ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 5, May 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Improving HDFS Storage Efficiency by Implementing Block Level De-Duplication

Shweta U Goravanakoll[1]
Computer Science & Engineering
C.M.R Institute of Technology
Bangalore – India

Sahana. V[2]
Asst. Professor, CSE
C.M.R Institute of Technology
Bangalore – India

Abstract: ***Hadoop Distributed File System is powerful, versatile & reliable distributed storage system. It has capacity to store any vast amount of data ranging from terabytes to several petabytes. This file system is designed to store unstructured and semi structured data such as text, audio, video, images, documents etc. This HDFS is very well suited for backup recovery, sync and share, Enterprise Content Management type of applications that store enormous amount of data. The file is divided into blocks and each block is replicated. In real life it's very common to store duplicated data. Storing this duplicated data eats up huge storage and bandwidth. If we manage not to store duplicate data without impacting the data integrity then we can save enormous storage and network bandwidth. This paper implements one such technique to avoid storing duplicate data, which is called client side block level de-duplication.***

Keywords: Hadoop; de-duplication; HDFS; storage; backup; unstructured; semi-structured.

I. INTRODUCTION

The design of File systems is meant for the Operation on files. Storing and retrieving of files are controlled by File systems. A file system often stores many copies of the same files, which may lead to the redundant consumption of storage space and bandwidth. This storing of redundant files happen as file system is not aware of the structure and semantics of file contents.

The Social applications like YouTube, Drop box and Facebook, the amount of file duplication also grows when the users share and synchronize the files between each other. Many of the companies or Enterprises are facing the problem of storage space and storage cost and it has been a challenging issue for them to reduce the redundant cost of storage spaces. To get a solution for this many enterprises are seeking the de-duplication technologies to increase the storage efficiency and to reduce cost.

Hadoop DFS is powerful, flexible & reliable distributed storage system. It has capacity to store any vast amount of data ranging from terabytes to several petabytes. This file system is designed to store unstructured and semi structured data such as text, audio, video, images, documents etc. This HDFS is very well suited for backup recovery, sync and share, Enterprise Content Management type of applications that store enormous amount of data. The record is secluded into pieces and each square is reproduced. In real life it's very common to store duplicated data. Storing this duplicated data eats up huge storage and bandwidth. If we manage not to store duplicate data without impacting the data integrity then we can save enormous storage and network bandwidth. This paper implements one such technique to avoid storing duplicate data, which is called client side block level de-duplication.

It has been a complex and challenging issue for enter-prises to reduce the redundant or duplicate cost of storage spaces. According to the IDC, more than 80% of enterprises are looking de-duplication technologies to reduce cost and increase the storage efficiency. File de-duplication is the technique or task of identifying entities of the same real-world object. In a company, files are usually stored in local disk or primary storages, network file repositories, document management systems,

and secondary storages, such as backup storage or tapes. For Internet service providers, files are geographically dispersed locations. Taking Google as an example, files are stored in Google file system. Generally speaking, most companies combined the storage architecture in a mixture manner, and it further increases the complexity of file de-duplication solutions design. Advancement of de-duplication technologies are proposed based on the application characteristics. The data de-duplication techniques and analysis on applications like data warehouse applications are tried to reduce the consumption of storage for database or online transaction processing. There are also studies that tried to indicate the considerations while choosing file de-duplication solution. For the granularity considerations, de-duplication can work at either the sub file (chunk) or whole-file level. This would raise the trade-off evaluation issue in space savings and performance impacts. Using more fine-grained de-duplication, which may cause significant performance impacts, can save the more space. While shifting the paradigm to the file de-duplication of cloud systems, the trade-off of space savings and performance impacts are still major issues for data de-duplication techniques. Chun-Ho Ng's Live DFS focused on the de-duplication of VM image in the Open-Source cloud and achieved at least.

II. EXISTING SYSTEM

Existing or current system is our current Hadoop Distributed File System (HDFS). This framework is chiefly intended to store the vast measure of information. The large amount of data we can say it is of gigabyte or terabytes of data. The current hadoop distributed file system is designed to have master – slave architecture. This system is designed very nicely that it is horizontally scalable i.e., it can scale in a cluster up to hundred nodes. One more advantage of the current system is it provides the fault tolerance and easily deployed on low cost hardware.

