Efficient Query Analysis and Performance Evaluation of the Nosql Data Store for BigData

Sangeeta Gupta and G. Narsimha

Abstract The voluminous amounts of data generated from the web applications and social networking and online auction sites are highly unstructured in nature. To store and analyze such data, traditional ways of using relational databases are not suitable. This yields path towards the acceptance of emerging nosql databases as an efficient means to deal with bigdata. This work presents an efficient nosql data store and proves its effectiveness by analyzing the results in terms of efficient querying by evaluating the performance estimation on read and write operations on simple and complex queries, also for storage and retrieval of increasing number of records. The results presented depict that the chosen nosql datastore—Cassansdra is efficient over the relational database—mysql and the other nosql databases—HBase and MongoDB, that leads to achieving cost saving benefits to any organization willing to use Nosql-Cassandra for managing Bigdata for heavy loads.

Keywords Mysql · Nosql · Bigdata · Hbase · Mongodb · Cassandra

1 Introduction

Enormous amounts of data flood across the internet and the storage capacities of the relational technologies have experienced inadequacy for the same. To store peta bytes of data, most of the organizations, particularly social networking sites and e-commerce sites are moving towards cloud to deploy their applications, but at increased security risks. This growing amounts of data which is too big and

S. Gupta (✉)
JNTUK, Kakinada, India
e-mail: ss4gupta13@gmail.com

G. Narsimha
CSE, JNTUHCEJ, Kondagattu, Karimnagar, India

complex to capture, store, process, and interpret is referred to as Bigdata. It is characterized by 4 Vs such as Volume, Velocity, Veracity and Variety [1]. The storage and analysis of such data can be made effective using the Nosql databases.

Cloud computing has evolved as a new computing paradigm, allowing end users to utilize the resources on a demand-driven basis, unlike grid and cluster computing which are the traditional approaches to access the resources. The foremost benefit of cloud is to pay only for the resources which users utilize. If there are an unexpected set of users bombarding for the resources, they would just have to pay for what they have been using. This usage is termed as elasticity of the cloud. 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 deployment models such as public, private, hybrid and community clouds. An application to be hosted on a scalable environment can use either of these models in a cost-efficient manner to reap their benefits. The other benefits provided by cloud can be utilized in terms of elasticity, scalability, efficiency, reusability [2].

Most of the modern world data are projected in the form of word documents, pdf files, audio and video formats and relational databases may not be suitable to serve such data. Also, using them for scalable applications impose heavy costs making them less attractive for deploying large-scale applications in cloud. An alternate approach is to use the emerging Nosql databases, which are not ACID compliant and which provide support to structured, unstructured, and semi-structured storage of massive data in terms of peta bytes. Nosql databases do not rely on a fixed schema, there are no join operations and they rely on CAP (Consistency, Availability, and Partitioning) features in contrast to the ACID properties supported by traditional databases. Eventual consistency is supported by a few Nosql types, where the updates may not propagate immediately across all the nodes in a cluster [3].

This paper presents a comparative analysis of various Nosql types such as Hbase, Mongodb, with Mysql and brings into light their limitations. It also presents a novel integrated Nosql, Cassandra which is advantageous and efficient over the mentioned Nosql species, by amalgamating their benefits and crossing out their shortcomings in terms of performance and scalability over the estimated read and write operations on the database for execution of simple and complex queries.

The paper is organized as follows: Sect. 2 presents background study of the work including the scenarios where the mentioned databases are being used; Sect. 3 presents a detailed comparative analysis with suggested solutions and integrated store development with performance estimation on read and write operations on simple and complex queries, 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 better performance of Cassandra over mysql. The fifth section concludes the work and throws light on the future enhancement.

2 Related Work

In this part of the section, several related works on mysql and nosql types are discussed and their limitations are observed.

