IEEE International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), August 01-02, 2014, Dr. Virendra Swarup Group of Institutions, Unnao, India

SETiNS: Storage Efficiency Techniques in No-SQL database for Cloud Based Design

Vandana Bhatia and Ajay Jangra
U.I.E.T , Kurukshetra University
Kurukshetra, India
vbhatia91@gmail.com , er_jangra@yahoo.co.in

Abstract— Cloud computing and Cloud data storage have become the preferred method for delivering services over a network. As the data in today's scenario is proliferative, there is a need to keep it safer which requires organizations to integrate their data usage and management patterns. To meet these requirements, No-SQL due to its better scalability, is a vantage over RDBMS. This paper presents a prototype implementation of storage efficiency techniques in No-SQL database (SETiNS) in which deduplication and compression together are used to avoid unnecessary storage purchases and reduce storage space for unraveling large volumes of data handling problems. The improved storage efficiency for No-SQL database is evaluated by using the combination of the MapReduce capabilities of Hadoop with the schema-less database MongoDB. The result shows improved storage efficiency by saving subsequent storage space and network bandwidth in the case of internode data transfer.

Keywords- Cloud Computing; Storage Efficiency; Deduplication; Compression; No-SQL; MapReduce; Hadoop; HDFS; MongoDB.

I. INTRODUCTION

Cloud Computing is an emerging technology in I.T and business industry which provides elasticity, scalability and cost-efficiency using internet. In recent years, database outsourcing has become a crucial component of cloud computing technology. Social Networking sites like Twitter and Facebook generates 500 plus terabytes of data every day. Stock exchange generates more than 1 TB data per hour. [1] From large video files, music files, pictures to sensitive documents, the cloud can virtually back up all the files and folders and alleviates the endless and costly search for extra storage space as long as user have internet connected.

All the live unstructured or semi-structured data that is added each day to cloud can't be handled by traditional RDBMS. To manage this enormous data, a new concept has come into existence known as No-SQL (Not only SQL) with the objective to provide better scalability, elasticity, and availability in cloud network. No-SQL database has de-normalized structure with Eventual Consistency (BASE) instead of Traditional ACID properties to dynamically add new attributes to data records. It has significant and growing industry for big data and real-time web applications. They have ability to replicate and to distribute partitioned data over many servers resulting in a significant amount of redundant data.

The major problem with processing of large amount of data is storage, as data management cost is much higher than the initial acquisition cost of data. There are many challenges that need to be solved to make cloud suitable for backup, shared file systems, and other internal system applications, among them cost, scalability, performance, security, and interface mismatch are the main. Among ways to surmount this challenge includes on-premise storage or network devices, using backup software, datacenter based offsite services, and cloud-based or online solutions [2]. To manage and process the data efficiently, it is important to deploy techniques such as data deduplication and compression for better storage efficiency and network bandwidth utilization.

This paper proposed a novel design for efficient storage technique in No-SQL Database known as SETiNS to employ on cloud. Data is fetched from HDFS in the form of metadata. Then Deduplication is done using MapReduce, which gives the key-value pairs. The respective pointer table is created and stored in a file with all key-value pairs, on which final compression is applied. The tool used for MapReduce is Hadoop. To create pointer table and to integrate all the key-value pair, MongoDB is used because it is a document oriented database and is highly scalable with high availability and performance. [3] By using MongoDB, we get the output in documented form on which it is easy to employ compression. The output from MongoDB is stored in HDFS back which is compressed using Gzip, which is a Hadoop library, to make the task of compressing and decompressing much faster and easier.

II. RELATED WORK

There is some recent work in which Deduplication has been applied on the data by using the idea from hashing and MapReduce. The platforms used for them were Hadoop distributed filesystem, Amazon EC2, Cassandra, etc. Daniel J. Abadi [4] in his work discussed the limitations and opportunities of deploying data management issues on some emerging cloud computing platforms. After doing all his research, he found that there is a need of a new DBMS, designed specifically for cloud computing environments.

