14716/ijtech.v6i4.1557

deploy their applications, but at increased security risks. These growing amounts of data which are too big and complex to capture, store, process, and interpret are referred to as Bigdata as specified in (Venkat et al., 2014). It is characterised by the 4V's, such as Volume, Velocity, Veracity and Variety. The storage and analysis of such data can be made effective using the Nosql databases.

The foremost benefit of the Cloud is to pay only for the resources which users utilise. If there is an unexpected set of users competing for access to resources, they would just have to pay only for what they have been using with every user's request being satisfied. This is known as elasticity of the Cloud. The Cloud provides a variety of service models, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), Database as a Service (DaaS) and other deployment models, such as public, private, hybrid and community clouds. To be hosted on a scalable environment, an application can use either of these models in a cost efficient manner to realise benefits. Other benefits provided by the Cloud can be utilised in terms of elasticity, scalability, efficiency and reusability (Thomas & Dongman, 2014).

Most modern world data is processed in the form of word documents, pdf files, audio and video formats. Relational databases may not be suitable to serve such data. Also, using Relational databases for scalable applications impose heavy costs and make them less attractive for deploying large scale applications in a cloud, (Divyakant et al., 2011). An alternative approach is to use the emerging Nosql databases, which are not ACID-compliant (Atomicity, Consistency, Isolation, and Durability). Atomicity requires actions (read/write) to be either fully complete or not done at all. Consistency ensures only valid data is to be stored in database. Isolation ensures that concurrent execution of actions results in a system state that would be obtained if actions are executed serially. Durability ensures that the committed actions will remain so in the event of system failures. In contrast to Relational databases, Nosql provides support to structured, unstructured, and semi-structured storage of massive data in terms of peta bytes.

The methodology of this paper is organised as follows: The first section is the introduction. The second section presents related work or a background study, including the scenarios where the aforementioned databases are being used. This section also defines what are the drawbacks observed in terms of the limitations of existing systems. The third section presents the proposed method with a new methodology to perform the comparison between Mysql and Cassandra in a way, which was not discussed in any of the previous works. The fourth section presents the results as obtained when inserting and retrieving records in multiples of hundred in both Mysql and Cassandra and shows the pitfalls in both Mysql and Cassandra for various operations with solutions designed to overcome those pitfalls in Cassandra. The fifth section concludes the work and throws light on the future enhancement.

2. RELATED WORK

Several related works on Mysql and Nosql types are discussed and their limitations are observed. Nosql databases like HBase, MongoDB and applications which use these databases are discussed, which enables an understanding of the advantages of Cassandra over these databases.

Mysql has been used as a prominent relational database for storing data samples in a wide variety of applications. According to Naim et al. (2011), Mysql has been used to store fingerprints data for biometrics, with the help of a virtual server. Tables were created for personal identification numbers, real end-point data and real branch-point data, which employed structured data storage. If the amount of information collected was drastically increased, then this would require a large number of tables to accommodate the growing number of data sources. Also if the data storage is in the form of text or image format rather than pixel data, then the use of Mysql will become inappropriate (Sudhanshu & Shelly, 2014). This research compared the performance of Mysql with the DB4o database on a sample hospital dataset. Also, this research indicated that object-oriented databases, such as DB4o are always better when compared to relational databases, such as Mysql in terms of time taken to access the data efficiently in the event of the growing amounts of huge data records. Though object-oriented databases deal well with respect to huge data records, they also occupy large storage space.

The column-oriented data store HBase is a distributed database developed on top of the Hadoop Distributed File System (HDFS), which adopts master-slave architecture with Name Node acting as Master and Data Nodes acting as slaves (Vora, 2011). The Nosql database HBase was used to perform random reads and writes on very large datasets in the form of image files and the results proved to be better than using Mysql on such data. Though the performance of HBase was shown to be better than Mysql, some literature stated that the model is appropriate to perform 'write-once, read-many' operations on the attributes, but it is not suitable to support multiple 'write' operations, i.e. the files in HDFS are accessible efficiently in the 'read' mode, but it does not support multiple 'writes'.

(Gansen et al., 2013) presented a comparison of Mongo DB, a document-based Nosql store with Mysql, highlighting the exceptional features of Mongo DB, like support for dynamic schemas, faster data integration, as well as support for adhoc queries, load balancing and automatic sharding. The comparison also depicted the support of Mongo DB in regard to relational calculus, again achieving a better performance rating than Mysql. However, no specialist tools were available to analyze the data efficiently.