A HDFS bunch includes a single master center point called Name Node, which manages the namespace of the report system and decides the entrance to records by customers. In addition, there are various slave or specialist hubs called Data Nodes, which manages to store the data in a capacity joined to the hubs that they keep running on. HDFS uncovered a record framework namespace and it permits client information to be put away in documents. Inside, a document is part into one or more pieces and these squares are put away in an arrangement of Data Nodes. The Name Node executes document framework namespace operations like opening, shutting, and renaming records and catalogs. It likewise decides the mapping of pieces to Data Nodes. The Data Nodes are in charge of serving read and compose demands from the document framework's customers. The Data Nodes taking into account the directions gave by the Name Node can likewise perform square creation, cancellation, and replication.

Current or Latest Hadoop Distributed File System doesn't support De-duplication feature. Due to unawareness of structure of the file contents, it stores duplicate copies of files, which results in redundant consumption of storage and network bandwidth.

III. PROBLEM STATEMENT

The storage space has become more important for the industries for enterprise content management. The features provided by the latest hadoop dfs can be utilized to hold the generous measure of data but the hadoop scattered file system without knowing the content or structure of the file can store duplicate files in its file system. This may lead to store several copies of the same file by consuming large amount of data. Another case is sometimes the file may contain most of the data similar to the content of the file, which is already stored, and a little amount of data, which is different than the stored file. In that case we are forced to store the whole new file rather than storing a small amount or modified data.

IV. PROPOSED SYSTEM

A new feature called de-duplication in hadoop distributed file system is proposed and implemented to remove or avoid the storing of duplicate data in the hadoop storage system. The main aim is to save the storage space and network bandwidth. This can be done at the client side itself i.e. before storing, rather than storing it at the storage side and then applying de-duplication.

This saves the network bandwidth to store large amount of duplicate data. In turn it saves the storage space. Instead of storing duplicate data in the hadoop system, a reference to the existing data will be provided to the new file.

A. *Advantages of Proposed System*

There are two main advantages of the Proposed System.

- Saves the storage space or Minimizes the consumption of storage space.
- Better utilization or reduction of network bandwidth.

V. THEORETICAL BACKGROUND

Theoretical background explains some topics, which are mainly required.

A. Data Storage in Hadoop DFS

Hadoop Scattered file system is used to place the records in a distributed manner. HDFS has master-slaves architecture and it reveals a document framework namespace and it stores the information in records and inside, a record is partitioned into one or more squares and these pieces are put away in Data Nodes. HDFS has one Name Node (an expert) and set of Data Nodes (slaves). The Name Node performs operations like open, close and rename documents and catalogs. Additionally defines the mapping of pieces to Data Nodes. The Data Nodes are in charge for serving the requests like read and compose from the file system's clients. And when Data Node gets instructions from Name Node, it performs block creation, deletion and replication.

HDFS is used to stock huge measure of information with no data de-duplication technologies. To elaborate this, take a simple following example. If the three identical files are stored to the HDFS using different file names, then the HDFS will store all the three files (even though all file's content is same) and also for each file it reserves three disk spaces. This means no de-duplication technologies are involved in the HDFS. In some cases only few bytes of file are changed and this leads to store the whole file in HDFS. So to provide the Efficiency for storage and bandwidth in HDFS, we are implementing a feature like block level de-duplication support for HDFS.