He et al. in paper [5] mentioned about various deduplication techniques in which the basic principle is to maintain a single copy of the duplicate data, and all the duplicate blocks must point to that copy. This can be achieved at block level, file level or byte level. About the problems that may arise during deduplication, Zhu et al. [6] discussed a method to solve the problem of disk bottlenecks while deduplication. Machines with insufficient RAM pose a challenge of maintaining large amounts of metadata about the stored blocks. Hence, they concluded that a major issue to be addressed is to identify and eliminate duplicate data segments on these low-cost systems. Nagapramod et al. [7] experimented with different chunking techniques against real life data to investigate the Data deduplication to conclude that no one algorithm can fit at all considering no multiple backups are there.

For the study of Compression, Balachandran et al. [8] proposed to use runs of hashes for file recipe compression. They find out that if a run of hashes occurs twice in the data stream, they replace it with the fingerprint of the first chunk and the length of the repeated sequence. Khaing et al. explained the importance of cloud distributed file system and removed the data redundancy in the files like pdf, word and text in cloud based private network and noticed the significant amount of storage savings. [9]

III. No-SQL DB Storage for Cloud Computing

With the expansion and increasing popularity of Cloud Computing, NoSQL databases are becoming the preferred choice of storing data in the Cloud. Platforms like Hadoop and MongoDB are well suited for storing and handling such database. Main types of No-SQL database that are used these days mostly are Key-value stores, Column oriented DBs, Document DBs, Graph Databases, etc.

No-SQL databases in this paper, adopts the Document oriented data model which is a widely used model for cloud DBMSs. These DBMSs support many features required in cloud, such as elastic scalability to match varying user demands, data replication, schema-less data storage, and many other features. Generally, the key-value data model on cloud implements the column oriented and document oriented approach, which is adopted from Google's Bigtable. [10] In document oriented data model of No-SQL, documents are stored with a key and this key is used to retrieve the document. The storage is done in documented form to better deal with larger objects as key-value DBs were designed to deal with primarily smaller objects.

All the No-SQL databases due to their highly de-normalized structure tend to store significant amounts of redundant data. For any organization, management of growing amount of redundant No-SQL data is difficult. To increase the efficiency of management environment, there is a need to identify right management tool which provide good storage Efficiency. Storage Efficiency can be defined as the ability to store and manage data that occupies the least amount of storage space with little or no impact on performance, which results in a lesser total operational cost. Due to rapid growth in networking and business demands, cost of resources needed in terms of space and network utilization also expand. To store terabytes of data, it is beneficial to use techniques like deduplication and compression for gaining significant savings in required raw storage.

IV. System Archtecture of SETiNS

In figure 1, the flow diagram of the proposed framework has been shown in multiple layers. First data is fetched from Hadoop Distributed file system (HDFS). It is designed for storing very large data files with streaming data access patterns, running on clusters of various nodes that are hundreds of megabytes, gigabytes, or terabytes in size. [11]