In this body of the literature, the following shortcomings are identified: The first database mentioned is Mysql, which is found to be unsuitable for unstructured/semi-structured data storage, and also it possesses an inability to share workloads across multiple servers. Next, in the applications using HBase, it was found that Hbase is suitable to perform random 'reads' and 'writes', but it is not suggested for applications that are required to perform multiple 'write' operations. The other database discussed is Mongodb, which though proven to be effective to implement relational calculus, it was inefficient for aggregate queries.

The drawbacks of all these databases could be overcome by using Nosql-Cassandra, which has peer-to-peer distributed architecture that is easy to set up and maintain. In Nosql-Cassandra, all nodes share an equal priority with no concept of a master node. There is no single point of failure and it is capable of offerring continuous availability and efficient replication of data across different data centres and cloud platforms. In spite of these advantages, it has flaws with respect to 'read' and 'write' operations on Bigdata. Nosql-Cassandra is used as the crawler application to show its merits and flaws. Furthermore, solutions are proposed to overcome the flaws.

3. PROPOSED SYSTEM FOR UNSTRUCTURED DATA ANALYSIS WITH WEB CRAWLER

The proposed system is designed to generate huge unstructured data from known sources like web applications (Social Media sites like Twitter or e-commerce sites like flipkart.com) using the web crawling technique. The data generated is parsed and then put through a connector and then it is inserted into the Mysql and Cassandra databases. Time stamps are placed at regular intervals to monitor the performance of both the databases through a log.

The architecture is designed in such a way that multiple web sites can be crawled through multi-threading and the obtained data is parsed on a row-by-row basis. Two connectors are

created to connect to the Mysql and Cassandra data servers. Every record generated by the parser, is inserted into the Mysql and Cassandra data service using connection drivers.
The application framework for the work is as shown in Figure 1.

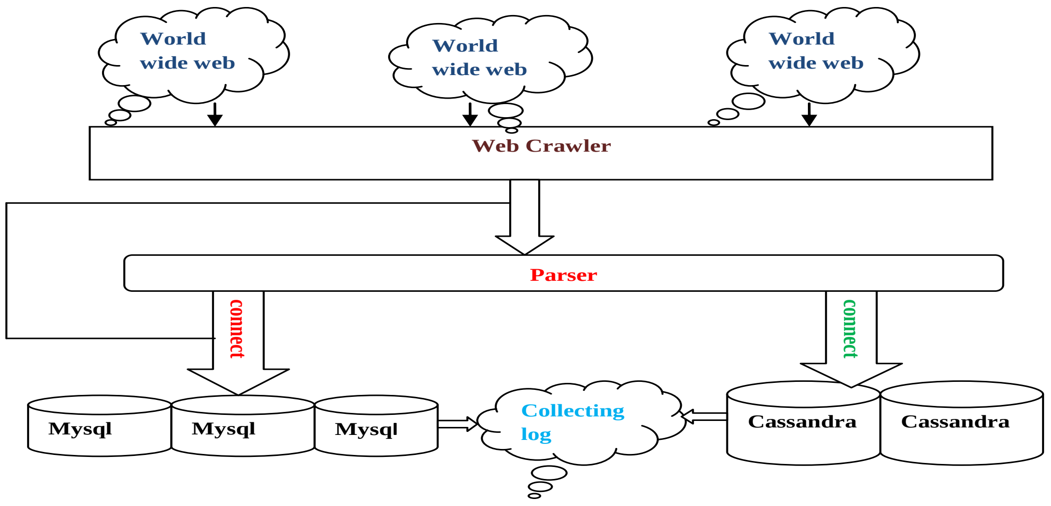

Figure 1 The application framework for proposed system

The Proposed Architectural view is as shown in Figure 2.

