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

S.Ranjitha
Dept of Computer Science & Engineering
Kamaraj College of Engineering and Technology
Virudhunagar, India
ranjithascse@gmail.com

K.S. SeethaRaman
Assistant Professor
Dept of Computer Science & Engineering
Velammal College of Engineering and Technology
Madurai, India
kssr@vcet.ac.in

Abstract— Real-time exposure towards dealing with large volume of data is called Big data. Big data is one of the emerging technologies in the modern era. Handling huge volume of data in a real time environment is a hectic process. To handle this issue Hadoop Distributed File System (HDFS) came into existence. Hadoop Distributed File System support data duplication to attain high data reliability. However additional utilization of storage space is needed due to duplication strategy. HDFS Storage space can be handling efficiently by implementing De-duplication strategy. The motto of the research is to eliminate file duplication by implementing De-duplication strategy. A Novel De-duplication system using HDFS approach is introduced in this research work. To implement De-duplication strategy, hash values are computed for files using MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithms. The generated hash value for the files is checked with the existing file to identify the presence of duplication. If the duplication exists, the system will not allow the user to upload the duplicate copy to the HDFS. Hence memory utilization is handled efficiently in HDFS environment.

Keywords—MD2, MD5, SHA-1, SHA-256, SHA-512, De-Duplication, Hadoop, Bigdata, Hadoop Distributed File System

I. INTRODUCTION

A. Bigdata

Big data is a technology that handles data that exceeds the processing capacity of the conventional database systems. Very large data moves rapidly does not fit into the structures of traditional database architectures. Big data is used to handle a tremendous amount of data such like structured data, unstructured data and semi-structured data[1].

a) Structured data : Relational data like .DB,.CSV.
b) Semi Structured data : .XML ,.JSON
c) Unstructured data : .Word, .PDF

B. History of Hadoop

Hadoop is an open-source software framework written in Java for distributed processing of very large data sets on computer clusters built from commodity hardware, managed under Apache software foundation developed by Doug cutting in 2005[2].

Hadoop performs two operations

a) Hadoop distributed File System (HDFS) for storing tremendous amount of data in the Hadoop echo system.
b) Processing huge volume of data using Map Reduce concept.

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

C. Map Reduce

Map Reduce is a software framework application for distributed architecture, work with parallel on large clusters (thousands of nodes) of commodity hardware in a reliable and fault tolerant manner. Hadoop provides a framework for the analysis and transformation of very large data sets using Map Reduce concept [5]. The Map Reduce framework consists of a single Master node, Job Tracker, one Slave node and Task tracker. The Master node is responsible for scheduling the job's component tasks on the Slaves, monitoring them and re-executing the failed tasks. The Slaves execute the tasks as directed by the Master node. Figure 1 explains about simple Map Reduce operation [6].

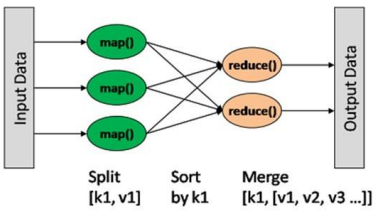

Figure 1.Map Reduce

D. *Inputs and Outputs*

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types. The key and value classes have to be serializable by the framework and hence need to implement the writable interface. Additionally, the key classes have to implement the Writable Comparable interface to facilitate sorting by the framework. Input and Output types of a Map Reduce job:

(input) < k1; v1 > → map → < k2; v2 > → combine → < k2; v2 > → reduce =< k3; v3 > (output) (1)

E. WordCount

a) WordCount is a simple application that counts the number of occurrences of each word in a given input set.
b) WordCount reads text files and counts how often words occur. The input is text files and the output is text files, each line of file which contains a word and the count of how often it occurred, separated by a tab.
c) Each mapped takes a line as input and breaks it into words. Each reducer sums the counts for each word and emits a single key/value with the word and sum of the word count.
d) As an optimization, the reducer is also used as a combiner on the map outputs. This reduces the amount of data sent across the network by combining each word into a single record.