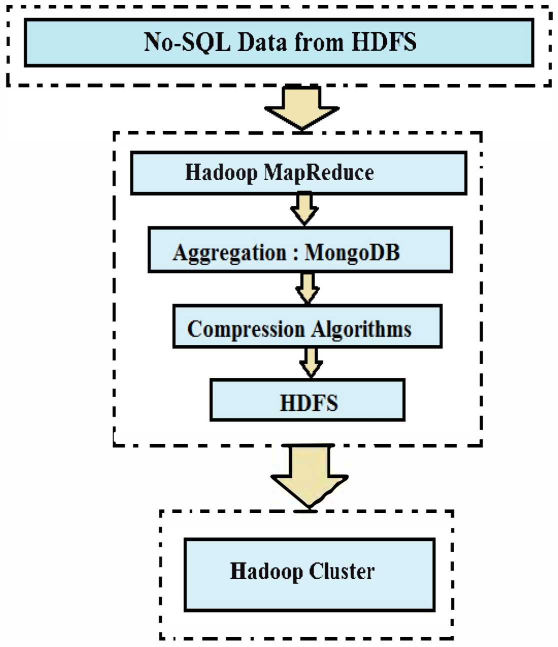


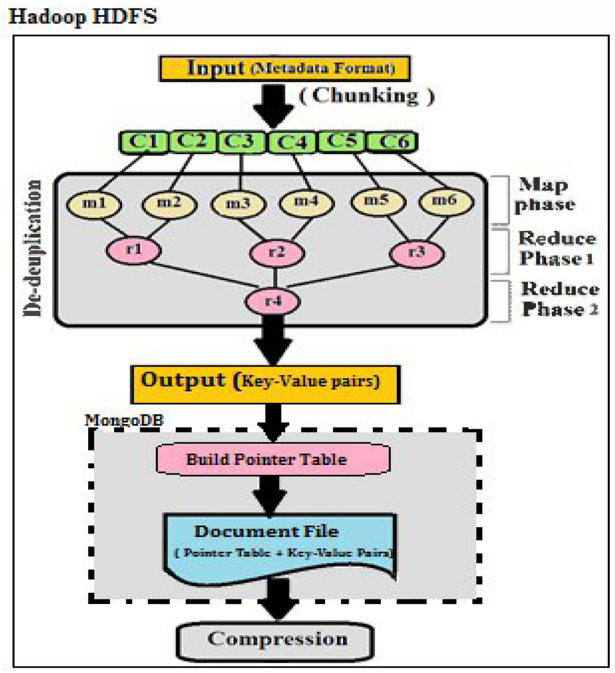
Figure 1. Flow Chart of the Proposed System

Data are stored in the form of flat text files oftenly in cloud. It is not efficient to store data in this raw form because it increases OS overhead in fetching and storing the data. Traditional RDBMS is not a suitable platform to handle the bulk data. MapReduce works great in such Conditions. To applying SETiNS efficiently for large data, the file is first deduplicated using MapReduce paradigm in Hadoop framework. The output of MapReduce phase are the various key-value pairs, on the basis of which pointer table is created using MongoDB. Key-value pairs with the pointer table are now stored as a single document. Using MongoDB, a document may contain further various nested documents with different key-value pair and key-array pairs. So, MongoDB is used for this purpose, as document based N0-SQL storage is more efficient and easy in handling. [3]

To get more efficient storage, compression is applied further. The Compressed file is then saved in HDFS. Now the file after applying SETiNS is ready to transfer on other nodes which also save bandwidth consumption because of the lesser size.

V. DETAILS OF EXECUTION PHASE

Hadoop and MongoDB are the open source platforms. For execution, Hadoop 2.2.0 and MongoDB 2.6.1 are installed in 32 bit Ubuntu 10.4 on a dual boot system with 4 GB RAM and hard disk of 500 GB. It is considered as a server node or master node for distributed file system architecture. After Configuring Hadoop and MongoDB, the native libraries of Hadoop for compression are placed. In this approach Hadoop GZip compression is used. All the execution is done in Eclipse Juno in which libraries for Hadoop and MongoDB are installed. To evaluate the bandwidth consumption, a client node is considered with the same configuration. In fig. 2, all the execution phases are shown



.

Figure 2. Execution Phase

Input from HDFS, in the form of metadata containing various text files, is fetched. Metadata is the key element used for chunking the records into data-sets, and to identify duplicates effectively by comparing only identical data-set types is taken. The variable size chunks are then passed to MapReduce phase for removing duplicate data.

A. *Deduplication*

Data Deduplication segments a stream of data into variable length files and writes those files to disk. Along the way, it creates a digital signature by using md5 hashing which act as _id to identify the data uniquely. Deduplication is done by using the MapReduce Paradigm, which follows the master-slave design where the master node is responsible for managing the jobs of giving the start instruction to the worker nodes, and assign the map/reduce tasks. Hadoop hashes all keys in the Map step and ensures that all values belonging to the relative key for corresponding field end up on the same Reducer. Here deduplication is done at file and sub-file level.

