Novateur Publication's
International Journal of Innovation in Engineering, Research and Technology [IJIERT]
ICITDCEME'15 Conference Proceedings
ISSN No - 2394-3696

EVALUATING CASSANDRA, MONGO DB LIKE NOSQL DATASETS USING HADOOP STREAMING

Dahatonde Varsha Sukhdev,
Department of Computer Engineering, G.H.Raisoni College of Engineering, chas, Ahmednagar India
vaishudahatonde@gmail.com
Ashish Kumar,
Department of Computer Engineering, G.H.Raisoni College of Engineering, chas, Ahmednagar India
ashish.kumar@raisoni.net

ABSTRACT
An unstructured data poses challenges to storing data. Experts estimate that 80 to 90 percent of the data in any organization is unstructured. And the amount of unstructured data in enterprises is growing significantly— often many times faster than structured databases are growing. As structured data is existing in table format i,e having proper scheme but unstructured data is schema less database So it's directly signifying the importance of NoSQL storage Model and Map Reduce platform. For processing unstructured data, where in existing it is given to Cassandra dataset. Here in present system along with Cassandra dataset, Mongo DB is to be implemented. As Mongo DB provide flexible data model and large amount of options for querying unstructured data. Whereas Cassandra model their data in such a way as to minimize the total number of queries through more careful planning and renormalizations. It offers basic secondary indexes but for the best performance it's recommended to model our data as to use them infrequently. So to process
KEYWORDS: Unstructured data, schema less database, secondary indexes, denormalization.

INTRODUCTION
Structured data is generally in the form of relational database i.e relational data and can be accessed through predesigned fields. In contrast unstructured data doesn't fit into any pre-defined data models. Bigdata is used to analyze the structured as well as unstructured data. As unstructured data grows more rapidly, as user content of database is text. For about 40 years, files were likewise most often comprised of just text. Now users want rich content, not just plain text. To handle huge amount of unstructured data by using different programs under varied conditions becomes difficult. The main problem while handling the NOSQL database is about the storage and search of the data requires high computational resources. NoSQL database are Non-relational, Schema-less data model, having low latency, highly scalable and gives high performance. NoSQL database is coded in district programming languages and available as open source software. Objective of this paper is to handle the unstructured data using widely used NoSQL database system, Cassandra and MongoDB [1]. The existing work uses Map Reduce pipeline that is adopted by Hadoop streaming and MARISSA. For evaluation of data the pipeline have three stages: Data preparation, Data Transformation and Data Processing [1]. This paper is organized as follow. Section 2 provides an introduction for NoSQL database, Cassandra and Mongo DB system. We discuss related work in section 3 and we present, at section 4 the proposed architecture of the system.

NOSQL DATABASE
"A NoSQL or Not Only SQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases"[2]. A major difference from relational databases is the lack of explicit data scheme. NoSQL databases infer scheme from stored data, if it requires it at all, depending on which model was used. The main benefit of using different data models is that they are very good at what they do. At the same time, don't force them to do something they aren't designed for. This means that it is of the upmost importance to understand and correctly use the data model when choosing NoSQL solutions.Generally, data models in NoSQL are grouped into four categories. However, particular NoSQL solutions may incorporate several models at once.

KEY-VALUE (K-V) STORES
K-V store is the simplest data model. The key is a unique identifier for a value, which can be any data application needs stored. This model is also the fastest way to get data by known key, but without the flexibility of more advanced querying. It may be used for data sharing between application instances like distributed cache or to store user session data.

DOCUMENT STORES

Document store is a data model for storing semi-structured document object data and metadata. The JSON format is normally used to represent such objects. Documents can be queried by their properties in a similar manner to relational databases but aren't required to adhere to the strict structure of a database table. Additionally, only parts of the object may be requested or updated.

Generally speaking, document stores are used for aggregate objects that have no shared complex data between them and to quickly search or filter by some object properties.

