An Efficient De-Duplication Mechanism in Hadoop Distributed File System Environment

S. Ranjitha[1]*, P. Sudhakar[2], K.S. Seetharaman[3]

[1]Computer Science & Engineering, Anna University Chennai, Kamaraj College of Engineering and Technology, Virudhunagar, India

[2]Computer Science & Engineering, Kamaraj College of Engineering and Technology, Virudhunagar, India

[3]Computer Science & Engineering, Velammal College of Engineering and Technology, Madurai, India

***Corresponding author: E-Mail: ranjithascse@gmail.com**

ABSTRACT

Objective: Day by Day the usage of internet increase exponentially due to growth of IT sector, Technological advancement, modern gadgets usages etc. Big data plays a vital role in manipulation of structured, semi-structured and unstructured data. Handling huge amount of data in real time is highly challenging, because of exponential growth of data. In order to extract information from huge amount of data many expensive hardware resources are required. Big data is one such tool that integrates commodity for tremendous amount of data in a very cheap cost. Hadoop framework is used for running Big data application.

Findings: In this research work, an efficient De-duplication (De-du) strategy for Hadoop framework has been proposed. To implement De-du strategy, hash values of each data are computed by implementing MD5 and SHA-256 algorithms

Methods: Using these algorithms the computed hash value for a file is checked with the file that is already existing, to identify the existence of duplication. If any duplicate is found, the system won't allow the user to upload the plagiarized copy to the HDFS. Hence memory utilization is managed efficiently in Hadoop Distributed File System.

KEY WORDS: MD5, SHA256, De-duplication, Hadoop, Big data, Hadoop Distributed File System.

1. INTRODUCTION

Every day the usage of the internet get increases. Due to Advent of new technology, various online applications, technological advancement, Social network, IT corporation and the number of users to share the information leads to an exponential magnitude of data transaction over the network as well as through offline mode. In order to compute information from large hunk of data huge expensive hardware resources are required (Ruay-Shiung Chang, 2014). Big data is one such technology that integrates commodity hardware for computing extensive amount of data in a cheaper cost. Big data is a technology that handles data that exceeds the processing capacity of the conventional database systems. Very large data that streams swiftly will not fit into the structure of conventional database architectures (Zhaocong Wen, 2014). Big data is used to handle a tremendous amount of data like,

- Structured data: RelationaldatabasesuchasCSV(CommaSeparatedValues), DB(Database)
- Unstructured data: Word, PDF(Portable Document Format)
- Semi-structure data: XML(Extensible Markup Language)

Dimensions of Big data: 4Vs (Volume, Variety, Velocity and Veracity) are four most important properties of Big data.

Volume: To increase the size of data from Peta byte to Zeta byte.

Variety: The versatile forms of data such as structured, unstructured and semi-structured.

Velocity: Analysis of streaming data, i.e. flow of different sources like machines, network, social media sites, mobile devices and human interaction, to retrieve the information quickly and rapidly (speed up of data).

Veracity: Lack of confidence of data i.e. noisy data that produces inaccurate results. Such data has to be precisely validated for making the right decision (Alexandru, 2013)

Various Sources of Big Data: Loads of data are created by many sources like sensors that are used to accumulate climate information, posts from social media sites such as Facebook, Twitter, LinkedIn, Google+, YouTube, Instagram, Tumblr, Redit, and digital pictures, videos, audio, mp3, purchase transaction records in various e-commerce websites, and cell phone data.

Brief History of Hadoop: Hadoop is an open-source software framework written in Java language for distributed processing (Sathian, 2016) of very large information sets on computer groups built from commodity hardware, managed under Apache software foundation developed by Doug cutting in 2005.

Hadoop performs two operations

- Hadoop distributed File System (HDFS) for storing tremendous amount of data in the Hadoop echo system.
- Process large volume of data using Map Reduce concept.

The proposed research work concentrates on the identification of the duplicates or replica of the file in the Hadoop Distributed File System (HDFS) using De-duplication strategy.

Comprehensive study of Data De-duplication:

Data De-duplication: The large volume of data is consistent to grow rapidly in today's world ,Data de-duplication is a particular technique of compression where duplicate data is eliminated, i.e.(eliminates the extra copies of data) specifically to improve storage utilization (Yueguang Zhu, 2014). In the de-duplication process, redundant data is deleted, and keeps only one copy of the data to be stored. Elimination of redundant sub files also known as chunks, blocks, or extents.