B. *MapReduce Paradigm*

MapReduce can be seen as an alternative to traditional RDBMS. MapReduce works well on unstructured and semi-structured data. After receiving instruction from the master node, now each worker can run a map or a reduce task at any instance. It is designed to interpret the data at processing time. The job execution begins by splitting the input data and assigning it to map tasks individually. When a worker or tasktracker completes executing a particular map task, it stores the map results as intermediary key-value schema in memory. The intermediate results of each map task are now assigned to the existing reducer. A reduce task begins by retrieving its corresponding intermediary results from all map outputs and then the reduce function is applied.

In SETiNS, to get better storage efficiency in less processing time, the intermediate key-value pairs got after every map task are not stored individually. Instead the data is kept in memory and then directly transferred to the reducers. The same tenet is applied in between two reducers. The data is not transformed into key-value pairs, but transferred as it was and is later manipulated by the reducer. These extra transformation and manipulation steps are eliminated for each pair of data-sets that gives better execution speed.

C. *Integration of the key-value pairs*

For storing the result of MapReduce phase, MongoDB is used. The purpose of using MongoDB is its high performance and easy scalability. It is best suited for No-SQL schema as it works on the concept of collection and document. A collection in MongoDB is a group of documents. Document based No-SQL Database are basically, a key-value database, where each record is stored as a document or object and can be identified by a unique ID which is provided by hashing in this case.[3] The pointer Table based on all key value pairs is created by fetching all key value pairs to MongoDB document. All key value pairs with the pointer table are stored in a document oriented file so that the original file can be recreated by combining them all together.

D. *Compression in Hadoop*

Hadoop has a pretty good read and near about optimal write performance. It uses the disk space efficiently by supporting compression algorithms that can be chosen based on the nature of the data in specific No-SQL families. It helps in saving disk I/O but have relatively higher CPU utilization for compression and decompression while writing and reading data.

In this case GZIP [12] compression algorithm is used for additional savings in storage space. It is among the native libraries of Hadoop. GZIP is based on the DEFLATE algorithm, which is a combination of Huffman coding and LZ77. It is easy to decompress the file back using DEFLATE algorithm with just a little extra execution time.

VI. RESULT AND ANALAYSIS

SETiNS is a novel approach for storage savings, so dataset taken is absolutely certain of its content and structure. The data is generated based on Size, redundancy and tuple complexity resulting in 12 text files.

A consistent backup of the databases was performed before and after applying the deduplication and compression process to record the size of the files. Similarly, the size of the backup files was measured after running the experiments on records of varying sizes. In table 1, the comparison is shown in terms of file sizes having different number of records.

Table 1 Comparison in File sizes

File Size (in kb)	No. Of rows taken at a time	File size after Deduplication	File size after deduplicated compression
26520.16	1000	20580.24	13510.68
55290.66	2000	41980.40	24570.60
111610.6	5000	75770.60	29690.6
231420.4	10000	145400.8	36860.4

It can be noticed clearly that for large file, SETiNS give better result than the small file having the similar type of contents. As redundancy increases with the amount of records, SETiNS in improving storage efficiency works great for large files.

More than 50% of storage space can be freed if deduplication is used. More the duplicate data more will be the storage savings. And more than 91% of space can be saved if compression on deduplicated system is applied i.e. if SETiNS is applied on No-SQL datastore.

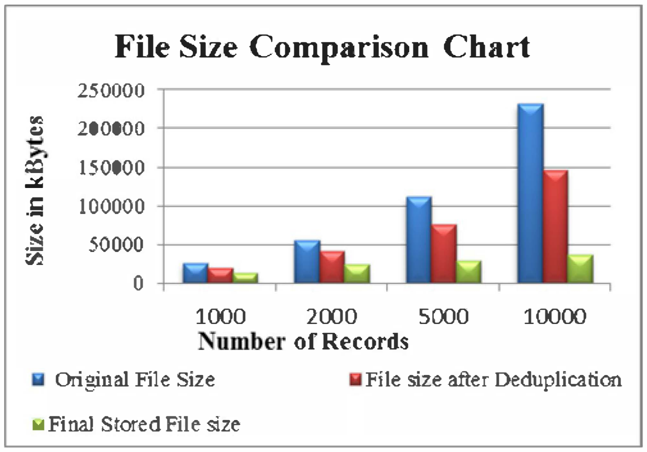


Figure 3. Comparison in terms of File size

In fig. 3, the comparison of file sizes after applying deduplication and then on it applying compression has been shown. It is clear from this that Compression can save a significant amount of space after applying deduplication for No-SQl data stores.