COLUMN-ORIENTED STORES

A more advanced K-V store data model is a column family. These are used for organizing data based on individual columns where actual data is used as a key to refer to whole data collections. It is similar to a relational database index; however a column family may be an arbitrary collection of columns. There are more complex aggregation structures like super columns and super column families to allow access to the data by several keys.

This particular approach is used for very large scalable databases to greatly reduce time for searching data. It is rarely used outside of enterprise level applications.

GRAPH DATABASES

As the name implies, this data model allows objects to link and be linked by several other objects thus constructing a graph structure. Links usually have additional properties to describe the relation between objects. Graph databases map more directly to object oriented programming models and are faster for highly associative data sets and graph queries. Furthermore they typically support ACID transaction properties in the same way as most RDBMS.

CASSANDRA

Cassandra's architecture is made of nodes, clusters, data centers and a partitioner. A node is a physical instance of Cassandra. Cassandra does not use a master-slave architecture; rather, Cassandra uses peer-to-peer architecture, which all nodes are equal. A cluster is a group of nodes or even a single node. A group of clusters is a data center. A partitioned is a hash function for computing the token of each row key.

When one row is inserted, a token is calculated, based on its unique row key. This token determines in what node that particular row will be stored. Each node of a cluster is responsible for a range of data based on a token. When the row is inserted and its token is calculated, this row is stored on a node responsible for this token. The advantage here is that multiple rows can be written in parallel into the database, as each node is responsible for its own write requests. However this may be seen as a drawback regarding data extraction, becoming a bottleneck. The MurMur3Partitioner [17] is a partitioner that uses tokens to assign equal portions of data to each node. This technique was selected because it provides fast hashing, and its hash function helps to evenly distribute data to all the nodes of a cluster.

LITERATURE SURVEY

E. Dede have proposed two different approaches, one working with the distributed Cassandra cluster[1] directly to perform MapReduce operations and the other exporting the dataset from the database servers to the file system for further processing. They also gives an approaches in solving the challenge of integrating NoSQL data stores with Map Reduce for non-Java application scenarios, along with advantages and disadvantages of each approach. Also compare Hadoop Streaming alongside their own streaming framework, MARISSA, to show performance implications of coupling NoSQL data stores like Cassandra with MapReduce frameworks that normally rely on file-system based data stores. Elif Dede have proposed Cassandra's Random Partitioned distributes data evenly, improving Hadoop's performance by a factor of 3 [3]. Also Increasing the replication-factor on Cassandra does not affect Hadoop turn around time; leveraging range scans reduces read repair calls on replicas, immunizing Hadoop from replication related performance degradation. CPU intensive loads perform better using Hadoop-native, but the difference using Cassandra is minimal.Z. Fadika [4] have proposed evaluate Hadoop specifically for data-intensive scientific operations -- filter, merge and reorder-- to understand its various design considerations and performance trade-offs. In this paper, we evaluate Hadoop for these data operations in the context of High Performance Computing (HPC) environments to understand the impact of the file system, network and programming modes on performance. Many research works [5-8] present results involving the performance of a Cassandra database system for massive data volumes. In this paper, we have decided to evaluate the performance of Cassandra NoSQL database system specifically for genomic data.

PROPOSED SYSTEM

This proposed system consists of following components:

1. Data Preparation: Data Preparation, Figure a, is the step of downloading the data from Cassandra servers to the corresponding file systems – HDFS for Hadoop Streaming and the shared file system for MARISSA. For both of these frameworks this step is initiated in parallel. Cassandra allows exporting the records of a dataset in JSON formatted files [9]. Using this feature, each node downloads the data from the local Cassandra server to the file system. In our experimental setup, each node that is running a Cassandra server is also a worker node for the Map Reduce framework in use.
2. Data Transformation (MR1): Cassandra allows users to export datasets as JSON formatted files. As our assumption is that the Map Reduce applications to be run are legacy applications which are either impossible or impractical to be modified and the input data needs to be converted into a format that is expected by these target executables. For this reason, our software pipeline includes a Map Reduce stage, Figure 1b, where JSON data can be transformed into other formats. In this phase each input record is processed to be converted to another format and stored in intermediary output files. This step does not involve any data or processing dependencies between nodes and therefore is a great fit for the Map Reduce.
3. Data Processing (MR2): This is the final step of the Map Reduce Streaming pipeline. We run the non-java executables, over the output of MR1 .