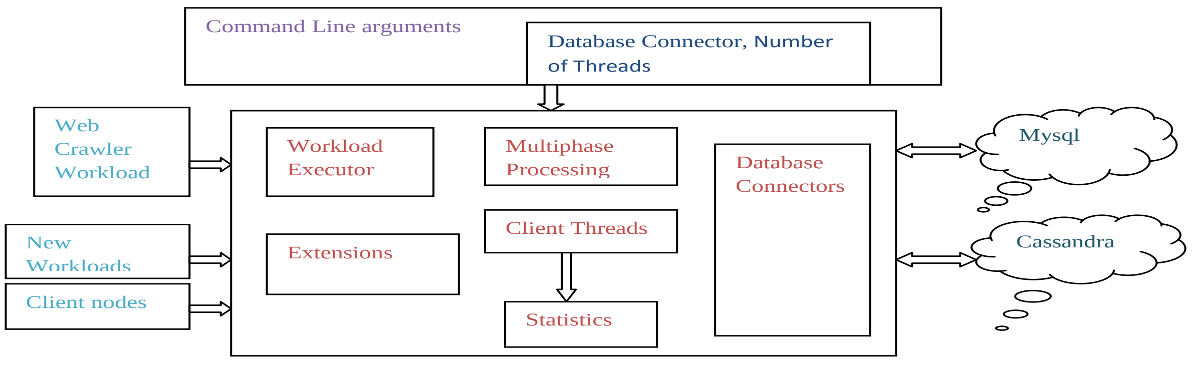

Figure 2 Architectural view for proposed system

The building block for the Architecture solution is in the form of a Bigdata workload generator through a web crawler application. The other components of the architecture include: Data collection layer, Database interface layer and statistics/performance collection matrices.

3.1. Workload Generator

The workload is the key to performance benchmarking and stability analysis. One application is needed to generate continuous data for batch processing or high streaming real and live data. Web crawler is the chosen application, which generates data from various e-commerce sites, which is highly unstructured. The application will also generate the 'read' and 'write' requests to Nosql-Cassadra and Mysql databases, suitable for benchmarking.

3.2. Workload Executor

The Workload executor is run in two phases:

1. Load Phase ('Write' Phase)
2. Retrieve Phase ('Read' Phase)

The load phase workload working set is created from 100 records to 1 million records. These records are loaded to Cassandra and Mysql through JDBC connectivity. The client threads create multiple threads to load data in parallel in both Cassandra and Mysql databases. Increasing the number of threads can increase the throughput of the database.

The Retrieval phase works on data loaded in databases during the load phase. This phase generate some queries which read data from clusters. These queries can retrieve the small data set as well as large data sets with simple 'SELECT' to complex joint queries.

3.3. Statistics/Metrics Collection

The statistics are collected through logs by writing the application to database and dashboard. Timestamps are placed at regular intervals to monitor the performance and the results are recorded for load and retrieval phases with a varying number of records.

4. PERFORMANCE BENCHMARKING

4.1. Benchmarking with Existing System

Benchmarking is referred to as the process of evaluating a system against some reference to determine the relative performance of the system. The basic primitive job of the database system remains generic, yet database systems exhibit different flavours and requirements based on the environment in which they operate. Also, database systems contribute hugely to the proper and efficient functioning of organizational and business information needs. Hence, selecting the right database with the right features is often a very critical decision. To aid such decisions, a bunch of benchmarking techniques have been designed, both commercially and in the open source platform as cited in a white paper by Datastax corporation (2014).

4.1.1. Load process

As part of Benchmarking, Bulk load was done ahead of each workload. Each database was allowed to perform non-durable 'writes' for this stage only to inject data as fast as possible.

For low loads (100 records), 'write' requests showed a steady performance but with increasing load (increasing the number of records from 100 to 1,000, 10,000 and so on) with hundreds of requests on the same node, the performance scaled up and reached a maximum level at a certain peak point. Even though the throughput is distributed for low loads, it was observed to be stable for high loads.

The performance for 'writes' is similar in Mysql as in Cassandra, but more time is taken more Mysql showing that Cassandra performs much better 'writes' over Mysql. The results were recorded as shown in Table 1 and graphically plotted as in Figure 3.

Table 1 Write Performance

Records (no. of inserts)	Cassandra WT (ms)	Mysql WT (ms)
100	1	5
200	2	9
500	4	19
1,000	8	43
10,000	60	400
100,000	456	3,000