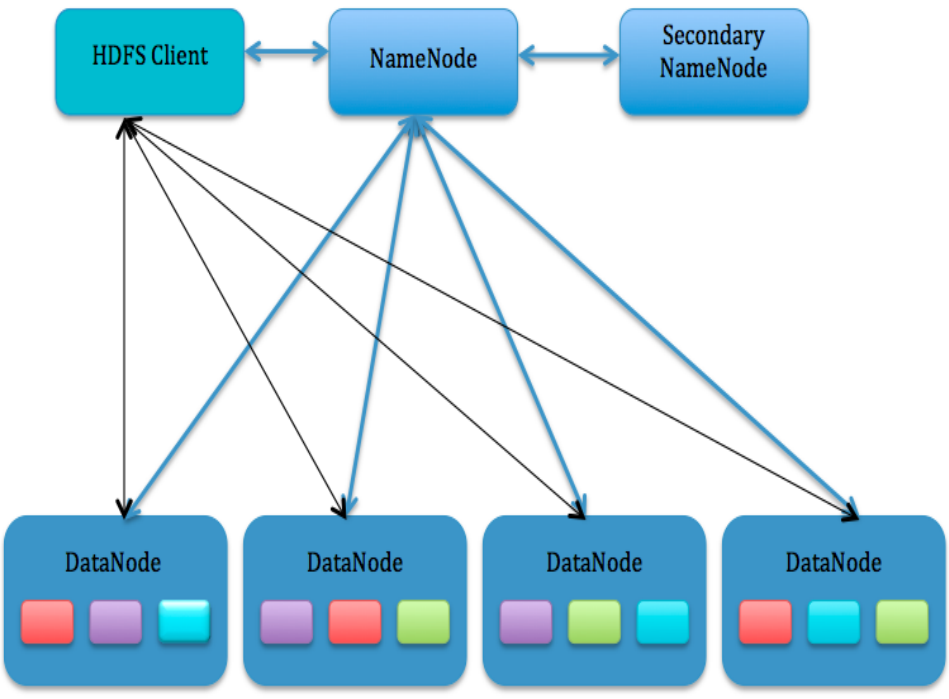

Fig 1: HDFS Architecture

B. Detection of Similar Data

There are two levels of granularities based on them its possible to identify the similar data. Those levels of granularities are named as File level and Block level. For entire record level discovery, the mapping of data is done through the hash development. For the information piece level recognition, the unique finger impression is checked through the settled size allocation.

C. Data De-duplication Technology

Data de-duplication is data reduction technique, mainly used to reduce the use of storage capacity in systems like backup systems, storage systems etc. It is a technique to remove redundant copies of the same data to provide the storage efficiency and reduce the network bandwidth. In most companies, the storage system contains redundant copies of data. For example, in different places the same file may be saved by different users or in some cases much of the data is same in non-identical files. De-duplication removes theses extra copies by saving only one copy and replacing the other copies with pointers to the original copy. Companies or Enterprises use de-duplication technique in backup and storage.

De-duplication Techniques

- **File level and Block level**

  In Record level de-duplication, de-duplication happens at the record level, which is it removes redundant duplicates of the same document. This write is called document level de-duplication or single occurrence stockpiling (SIS).

  In Chunk level, de-duplication is done at block level, which removes redundant blocks of data that occur in non-identical files. More space is freed up in block level rather than Single instance storage i.e file level. The phrase "data de-duplication" is often used as a synonym for block-level de-duplication.

- **Source and Target De-duplication:**

  Data intensive requests, for instance, reinforcement and replication, relocating files through network to target storage from source. The front-end application stands as the source and back up server to set up the unrefined information. The destination is usually storage. In Source de-duplication, the redundant data is indicated and deleted in the front-end application; this will minimizes the network transmission bandwidth and time. The disadvantage of this source de-duplication is that it will cost more for detection and deletion of duplicate data. In Target de-duplication, de-duplication task is done at target end. In this type, more resources will be utilized in redundant data processing, data transmission over networks and consumption of computing resources for redundant data detection.

  Therefore, among these two types it s recommended that source de-duplication is more feasible than target de-duplication.

D. HBase

HBase is intended to give the brisk arbitrary access to large volumes of organized data and this information model is like the Google's big table.

HBase is column-oriented distributed database and is built upon the Hadoop DFS file system and it uses fault tolerance feature provided by the hadoop-distributed file system.

HBase provides fast lookups for larger tables, for billions of records, to access single rows it provides low latency access. It utilizes hash tables internally and arbitrary access to the data will be given and stores the information in recorded HDFS documents for speedier lookups.

E. Hashing Functions

Any function that can be used to map arbitrary size data to fixed size data is called Hash function. The result of this hash function is called hash or hash values.