A. *Storage Efficiency*

Depending on the nature of the type of data, Storage efficiency Techniques give different results. For the measurement, SNIA Dictionary defined storage efficiency as the ratio of a storage system's effective capacity to its raw capacity. [13]

$$\text{Storage Efficiency} = \frac{fs - ls}{fs} \times 100 \qquad (1)$$

- o fs = Original File Size (Raw Capacity)
- o ls = File Size after applying Deduplication and then Compression.

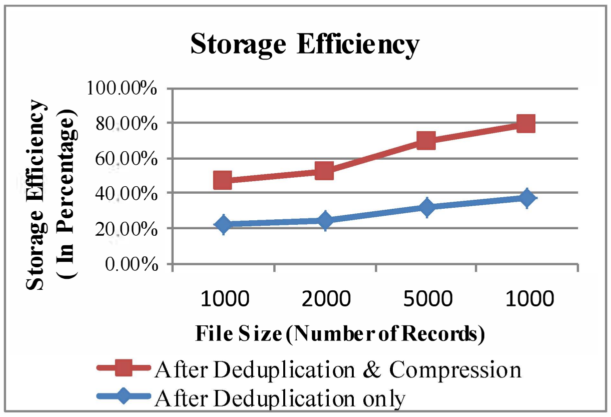


Figure 4. Storage Efficiency Comparison.

In fig. 4, the storage efficiency after performing deduplication and after performing compression on deduplicated system (SETiNS) is depicted. SETiNS saves much storage space and approximately doubles the storage savings of the deduplicated system. The average time consumed in SETiNS depends on the size and the amount of records in file.

In fig. 5, the storage efficiency percentage calculated for different database files with different tuple complexity (number of fields) is shown.

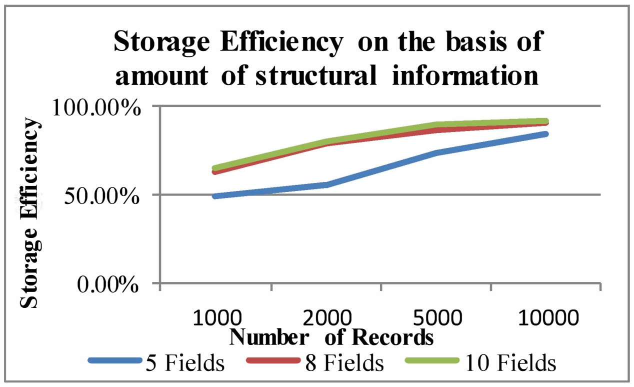


Figure 5. Storage Efficiency for files with varying structural information

Experiments have been done on different file sizes having different structural information on file level. Based on the structural information, the overall results show that with the increase in the amount of structural information, the performance of SETiNS raises. For more fields storage efficiency percentage may reach up to 90% as shown in fig. 5.

B. Bandwidth

When data has to be replicated over multiple nodes, less bandwidth is consumed for transfer because the compressed file having non-redundant data is sent on demand. It results in better bandwidth utilization and fast transfer. In fig. 6, the bandwidth that is preserved using SETiNS is shown graphically.