F. Data De-Duplication

The huge amount of data is consistent to grow rapidly in today's world, Data De-Duplication is a specific form of compression where redundant data is eliminated, i.e.(eliminates the extra copies of data) typically to improve storage utilization. In the De-Duplication process, duplicate data is deleted, leaving only one copy of the data to be stored. Elimination of redundant sub files also known as chunks, blocks, or extents. However, indexing of all data is still retained that data ever be required. De-Duplication is able to reduce the required storage capacity since only the unique data is stored . De-Duplication takes place on

a) File level De-Duplication
b) Block level De-Duplication

File level De-Duplication

It eliminates duplicate copies of the same file. This is also called as Single instance storage (SIS) [3]. File-level De-Duplication performs to identify the multiple copies of the same file, that stores as first copy, and then just links the other references to the first file[4]. Only one copy gets stored on the disk/tape archive. Figure 2 explains about simple File-level Data De-Duplication.

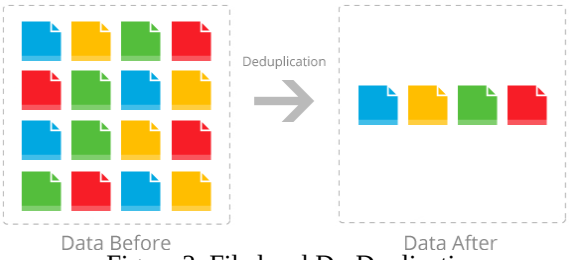

Figure 2. File level De-Duplication

II SYSTEM STUDY

A. Existing System Problems

The Data De-duplication technology is widely used in business file server, Database, RAID, Backup devices or lot storage devices, but there is no footprint in Hadoop. Hadoop is widely used in the kinds of distributed computing and massive data storage. HDFS (Hadoop Distributed File System) did not apply any common Data De-Duplication technology [7].

B. Data Replication in the Hadoop system

Data Replication is the process of copying (duplicating) data in more than one site or node that make up a Hadoop distributed file system. This is necessary for improving the availability of data in the Hadoop system. Normally Hadoop has 3 replication factors. The replication factor might be increase based on the Hadoop configuration setup. Due to Replication file occupies the extra storage space is utilized in the Hadoop System.

C. Single node Cluster vs Multi node Cluster

SingleNode

Single Node or Psuedo-Distributed Cluster is the one in which all the needy daemons (like NameNode, DataNode, JobTracker, TaskTracker and Secondary NameNode) run on the same machine. The default replication factor for a Single node cluster is one [12].

Multi node Cluster

Multi node or Fully Distributed Cluster is a typical hadoop cluster which follows Master-Slave architecture. It comprise of one Master machine (running the NameNode and JobTracker daemon) and one or more Slave machines

(running DataNode and TaskTracker daemon). The default replication factor for a Multi Node cluster is three.

III PROPOSED SYSTEM

There is no mechanism to find the Data duplication in the Hadoop system. In the proposed system introduces the mechanism called "A Novel and efficient De-Duplication system using HDFS approaches". Using this De-Duplication approach, the Hadoop system ensures the data integrity and efficient utilization of storage space. De-Duplication is a technology where it compresses the data and saves lot of space. Data De-Duplication is a method of reducing storage needs by eliminating redundant data. Only one unique instance of the data is actually retained on storage media, such as disk or tape. Redundant data is replaced with a pointer to the unique data copy [8].

A. File Storage in the Hadoop System

In the system there are three main steps to store a file. First, make a file hash value at the client side. Second, identify any duplication of file available in the Hadoop system or not. Third, store the file in the datanode by using De-Du application. In second level, HDFS keeps all the file hash values. It will compare the new hash value with the existing values. If does not exist, a new hash value will be stored in the HDFS, and then it will ask to the user upload the file in the Hadoop system .

B. Overall Design of De-Duplication System Architecture in Multi node Cluster Environment

Figure 3 infers that, the client uploads a file to the hadoop system through the hadoop interface. The De-Du strategy is implemented by calculating hash value for the files. MD2, MD5, SHA-1, SHA-256, SHA-512 algorithms used to compute hash values for this purpose. The Hash values are stored in all the Slaves and the Master. This De-Du process can be done both in Master and Slave mode. Duplication files are found by using De-Du.jar. De-Du.jar is one of the Map Reduce applications that count the number of occurrence of hash value in the Hadoop system.