Mysql has been used as a prominent relational database for storing data samples in a wide variety of applications. Naim et al. [4] have used mysql to store finger prints data for biometric, with the help of a virtual server. Tables created were person identification number, real end-points data and real branch-points data, which employ structured data storage. If the amount of information collected is drastically increased, this would require large number of tables to accommodate the growing data and also if the data storage is in form of text or image format rather than pixel data as in [4], then usage of mysql will become inappropriate.

Kulshreshta and Sachdeva [5] have compared the performance of mysql with DB4o database on sample hospital dataset and showed that object-oriented databases such as DB4o are always better as compared to relational databases such as mysql in terms of time taken to persist the data in the events of huge amounts of growing records. Though object-database deals well with respect to the huge data, they occupy large storage space.

The column-oriented data store HBase is a distributed database developed on top of Hadoop Distributed File System (HDFS), which adopts master–slave architecture with Name Node acting as a Master and Data Nodes acting as slaves. Vora [6] used the Nosql database HBase to perform random reads and writes on very large datasets in the form of image files and the results were proved to be better than using mysql on such data. Though the performance of HBase was shown to be better than mysql, the limitation of [5] presented the model to be appropriate to perform write-once read-many operations on the attributes, but not suitable to support multiple write operations, i.e., the files in HDFS were accessible efficiently in read mode but does not support multiple writes. Also, another limitation observed was, in order to use HBase, an in-depth understanding about Hadoop framework, MapReduce Programming model is required.

Zhao et al. [7] 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, support for ad hoc queries, load balancing and automatic sharding and also depicted the support of mongo DB at relational calculus, achieving better performance than mysql. But the limitation of this approach is that there is no expertise in this area and no specialist tools are available to analyze data efficiently.

In this section, apart from identifying various areas of application of both Relational and Nosql databases, their limitations are brought into consideration and solutions are presented by using an integrated data store-Cassandra in the next section to overcome the mentioned limitations.

3 Cassandra—An Integrated Data Store

Cassandra is a novel integrated Nosql, used which aims at providing support to any kind of data (structured, semi-structured or unstructured) as emerging from the real-world social networking websites such as facebook, and e-commerce sites such as eBay concatenating the scalability aspects of BigData, leading to eventual consistency and providing an efficient way to solve complex queries by avoiding join operations. Cassandra is used to overcome the limitations possessed by the mentioned data stores and it integrates the benefits of Mysql, Hbase and MongoDB data stores. Hence, any modern-world application would be greatly benefitted in migrating their applications from Mysql to Cassandra. It uses peer-to-peer architecture, where all nodes are given equal priority in a cluster and the nodes are said to communicate with each other through gossip protocol. There is no single point of failure in Cassandra; hence there is no down time for running an application. The query language used to perform operation with the database is CQL (Cassandra Query Language).

3.1 Comparison of Nosql with Mysql

Relational Databases are confined to ACID properties in contrast to Nosql's CAP properties. Nosql databases support both strict and eventual consistency, in which changes need not propagate instantly across all the nodes in a cluster as compared to the relational databases, which provide support only to strict consistency. Nosql serves horizontal scalability aspect than the vertical scalability of relational model as in mysql [8].

Table 1 depicts differences between relational and Nosql databases in terms of features like consistency, scalability, join operations and data formats.

Table 1 Comparison of relational with Nosql databases

	Supporting features			
Database	Consistency	Scalability	Data format	Joins
Relational (mysql)	Strict	Vertical	Structured	Complex tasks require joins
Nosql (Hbase, Mongodb, cassandra)	Strict/eventual	Horizontal	Structured, semi-structured, unstructured	No joins are to be performed

3.2 Querying Differences

Relational databases like mysql, oracle, etc., use SQL for storing, retrieving and manipulating data, whereas in nosql types, there is no single standard query language to meet varying users requirements. Querying data stored in nosql databases is specific to the data model. So, each nosql comes with its own query language like, Cassandra has CQL, HBase has HQL, etc.

To explain about the differences among mysql, hbase, Cassandra, we have considered sample tables titled journal and conference. The syntaxes used by various data stores vary as shown in Table 2 to perform insert, update and delete operations. We have also taken simple and complex queries to analyze these differences for data retrieval operation (select) as in Tables 3 and 4.