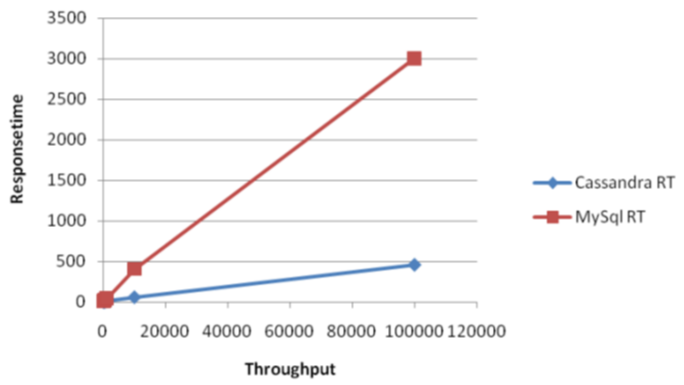

Figure 3 ‘Write’ performance for throughput versus response time

4.1.2. Retrieval process

During the retrieval phase, the time taken to retrieve the records increased drastically in Cassandra, while it was a gradual increase in Mysql with an increasing number of records (100; 1,000; 10,000; 100,000) on the same hardware configuration. Hence Mysql shows better results during the retrieval phase over Cassandra. The results for retrieval of records from both Cassandra and Mysql databases are as shown in Table 2 and Figure 4.

Table 2 ‘Read’ performance

Records (no. of retrievals)	Cassandra RT(ms)	Mysql RT(ms)
100	2	1
200	3	2
500	5	2
1,000	8	5
10,000	10	6
100,000	12	8

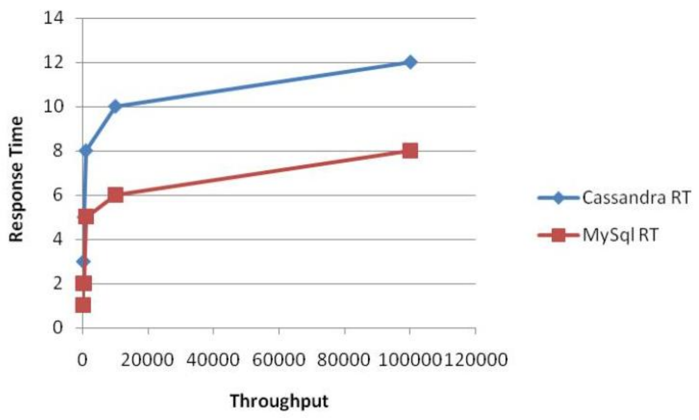

Figure 4 ‘Read’ performance for throughput versus response time

4.1.3. Resources utilisation in Cassandra

Figure 5 represents a dashboard to monitor and manage a Cassandra cluster. This is a visual management and monitoring solution for Cassandra. The DataStax OpsCenter can be installed on any server – on the premise or in the Cloud – that has connectivity to clusters running Cassandra or DataStax.

Each node in a Cassandra or DataStax cluster contains a DataStax agent, which communicates with the central OpsCenter service. The DataStax agent and OpsCenter service work together to monitor and handle tasks on every managed cluster. This content is taken from a white paper on the modern online application for Internet economy by Datastax Corporation (2014).

The OpsCenter provides a Web-based console from which everything can be centrally managed. The OpsCenter interface provides a visual point-and-click environment for quickly carrying out many administration and performance monitoring activities.

The disk utilization and increasing loads on the cluster drastically exceed the maximum available capacity as shown in Figure 5. This figure has been taken by installing a Datastax community version on the desktop and hence it cannot be modified. Figure 5 represents a dashboard to monitor and manage a Cassandra cluster. This dashboard is a visual management and monitoring tool for Cassandra that enables representation of various statistics for ‘read’/’write’ disk operations, suitable for an efficient analysis. A single node cluster is opted for in initial phase of the work, with storage capacity of 100 GB. The ‘write’ requests and disk utilization are recorded for different time intervals. Disk utilization exceeds the maximum available storage capacity with increasing loads as shown in Figure 5.