De-duplication approach reduces the additional storage space thereby ensuring that only unique date is stored (Chan-I Ku, 2013). De-duplication take place at the

File level

Block level

File level De-duplication: It does with the duplicate copies of the same file. This is also called as Storage of Single instance (SIS). File-level de-duplication is performed to identify the multiple copies of the same file that stores it as a first copy, and then just links the other references to the first file. Just one copy gets stored on the disk/tape archive. The simple process of File-level data de-duplication with examples.

Block level De-duplication: It removes the redundancy across blocks of information that occur in non-identical files. Block level De-duplication emancipates more space than SIS, This technique is also called as Variable length De-duplication.

Existing System: The Data De-duplication technology is vastly applied in business file server, Database, RAID, Backup devices and other storage devices, where there is no footprint in Hadoop. Hadoop is extensively used in the kinds of distributed computing and massive data storage (Farag Azzedin, 2013).

Distributed File System (DFS): A distributed file system is planned to accommodate a heavy quantity of data and furnish access to this data to many clients spread across a net. There are a variety of distributed file systems that solves the problem in different ways (Sathian, 2016).

Disadvantage of Distributed File System: In distributed file system, it is restricted in its ability. The files in DFS volume all reside on a single car. This implies that it will only lay in as much data as can be stored on one car, and does not provide any reliability guarantees if that car breaks down (e.g., by replicating the files to other servers). This can overload the host if a large number of clients access the data at the same time. The drawbacks are scalability, replication, availability and very expensive to purchase a hardware host.

HDFS (Hadoop Distributed File System): HDFS is designed to be vigorous to a numeral of the issues in DFS. HDFS is designed to store a very huge quantity of information (Terabytes or Petabytes). This calls for opening the data across a heavy number of cars. It also holds much bigger file sizes than DFS. HDFS should store information reliably. If individual machines in the group malfunction, data should be still usable. HDFS provides fast, scalable access to this data. It helps a large act of clients by simply adding more details to the cluster.

Proposed system: De-duplication is a technology that compresses the data and liberates more space. Data de-duplication is a method of reducing storage needs by removing duplicate data. Only one unique instance of the data is retained on storage media. Redundant data is replaced with a pointer to the unique data copy.

Overall Design of Hadoop De-Duplication System Architecture:

Hadoop Operation: HADOOP provides five forms of services HADOOP services are Name node (Master node), Secondary name node (Master node), Data node (Slave node), Job trackers (Master node) and Task tracker (Slave node) (Deepak Mishra, 2015).

Name Node: The name node acts as master host. To make the file system namespace, name node executes the file system performance such as renaming, closing, opening files and opening directory. Name node contains all the data of data nodes and information will be kept in a tree structure.

Data Nodes: The data node act as slave. To perform read and write operations on the file system as requested by the client. Also Data nodes will perform operations such as block creation, deletion, and replication based on name node instructions.

Job Tracker: Job tracker will schedule jobs and pass over the designated tasks to task tracker

Task Tracker: Task tracker will track the project and report status to job tracker.

Secondary Name node: Secondary Name node performs some internal housekeeping for the name node. Despite its name, the secondary name node is not a backup for the name node and performs a completely different function. The purpose of a secondary Name node is to create a checkpoint in HDFS. It is just a helper node for name node.

Design of HDFS: HDFS is a file system designed for laying in very large files. Files may be terabytes (1012) or zeta byte (1024) in size with swarming data access practices, that is Write-once, read-many-times pattern, working on clusters of commodity hardware, so HADOOP doesn't require expensive and extremely reliable hardware Hadoop will run across low-cost commodity hardware like laptop, personal data processor as well HDFS architecture is similar as master-slave architecture. HDFS architecture contains name node and the data node, which can be instaledl

on the same cluster or in a different cluster. Name node to be a single cluster node (node contains one machine) and data nodes which hold a hundred machines that are (data will be maintaining multiple machines) connected.

Secure mode: Name node periodically receives a Heartbeat of data nodes and a Block report from each of the Data nodes in the bunch. And then taking in the Heartbeat tells that the Data nodes are properly worked or alive. Each Block report contains a list of all blocks on a Data node operation. Suppose, the name node fails, HADOOP will utilize the meta data replication to keep the operation specified.

HDFS Client: Client has access to the files using the HDFS Client requests to manipulate file requests such as scan, write, delete and create files & namespace

Block allocation: User data are separated into various segments and each section is split into a number of **blockages.** Each block of this segment is stored in various nodes. The default block size is 64MB-128MB. Normally Hadoop block size is 64MB, but it can be raised based on the need of HDFS configuration.

Data replication: Hadoop has three replications, one at local rock and another two at remote rack to ensure data reliability and accessibility.