Secure Hashing Algorithms such as Secure Hash Algorithm 2 (SHA-2), it is an arrangement of cryptographic hash capacities SHA 224, SHA 256, SHA 384 and SHA 512 are planned by National Security Agency (NSA) and distributed by NIST in 2001. SHA-2 incorporates a noteworthy number of changes from its previous, SHA-1. SHA-2 comprises of an arrangement of four hash capacities with condensations that are 224, 256, 384, or 512bits. In SHA 1 some security flaws were recognized in 2005; in particular, a numerical shortcoming may exist, showing that a more grounded hash capacity would be attractive. In spite of the fact that SHA 2 bears some comparability to the SHA 1 calculation, these assaults have not been effectively stretched out to SHA-2. In software engineering, a crash or conflict is a circumstance that happens when two particular bits of information have the same hash esteem, checksum, unique finger impression, or cryptographic digest.

VI. System Architecture

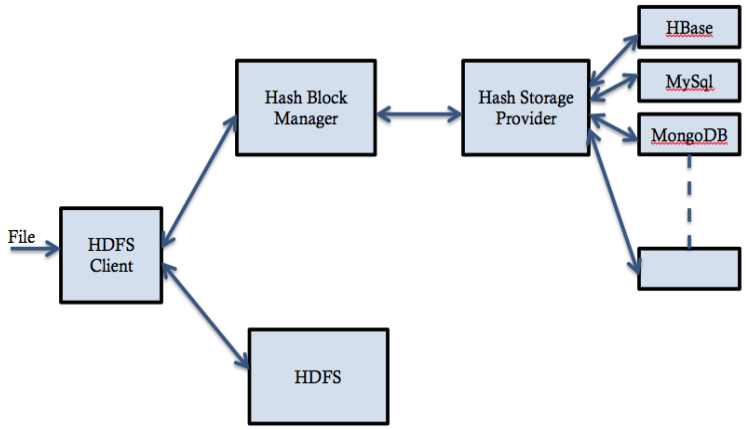

Fig. 2 Architecture of a Framework

Fig 2 shows the Architecture design where the client provides the file to be stored in HDFS. Then a namespace to that file will be created and the record will be split into pieces or chunks of stable size (configured block size). Before storing a chunk in to HDFS the existence of that block will be checked by communicating with HBase or MySql through Hash Block Manager. If the block already exists, then existing block id and file id of that block will be provided so that a reference has to be created for that new block in a Name node and no need to store that block again. If the block doesn't exist then that block will be written to the HDFS and the block Id will be returned and will be updated in HBase after writing that block successfully to the HDFS.

The operation or working of each module is specified as following.

HDFS Client

It is responsible for the communication between a client and name node and communication between client and data nodes and also responsible for the communication to the Block Hash Manager. This Module first communicates to the Name Node to create a namespace for the file when a user ready to store the file into HDFS. When the client gets acknowledged with the file Id from Name Node it then creates blocks for that file and calculate the Hash for that block and has to communicate to the Block Hash Manager to check whether that block is already stored in the HDFS data node by comparing the hash with the stored hash values. Based on the results return by the Block Hash Manager the client communicates to the Name node, if the return result is - existing block id and file id. Then it has to indicate the name node to refer that new node. If the result is null then client communicate to the Name Node to get the details of the data node to store the block. Then client communicates to the data node to store the blocks.

Hash Block Manager

Communicates with the Hbase Storage provider to check whether the hash of that block already exists. If the Hash value already presents in the HBase then the corresponding block Id and file Id will be returned and the reference will be created in the Namenode for the current block. Otherwise the file id, block id and hash value will be stored in the Hbase.

HBase

HBase is the columnar oriented database we can say that data model identical to the Google's big table. It allows for the arbitrary access to data, which is in huge amount. This provides a faster access to the data. Some other databases, which are similarly providing the random access to the data, are MongoDB, Cassandra, couchDB, Dynamo.

This Module used to store and retrieves the details to and from the HBase database.

VII. Design and Implementation

The Design and Implementation of the de-duplication system performs in the following way.