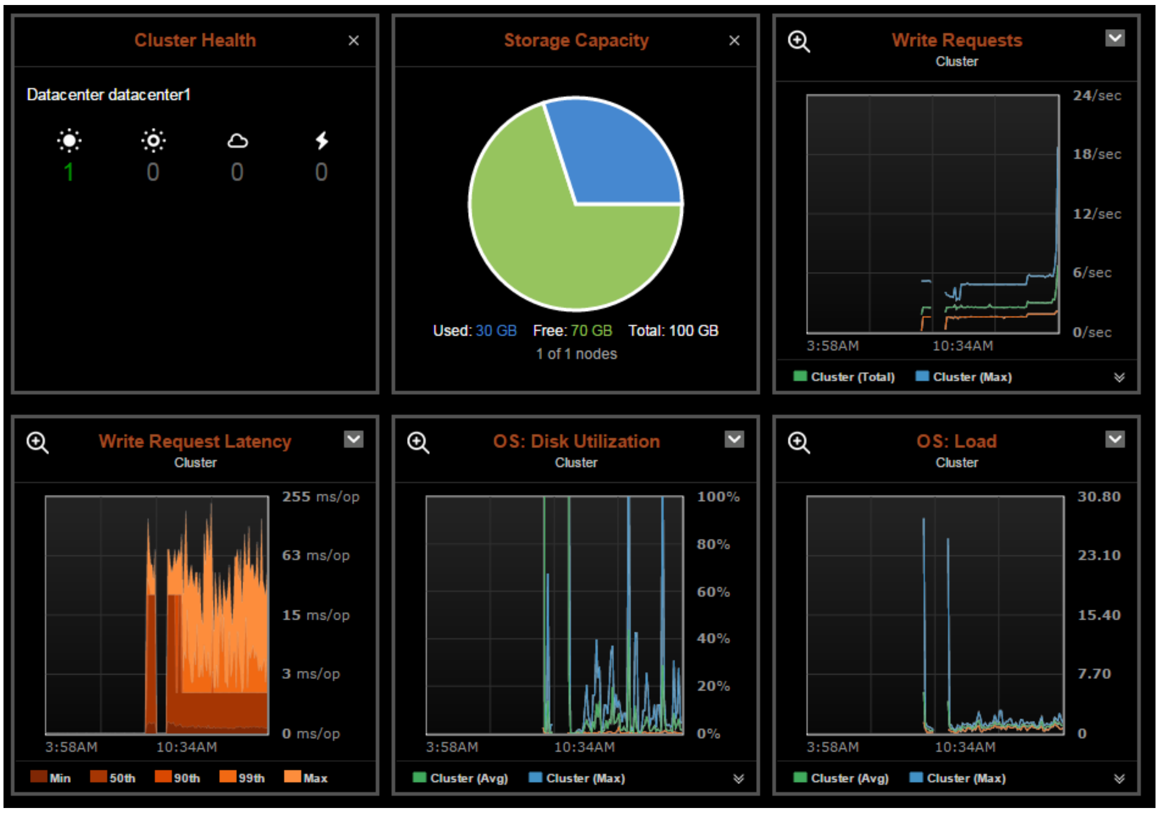

Figure 5 OpsCenter interface showing the ‘write’ and disk performance

4.1.4. Overall observations

Cassandra has performed far better compared to MySQL on I/O bound (‘write’) operations, but shows poor performance on CPU bound (‘read’) operations. Also the disk utilization is exceeding the maximum allotted space in Cassandra, i.e. it is very high for low loads, which may crash the entire system.

To overcome the above problems with the retrieval phase and to optimize the disk performance, changes have been made at the hardware configuration level to yield better results with the proposed model.

4.2. Benchmarking with Proposed System

With the above existing configuration, Cassandra, though better than Mysql in some respects ('writes'), it is still performing poorly on 'read'-bound operations. As a part of the proposed work, the following configuration is designed to achieve better results.

4.2.1. Proposed Cassandra configuration

Random partitioner Initial token space: 12^{10} / 4
Memory table space: 4 GB
Commit log is on the separate disk Concurrent reads: 8 (4 cores)
Concurrent writes: 16 (8×2 disks)
Compression: Snappy Thrift Compression
max_heap_size: 1 GB
heap_newsize: 524 MB
Rest of 2GB RAM is for OS caching

4.2.2. Proposed Cassandra Keyspace/Column Family Setup details

```
CREATE KEYSPACE TestKeyspace
WITH placement strategy = SimpleStrategy
AND strategy options = {replication_factor:1}
AND durable_writes = true;
CREATE COLUMN FAMILY TestColumnFamily WITH comparator = UTF8 Type
AND key_validation_class = UTF8Type
AND keys_cached = 100000
AND rows_cached = 1000 AND
row_cache_provider = 'SerializingCacheProvider'
AND compression options = {sstable_compression:SnappyCompressor}
```

Table 3 Improved Read Performance with proposed configuration

Records (no. of retrievals)	Cassandra RT(ms)	Mysql RT(ms)
100	1	1
200	2	2
500	3	2
1,000	4	5
10,000	5	6
100,000	6	8

The Read performances for mysql and Cassandra are recorded in Table 3 for the proposed configuration for increasing number of records (100; 1,000; 10,000; 100,000). Cassandra shows an improvement (decrease) in its retrieval operations over Mysql with the proposed configuration.