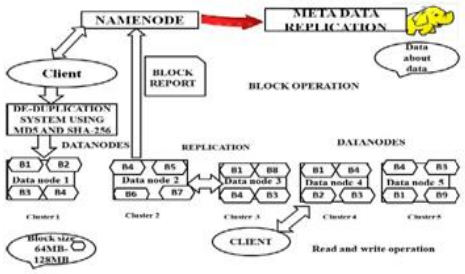

Figure.1. Hadoop De-duplication system Architecture

In Figure.1, the De-duplication strategy is implemented to avoid the duplicate file by using MD5 and SHA-256 algorithm. By using this De-duplication strategy the Hadoop system ensures that the data integrity and efficient use of memory place

Data Replication in the Hadoop system: Data Replication is the operation of copying (duplicating) data from more than one site or node that make up a Hadoop distributed file system. This is necessary for improving the availability of information in the Hadoop system. Normally Hadoop has a three replication factor. The replication factor might increase based on the Hadoop configuration setup. Due to this replication factor, there will be increase in storage space in the Hadoop environment (Neha Kurav, 2015).

File manipulation in the Hadoop environment: Figure.3, indicates that, the system there are three main steps to store a file. Foremost, create a file hash value at the customer side. Second, identify any duplication of file available in the Hadoop system or not. Third, store the file in the data node by using De-du strategy. In the second level, HDFS keeps all the file hash values. It will compare the new hash value with existing values. If does not exist, a new hash value will be stored in the HDFS, and then will ask to upload the file into the Hadoop system.

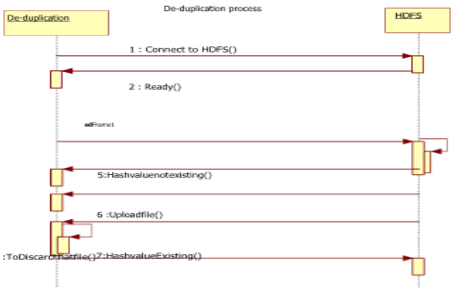

Figure.2. Procedure for storing a files

Techniques Used In the Proposed System:

Hash Based De-Duplication: Hash based De-duplication method use a Hashing algorithm to identify "chucks"of data. Commonly used algorithm are Secure Hash Algorithm (SHA-256) and Message Digest (MD5).When file is processed by a hashing algorithm, a hash is created that represent the data. A hash is a bit string (2^{64} for MD5 and 2^{128} for SHA-256) that represent the file processed. If the same data is processed through the hashing algorithm multiple times, the same hash value is created each time (Jyoti Malhotra, 2013). MD5 and SHA-256 hash functions to calculate the file's hash value and then pass the value to HDFS (Hadoop Distributed File System).

2. EXPERIMENTAL SETUP

The following experimental setup is made to validate the De-duplication system in a Hadoop single node cluster. HDFS is installed in ubuntu14.04 version, which uses Hadoop 1.2.1 version. Hadoop configuration is written in Java 1.7.0_65 version. De-duplication strategy using MD5 and SHA-256 hash functions to calculate

the file's hash values. To implement De-duplication techniques various file types and different size of files are used in De-du experiment. File format used in this experiments are structured, unstructured and semi-structure.

3. EXPERIMENTAL RESULTS

Structured file format like csv, db is considered for De-du experiment. Various file sizes ranging from 10KB to 1GB are considered for this experiment. Each experiment outcomes are projected in the table.1.

Table.1. Represents the time taken for MD5 and SHA-256 algorithms.

Size of file	MD5 Time taken (ms)	SHA-256 Time taken (ms)
12.1KB	846	841
20.2KB	868	863
1MB	887	963
5MB	1160	1172
13.6MB	1138	1184
24.3MB	1182	1273
29.6MB	2208	2294
40MB	2383	2276
1.1GB	3948	7173

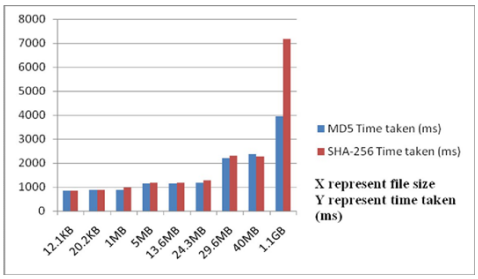

Figure.3. Time complexity graph for structured file format

Unstructured file format like pdf, text, document considered for De-du experiment. Sizes ranging from 12.4KB to 901MB are considered for this experiment. Each experiment outcomes are projected in the table.2.