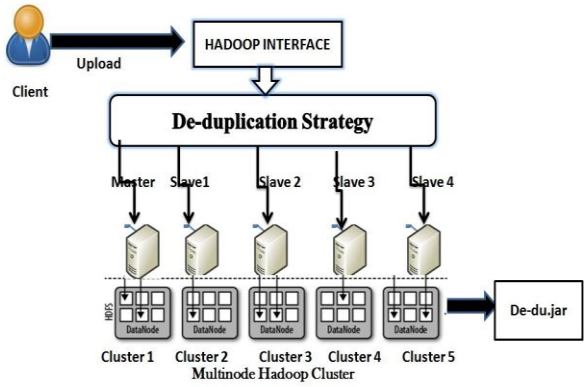

Figure 3. De-Duplication System Architecture

C. Techniques used for 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 Algorithms such as SHA-1, SHA-256, SHA-512 [10] and Message Digest Algorithms such as MD2, MD5 [11].When file is processed by a hashing algorithm; a hash is created that represent the data. If the same data is processed then the same hash value are generated. Use various hash functions algorithms to calculate the file's hash value and then pass the value to HDFS (Hadoop Distributed File System). Now De-Du.jar will compare the new hash value with the existing values. If it exists earlier in the HDFS system, HDFS system acknowledges, the content of the file is already available in the Hadoop system, No need to upload the file and it also check the number of links. If the hash value did not exist earlier, HDFS will ask the client to upload the file and update the logical path of that file [9].

D. Sequence Diagram for Identifying Duplicate Files in Mater mode

Figure 4 indicates that, the Connection is established with the Master node by the De-Duplication system. Master sends the request to all the Slaves. The Slaves responds to the Master by sending the hash value for files The Master will check the hash values of all the Slaves. If the hash value does not exist among the Slaves, the files will be uploaded. If the hash value exists files will be discarded by Master node itself.

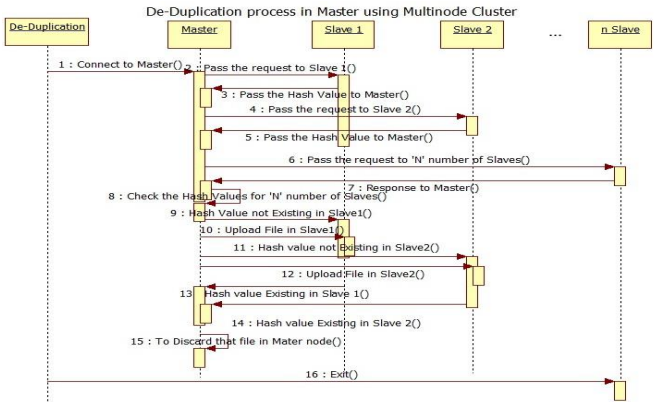

Figure 4. De-Duplication process in Master node

E. Sequence Diagram for Identifying Duplicate Files in Slave mode

Figure 5 shows that, the Connection is established by De-Duplication system with all the Slaves. De-Du system sends the request to all the Slaves. The Slaves responds to the De-Du system by sending the hash value for files. The De-Du system will check the hash values of all the Slaves. If the hash value does not exist among the Slaves, the files will be uploaded. If the hash value exists, the files will be discarded by De-Du system itself.

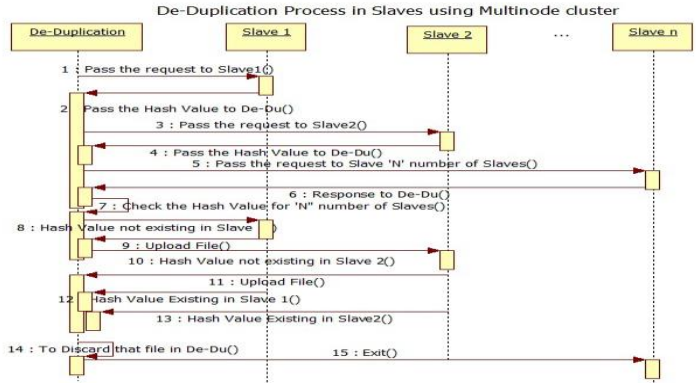

Figure 5. De-Duplication Process in Slaves Node

F. Data Integrity

It is possible that a block of data fetched from a DataNode can be corrupted. This corruption can occur because of faults in a storage device, network faults, or buggy software. The HDFS client software implements checksum, checking on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate file in the same HDFS namespace. When a client retrieves file contents, it verifies that the data is received from each Data node matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another Data node that has a replica of that block.