To show the full operation, we assume the time taken for Data Preparation and data Transformation under each Mapreduce framework and repeat our comparisons[1].

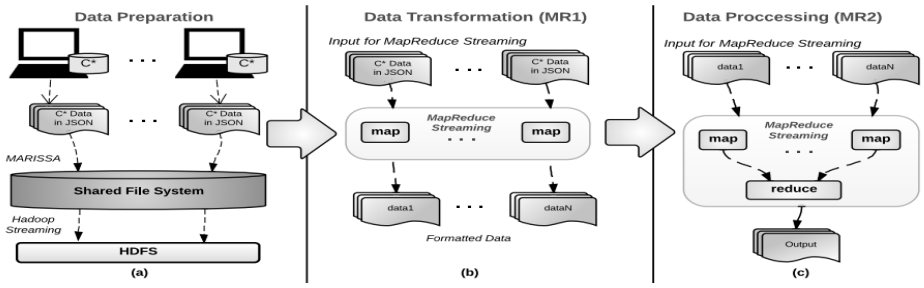

Figure: Block diagram of Proposed System

CONCLUSION

NoSQL databases or new tests using Cassandra with different hardware configurations seeking improvements in performance. Comparing the performance of Cassandra to the Mongo DB database will definitely help in the processing of unstructured data. Further it is possible to outline new approaches in studies of processing the unstructured data

REFERENCES

1. E. Dede, B. Sendir, P. Kuzlu, J. Weachock, M. Govindaraju, "A Processing Pipeline for Cassandra Datasets Based on Hadoop Streaming "DOI 10.1109/BigData.Congress.2014.32, 2014 IEEE International Congress on Big Data.
2. NoSQL wiki, https://en.wikipedia.org/wiki/NoSQL
3. Elif Dede, Bedri Sendir, Pinar Kuzlu, Jessica Hartog, Madhusudhan Govindaraju : "An Evaluation of Cassandra for Hadoop", Grid and Cloud Computing Research Laboratory SUNY Binghamton, New York, USA , 2013 IEEE Sixth International Conference on Cloud Computing
4. Z. Fadika, M. Govindaraju, R. Canon, and L. Ramakrishnan. Evaluating Hadoop for Data-Intensive Scientific Operations. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pages 67–74. IEEE, 2012
5. Z. Ye and S. Li, "A request skew aware heterogeneous distributed storage system based on Cassandra," in Proceedings of the International Conference on Computer and Management (CAMAN '11), pp. 1–5, May 2011.
6. G. Wang and J. Tang, "The NoSQL principles and basic application of cassandra model," in Proceedings of the International Conference on Computer Science and Service System (CSSS '12), pp. 1332–1335, August 2012.

7. B. G. Tudorica and C. Bucur, "A comparison between several NoSQL databases with comments and notes," in Proceedings of the 10th RoEduNet International Conference on Networking in Education and Research (RoEduNet '11), pp. 1–5, June 2011.
8. Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in Proceedings of the 14th IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM '13), pp. 15–19, August 2013.
9. Cassandra wiki, operations. http://wiki.apache.org/cassandra/Operations.
10. M. Klems, D. Bermbach, and R. Weinert, "A runtime quality measurement framework for cloud database service systems," in Proceedings of the 8th International Conference on the Quality of Information and Communications Technology (QUATIC '12), pp. 38–46, September 2012.