In the above example, the syntax for retrieving (reading) data from mysql and Cassandra are similar, while in mongoDB find() is used to retrieve the data, and in hbase, get is used for the same.

Table 2 Querying differences between mysql and nosql with insert, update and delete operations

Database	Insert operation	Update (write) operation	Delete operation
Mysql	Insert into journal values (‘ieee’,1234,‘openaccess’);	Update journal set jid = 1234 where jid = 1345;	Delete from journal where jname = ‘mnuoo’;
HBase	Put ‘journal’,‘row1’,‘jid: a’,‘ieee’;	Same as insert	Disable ‘journal’;
MongoDB	Db.journal.insert ({jname:“ieee”,jid:1234, accesstype:“openaccess”})	Db.journal.update ({}, {‘$set’:‘jid’:‘jid’}});	Db.journal.remove ();
Cassandra	Insert into journal values (‘ieee’,1234,‘openaccess’);	Update journal set jid = 1234 where jid = 1345;	Update journal set jid = 1234 where jid = 1345

Table 3 Simple query: query to find the name of journal with id 1234

Database	Retrieval operation
Mysql	Select j.jname from journal j where j. jid = 1234;
HBase	Get ‘journal’,‘jname’;
MongoDB	Db.journal.find({}, {“jname”:1,“jid”:0,“jtype”:0});
Cassandra	Select jname from journal where jid = 1234;

Table 4 Complex query (joins/nested): query to find the journal names whose ids match with that of the ids in conference table

Database	Retrieval operation
Mysql	Select j.jname from journal j where jid in(select c.jid from conference c). Here jid in conference table is a foreign key
HBase	Get command can't be used to run the same query as in mysql, but if integrated with mapreduce code, the query will be executed
MongoDB	As in HBase, MongoDB also requires mapreduce command integration
Cassandra	Super column families and data denormalization can be done to execute complex queries in an efficient way

Cassandra Super column families and data denormalization can be done to execute complex queries in an efficient way.

To perform complex joins or nested queries, Mysql requires foreign keys to be created performing joins across multiple tables. But, this method may lead to increase in execution time in order to retrieve data from multiple tables, thereby degrading the overall performance. In HBase, complex joins are supported by integrating hbase code with mapreduce code using nested loops, which is again a time-consuming process. Mongo DB also uses mapreduce command to process such data.

In Cassandra, performing complex joins or nested queries requires denormalization of data into partitions, leading to efficient querying from a single replica node, rather than gathering the data from across the entire cluster. Thus, it provides an efficient mechanism to retrieve data in a simpler way, and also the speed of query execution is much better than in mysql, mongodb and hbase.

4 Result Evaluation for Unstructured Data Using Nosql and Mysql Data Store

This section presents the results of evaluating increasing number of records for both read and write operations using mysql and Cassandra. It also shows the better performance of Cassandra over mysql for write operations. However, the other nosql types have not been used due to time constraints.

4.1 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-Cassandra and Mysql databases, suitable for benchmarking.

4.2 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 generates 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.

4.3 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.

4.4 Performance Benchmarking

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 flavors 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.

4.4.1 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.

Table 5 Write performance

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

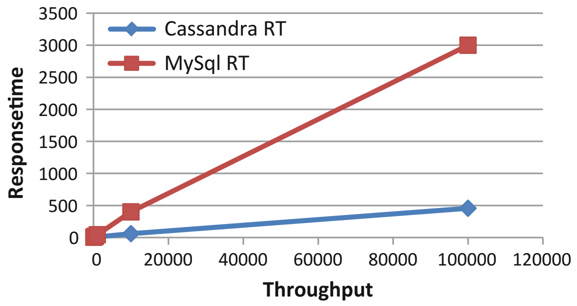

Fig. 1 Write performance for throughput versus response time

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 by Mysql showing that Cassandra performs much better ‘writes’ over Mysql.