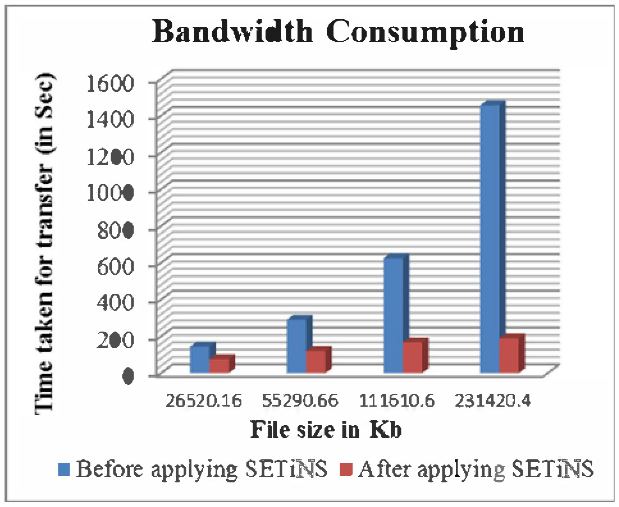


Figure 6. Bandwidth Consumption during Inter node transfer

The time taken in transfer of file from one node to another is represented in seconds. Time Consumed to transfer a file after using SETiNS is much lesser than the file transfer of the domain size file.

VII. Conclusion

Existence of duplication is a major issue for NoSQL databases. The proposed approach SETiNS uses the structural information of the data to reduce the data footprint appropriately in the NoSQL Databases. For better storage, compression can be applied after deduplication in cloud data backups. It allows easy integration with existing backup tools, rather than having to develop new ones. In SETiNS, Data Deduplication and Compression has been deployed on HDFS data store which results in less storage space and better Bandwidth utilization. For relatively large clusters and big jobs, compression can lead to substantial benefits. Deduplication helps in improving overall performance of the NoSQL column oriented data-stores by optimally saving memory used and network bandwidth.

As in this work effect of applying storage efficiency techniques has been shown on human readable text data, no matter how will it perform in controlled or uncontrolled environment. In the future it can be extended to many other types of data like images and on live data. It will also be interesting to note that weather analytical tools like pig, hive, etc. can be used to predict the data size and pattern on the basis of which deduplication and compression can be applied more effectively.

References

[1] Big-data-tradeoffs, [online] availaible at: Http://breakinggov.com/2012/11/12/big-data-tradeoffs-what-agencies-need-to-know-nists-peter-me/, (accessed on June 2 2014).

[2] Storage-efficiency, [Online] available at: http://www.crn.com/news/storage/231902774/storage-efficiency-key-to-managing-fast-growing-data.htm, (accessed on June 3 2014).

[3] MongoDB Overview - Tutorials for Swing, Retrieved from: http://www.tutorialspoint.com/mongodb/mongodb_overview.htm. (Accessed on 10 June 2014).

[4] Daniel J. Abadi, "Data Management in the Cloud: Limitations and Opportunities", IEEE Data(base) Engineering bulletin,2009, pp. 3 to 12.

[5] Q. He, Z. Li, and X. Zhang, "Data deduplication techniques," in International Conference on Future Information Technology and Management Engineering (FITME), Changzhou, China, October 2010, pp. 430–433.

[6] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in 6th USENIX Conference on File and Storage Technologies (FAST '08), 2008, pp. 269–271.

[7] N. Mandagere, P. Zhou, M. A. Smith and S. Uttamchandani, "Demystifying data deduplication," in Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion, Leuven, Belgium, 2008, pp. 12-17.

[8] Balachandran, S., And Constantinescu, C. "Sequence of Hashes Compression in Data De-duplication" in Proceedings of the 18thData Compression Conference (DCC) (2008), pp. 671–682.

[9] Cho Cho Khaing, Thinn Thu Naing,"The efficient data storage management system on Cluster-based private cloud data centre" in Proceedings of IEEE CCIS 2011,pp 235-239.

[10] F. Chang, J. Dean, S. Ghemawat,W. C. Hsieh, D. A.Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: a Distributed Storage System for Structured Data. In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association

[11] Hadoop: The Definitive Guide, 3rd Edition > Chapter 3: The Hadoop Distributed File system, Available at http://my.safaribooksonline.com/book/software-engineering-and-development/9781449328917/3dot-the-hadoop-distributedfilesystem/id2412156, (accessed June 6, 2014).

[12] "Gzip." Wikipedia, the free encyclopedia, Web. [Online] http://en.wikipedia.org/wiki/Gzip/html (accessed on june 4 2014)

[13] SNIA, "The 2013 SNIA Dictionary" pp 244. (Accessed on June 4 2014).