III RESULTS AND DISCUSSION

A. Experimental Setup

Hadoop JAVA API

Hadoop's org.apache.hadoop.fs.FileSystem is generic class to access and manage HDFS (Hadoop Distributed File System) files/directories located in distributed environment.

B. Running Hadoop on Ubuntu (Multi-Node Cluster)

The Hadoop cluster environment uses three systems (one Master and two Slaves), given below are their IP addresses.

a) Hadoop Master: 172.16.7.155 (Hadoop-Master)
b) Hadoop Slave: 172.16.7.156 (Hadoop-Slave-1)
c) Hadoop Slave: 172.16.7.154 (Hadoop-Slave-2)

First step is to install two Single-node Hadoop machines, configure and test them as local Hadoop systems. Second step is to merge both the Single-node systems into a Multi-node cluster. The Master would also act as a Slave for data storage and processing. File format used in this experiments

a) De-Duplication strategy using MD2, MD5, SHA-1, SHA-256 and SHA-512 hash functions to calculate the file's hash values.
b) To implement De-Duplication techniques various file types and different size of files are used in De-Du experiment.
c) There are structured, unstructured and semi-structure files took from OPEN DATA [13] website.
d) Various sizes of files like 10MB, 15MB, 25MB and 40MB up to 3GB are used in De-Du experiment.

C. Results

1. Unstructured file format like PDF, text are considered for De-Du experiment. This results are projected in the Table I

Table I: Represents the time taken for MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithms using Unstructured documents

Size of File	MD2 Time taken (ms)	MD5 Time taken (ms)	SHA-1 Time taken (ms)	SHA-256 Time taken (ms)	SHA-512 Time taken (ms)
500KB	1151	1100	1008	1011	942
4MB	1293	1129	1064	1052	1191
6MB	1235	1174	1059	1085	1164
12MB	1561	1110	1104	1144	1260
40MB	5056	1712	1373	1508	1445
84.4MB	4195	2198	1486	1559	1822
1.66GB	10152	9564	10425	11478	10549

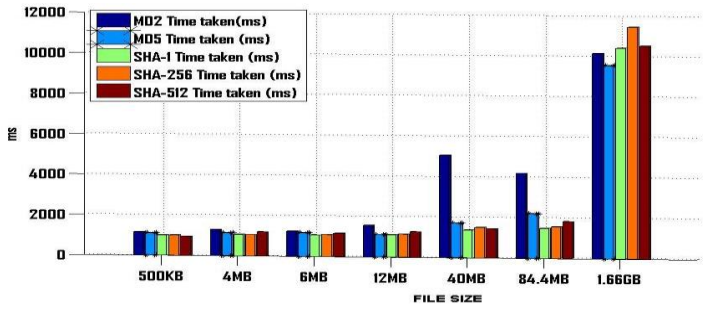

Figure 6. Time complexity graph for Unstructured file format

2. Semi-Structured file format like xml is considered for De-Du experiment. This results are projected in the Table II.

Table II: Represents the time taken for MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithms using Semi-Structured Documents

Size of File	MD2 Time taken (ms)	MD5 Time taken (ms)	SHA-1 Time taken (ms)	SHA-256 Time taken (ms)	SHA-512 Time taken (ms)
1.9KB	982	1634	1228	1494	1013
23KB	1104	844	854	1654	1545
1.6MB	5602	980	966	1980	4857
38.4MB	5887	1359	1249	3526	5478
70MB	8787	4980	2457	5698	5878
978MB	10154	8051	7145	8545	6567