Table.2. Represents the time taken for MD5 and SHA-256 algorithms

Size of file	MD5 Time taken (ms)	SHA-256 Time taken (ms)
12.4KB	843	838
18.2KB	865	861
1MB	987	963
13.6MB	1184	1132
38.6MB	2318	2213
901MB	3459	3129

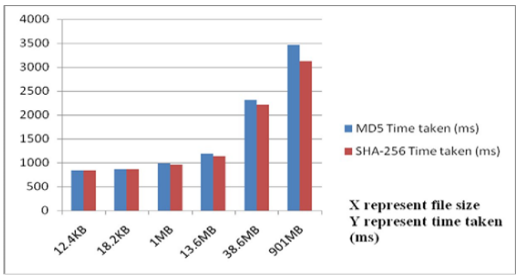

Figure.4. Time complexity graph for Unstructured file format

Semi-structured file format like xml is considered for De-du experiment. Sizes ranging from 20.1KB to 867MB are considered for this experiment. Each experiment outcomes are projected in the table.3.

Table.3. Represents the time taken for MD5 and SHA-256 algorithms.

Size of file	MD5 Time taken (ms)	SHA-256 Time taken (ms)
20.1KB	832	828
10.2MB	1132	1162
15.6MB	1154	1186
70.6MB	1896	2390
867MB	4219	6754

Findings: The De-du strategy is implemented with respect to MD5 and SHA-256 algorithm to generate hash values of various file formats and different file size. The experimental results projects the following comparison.

The concluding remarks from Figures 4 and 6 are: the time complexity of a structured and semi structure file format when using MD5 algorithm is low, compare to that of SHA-256 algorithm. While using unstructured files the time complexity of MD5 algorithm is high compare to that of SHA-256 algorithm and it is projected in Figure.5. Hence, experimental results shows that MD5 algorithm works efficiently over SHA-256 algorithm.

4. CONCLUSION

Big data is the one of the emerging technology in today trends. Hadoop tool is used for running big data application. However, the arrival of Hadoop some difficult issues are considered. One of the issue is File duplication in the Hadoop System. File De-duplication is a useful techniques for eliminating duplicate copies of file in the Hadoop system. The proposed research focuses on identification of file duplication by implementing De-duplication strategy. The experimental result shows that, MD5 algorithm works efficiently when compared to SHA-256algorithm.

REFERENCES

Alexandru Adrian TOLE, Big Data Challenges, Database Systems Journal, 4, 2013.

Chan-I Ku, Guo-HengLuo, Che-Pin Chang, Shyan-Ming Yuan, File De-duplication with Cloud Storage File System, 16th International Conference on Computational Science and Engineering, IEEE, 2013.

Deepak Mishra, Sanjeev Sharma, Comprehensive study of data duplication, International conferences on cloud, Big data, 2015.

FaragAzzedin, Towards A Scalable HDFS Architecture, Collaboration Technologies and System (CTS), International Conference on Cloud, 2013.

Jyoti Malhotra, Priya Ghyare, A Novel Way of De-duplication Approach for Cloud Backup Services Using Block Index Caching Technique, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 3 (7), 2013.

Neha Kurav, Preeti Jain, A Parallel Architecture for Inline Data De-duplication Using SHA-2 Hash, International Journal of Advanced Research in Computer Science and Software Engineering, 5, 2015.

Ruay-Shiung Chang, Chih-Shan Liao, Kuo-Zheng Fan and Chia-Ming Wu, Dynamic De-duplication Decision in a Hadoop Distributed File System, International Journal of Distributed Sensor Networks, Article ID 630380, 14pages, IEEE, 2014.

Sathian D, RIlamathi R, Praveen Kumar R, Amudhavel J, Dhavachelvan P, A Comprehensive survey on taxonomy and challenges of Distributed File System, Indian Journal of Science and Technology, 2016.

Shilpa Manjit Kaur, BIG Data and Methodology -A review, International journal of advanced research in computer science and software engineering, 3 (10), 2013.

Thekkath CA, T mann, A scalable distributed file system, proceedings of the 6th ACM operating system principles on, 2000.

Yueguang Zhu, Data De-duplication on similar file detection, Innovative Mobile and Internet Services in Ubiquitous Computing(IIMIS), Eighth International conference, 2014.

Zhaocong Wen, Jinman Luo, Huajun Chen, JiaxiaoMeng, Xuan Li, Jin Li, A Verifiable Data Deduplication Scheme in Cloud Computing, International Conference on Intelligent Networking and Collaborative Systems on, 2014.