The results were recorded as shown in Table 5 and graphically plotted as in Fig. 1.

4.4.2 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 6 and Fig. 2.

Table 6 Read performance

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

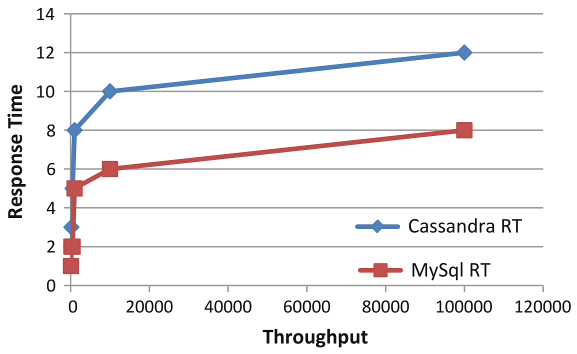

Fig. 2 Read performance for throughput versus response time

5 Conclusion and Future Work

Most of the organizations rely on structures databases like Mysql, which do not harness the requirements of scalability and availability of real-world data. The available set of Nosql databases support various aspects to meet the upcoming trends in growing data like support for eventual consistency, scalability, availability, and fault-tolerance. In this paper, Nosql databases Hbase and Mongodb are discussed and apart from mentioning their advantages as compared to Mysql, their limitations are also presented and solutions are suggested to overcome the limitations, heading towards the adoption of Integrated Nosql database-Cassandra. Modern world requirements in form of Bigdata can be efficiently analyzed and interpreted using this integrated nosql database with respect to query analyzation. Future work can be taken up to conduct more experiments on Cassandra which has performed far better compared to MySQL on write operations, but showed poor performance on read operations. Hence, showing improvement in the read operations with Cassandra making necessary modifications at the selection of appropriate algorithm, compression techniques can be taken up as the future work.

References

1. Venkat N Gudivada, Dhana Rao, Vijay V Raghavan,"Nosql systems for Big Data Management " in Proceedings of 10th World Congress on Services, IEEE, doi 10.1109/SERVICES.2014.42, pp:190–197(2014).
2. Thomas Sandholm, Dongman Lee, "Notes on Cloud Computing Principles", in: Sandholm and Lee Journal of Cloud Computing: Advances, Systems and applications, 3:21, Springer 2014.
3. Divyakant Agarwal, Sudipto Das, Amr EI Abbadi, "Bigdata and Cloud Computing: Current State and Future opportunities", in: EDBT 2011/ACM, March 22–24, Uppsala, Sweden (2011).
4. Nani Fadzlina Naim, Ahmad Ihsan Mohd Yassin, Wan Mohd Ameerul Wan Zamri, Suzi Seroja Sarnin, "Mysql Database for storage of fingerprint data", in: Proceedings of 13th International Conference on Modelling and Simulation, IEEE, doi:10.1109/UKSIM.2011.62, pp:293–298, (2011).
5. Sudhanshu Kulshreshta, Shelly Sachdeva, "Performance Comparison for Data Storage-DB4o and Mysql Databases", in: IEEE 2014, 978-1-4799-5173-4/14, (2014).
6. Mehul Nalin Vora, "Hadoop-HBase for Large Scale Data", in: Proceedings of International Conference on Computer Science and Network Technology, IEEE, December 24–26, pp: 601–605, (2011).
7. Gansen Zhao, Weichai Huang, ShunlinLiang, Yong Tang "Modelling MongoDB with Relational Model", in: Proceedings of Fourth International Conference on Emerging Intelligent Data and Web Technologies, IEEE, doi:10.1109/EIDWT.2013.25, pp:115–121 (2013).
8. Katarina Grolinger, Wilson A Higashino, Abhivav Tiwari and Miriam AM Capretz"Data Management in Cloud Environments: Nosql and Newsql data stores",in Journal of Cloud Computing: Advances, Systems and Applications, Volume 2, doi:10.1186/2192-113X-2-22, (2013).