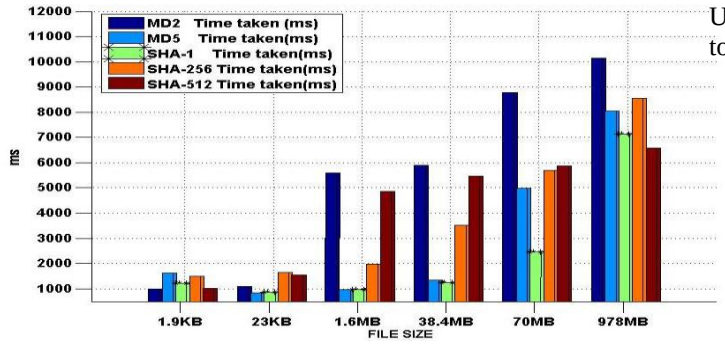

Figure 7. Time complexity graph for Semi-Structured file format

3. Structured file format like CSV is considered for De-Du experiment. This results are projected in the Table III

Table III: Represents the time taken for MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithms using Structured documents

Size of File	MD2 Time taken (ms)	MD5 Time taken (ms)	SHA-1 Time taken (ms)	SHA-256 Time taken	SHA-512 Time taken
12.1KB	1125	974	1081	1068 (ms)	1098 (ms)
1KB	1139	1055	985	1043	1182
1MB	1056	1252	1055	1291	1105
5MB	995	1357	1075	1062	1237
13.56MB	1186	1434	1156	1584	1468
24.3MB	1598	1658	1248	1681	1895
40MB	5846	2182	1430	1606	1437
60.2MB	5703	1345	1839	1753	1634
1.1GB	5975	3947	7173	7265	8012
1.54GB	6854	4578	8457	7585	8457
1.79GB	9451	6254	8754	9545	7824
2.54GB	10990	11587	10864	11424	9867

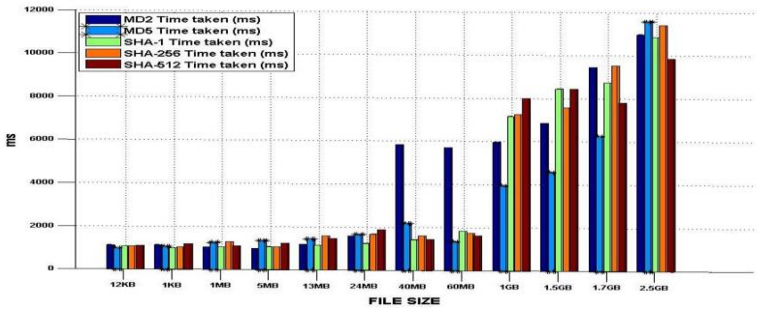

Figure 8. Time complexity graph for Structured file format

D. Discussion

The De-Du strategy is implemented with respect to MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithm to generate hash values of various files format and different size.
Using De-Du.jar the duplicate files are identified with respect to following condition

a) If the hash value count is one then, there is no duplicate file
b) If it is more than one then, duplicate file are present that indicates the current file should not be uploaded.

E. The experimental results project the following comparison.

1. The concluding remarks from Figures 6 and 8 are: the time complexity of a Unstructured and Structure file format when using MD5 algorithm is low, when compare to that of MD2, SHA-1, SHA-256, SHA-512 algorithms.
2. While using Semi-Structure files the time complexity of SHA-1 algorithm is low, it is projected in Figure 7, compare to that of MD2, MD5, SHA-256, SHA-512 algorithms.
3. Hence, an experimental result shows that MD5 algorithm works efficiently over MD2, SHA-1, SHA-256, SHA-512 algorithms.