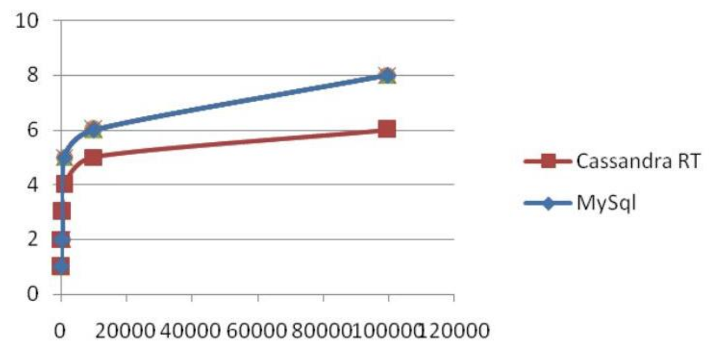

Figure 6 Graph to show the improved performance with the proposed model. (*x*-axis represents Throughput and *y*-axis represents Response Time)

Figure 6 represents graphically the improvement in 'read' performance for Cassandra over Mysql with the proposed configuration.

5. CONCLUSION

In this paper, a comprehensive performance and scalability evaluation of large web collection data in data stores, such as Nosql-Cassandra and Mysql is presented. These systems were evaluated with data and workloads that can be found related to Bigdata, yielding scalability of applications. Initially, the web crawler application was deployed in a single node Mysql database and Cassandra. Then the number of nodes were increased to monitor the scalability. It was found that Cassandra, though it performs better than Mysql with respect to 'write' operations; it performs poorly towards 'read' operations. To overcome this problem, a solution is proposed by making changes at the configuration level and hence better results were obtained.

With the proposed configuration, though able to maintain a balance of 'read' and 'write' operations, the disk utilization exceeded peak levels with Cassandra. So, our next phase of work is to further finetune the parameters in such a way that the loading, retrieval and disk utilization operations will all achieve better optimized results than with Mysql. Also, data security is an unsolved issue for applications deployed in the public cloud. As a further extension to this work, achieving security, while deploying the proposed model in the public cloud at application and at database levels will add value to the real time Bigdata applications.

6. REFERENCES

Datastax Corporation, 2014. *The Modern Online application for the Internet economy: 5 Key Requirements that Ensure Success*. White paper by Datastax Corporation, Santa Clara, Calif., Available at datastax.com

Divyakant, A., Das, S., Abbadi, A.E., 2011. Bigdata and Cloud Computing: Current State and Future opportunities. In: *Proceedings of the EDBT* 2011/ACM, March 22–24 2011, Uppsala, Sweden

Gansen, Z., Huang, W., Liang, S., Tang, Y., 2013. Modelling Mongo DB with Relational Model. In: *Proceedings of the Fourth International Conference on Emerging Intelligent Data and Web Technologies, IEEE,* Volume 25, pp. 115–121

Introduction to Apache Cassandra, 2013. White paper by Datastax Corporation, San Mateo, Calif., July 2013

Naim, N.F., Mohd Yassin, A.I., Wan Zamri, W.M.A., Sarnin, S.S., 2011. Mysql Database for Storage of Fingerprint Data. In: *Proceedings of the 13th International Conference on Modelling and Simulation, IEEE*, Volume 62, pp. 293–298

Sudhanshu, K., Shelly, S., 2014. Performance Comparison for Data Storage-DB4o and Mysql Databases. In: *Proceedings Seventh International Conference on Contemporary Computing, IEEE, 2014*

Thomas, S., Dongman, L., 2014. Notes on Cloud Computing Principles. *Journal of Cloud Computing: Advances, Systems and Applications*, Volume 3(21), pp. 1–10

Venkat, N.G., Dhana, R., Vijay, V.R., 2014. Nosql Systems for Bigdata Management. In: *Proceedings of the 10th World Congress on Services*, *IEEE*, Volume 42, pp.190–197

Vora, M.N., 2011. Hadoop-HBase for Large Scale Data. In: *Proceedings of the IEEE International Conference on Computer Science and Network Technology*, December 24–26, 2011, pp. 601–605