F. Single node vs Multi node Cluster Performance with De-Duplication Strategy

The De-Du.jar take more time to find duplicate files in Single node cluster environment, when compare to that of Multinode cluster environment. Since the Map and Reduce tasks works parallel in Multinode cluster environment, there is betterment in processing speed. Figure 9 shows the performance of Singlenode vs Multi node Cluster

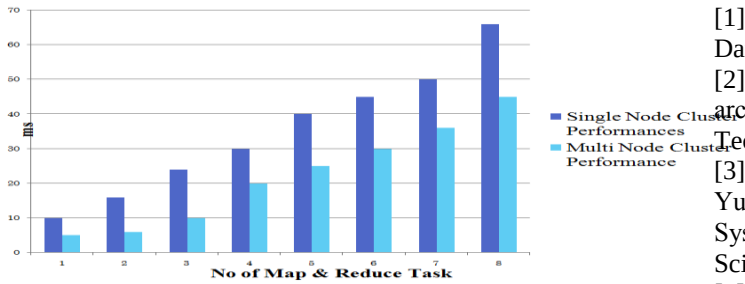

Figure 9. Performance Evaluation of De-Du Experiment

G. Memory Utilization of Hadoop Environment

a) To perform De-Duplication, hash values are computed using MD2, MD5, SHA-1, SHA-256 and SHA-512 algorithms.
b) All the above algorithms consume less memory space (size) while generating
hash values.
c) Before implementing De-Du Experiment, duplicate files are accepted by
the Hadoop Distributed File System(HDFS).
d) After implementing De-Du Experiment, duplicates files are not accepted by the Hadoop Distributed File System (HDFS), thereby, ensuring efficient utilization of memory space. A pie chart representation of the aforementioned details are showcased as Figure 10

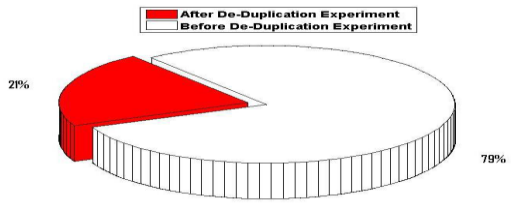

Figure 10. De-Du Experiment

IV CONCLUSION

Big data is the one of the emerging technology in today trends. Hadoop tool is used for running big data application. However, the Hadoop Distributed File System (HDFS) has some difficulties. One of the issues is File duplication in the Hadoop System. File De-Duplication is a useful technique 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 compare to MD2, SHA-1, SHA-256, SHA-512 algorithm in De-Duplication process.

V REFERENCES

[1] Alexandru Adrian TOLE. (2013), 'Big Data Challenges', Database Systems Journal Vol. IV.
[2] Azzedin, F. (2013), 'Towards a Scalable HDFS architecture', International Conference on Collaboration Technologies and Systems(CTS) .
[3] Chan-I Ku. Guo-Heng Luo. Che-Pin Chang. Shyan-Ming Yuan. (2013),'File De-Duplication with Cloud Storage File System', 16th International Conference on Computational Science and Engineering, IEEE.
[4] Deepak Mishra. Sanjeev Sharma. (2015) 'Comprehensive study of data duplication', International conferences on cloud, Big data.
[5] 'Google's Colossus Makes Search Real-Time by Dumping Map Reduce', High Scalability (World Wide Web log), 2010-09-11.
[6] Ibrahim Abaker. Targio Hashem. Ibrar Yaqoob. Nor Badrul Anuar. Salinah Mokhtar. Abdullah Gani. Samee Ullah Khan. (2014), 'The rise of big data on cloud computing :Review and open research issues'.
[7] Jyoti Malhotra. Priya Ghyare. (2013), '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, Vol. 3, Issue .
[8] Meister. Kaiser. Brinkmann. Cortes. Kuhn. Kunkel. (2012), 'A study on data De-Duplication in HPC storage systems', High Performance Computing, Networking, Storage and Analysis(SC), International Conference.
[9] Neha Kurav. Preeti Jain. (2015), 'A Parallel Architecture for Inline Data De-Duplication Using SHA-2 Hash', International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5.
[10] Piyush Gupta.Sandeep Kumar. (2014), 'A Comparative Analysis of SHA and MD5 Algorithm', (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) .
[11] Zhe SUN. Jianming YON. (2011), 'DeDu: Building a Deduplication Storage System over Cloud Computing', 15th International Conference on Computer Supported Cooperative Work in Design.
[12] Zhanye Wang. (2013), 'NCluster:Using Multiple Active Name node to Achieve High Availability for HDFS', High Performance Computing and Communication and 2013 IEEE .
[13] https://data.gov.in/