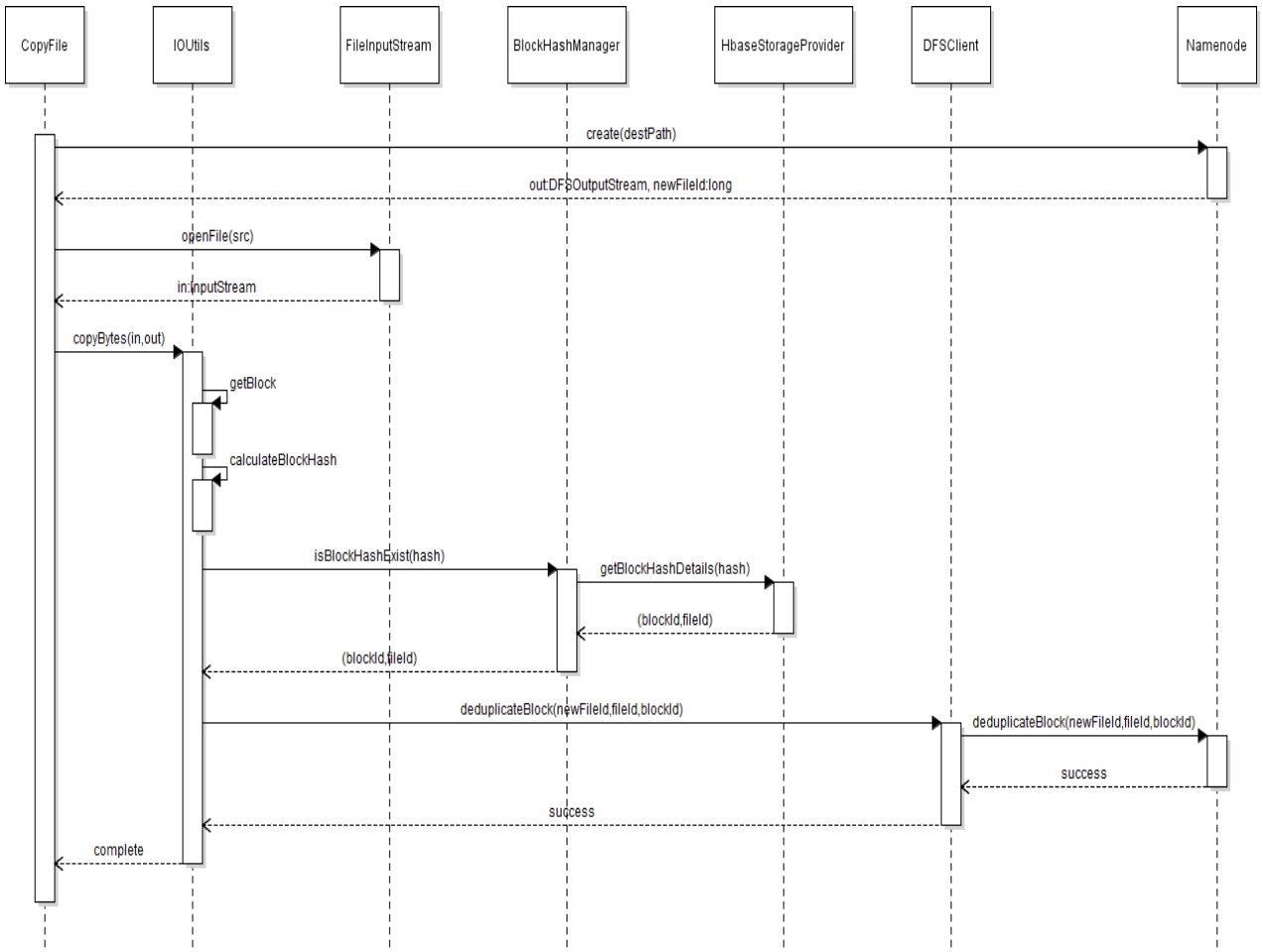

Fig. 3 Sequence diagram when block already exist

In Fig 3 When a client is ready to store the file in the hadoop file system, it first creates a namespace reference for the file in the name node. DFS output stream will be created and returned to write a file. Then source file, which has to be stored from source file system to HDFS) will be read using input stream and each block will be created based on the block size. Once the block is created Hash for that block will be generated. isBlockHashExist(hash) will be called on the Block Hash Manager and inturn it calls the getBlockHashDetails(hash) will be called to get the details of that generated hash. If that hash is present then details like existing document id and chunk or piece id will be returned. Then using this detail record id and chunk id, a reference to the existing one will be created for new block.

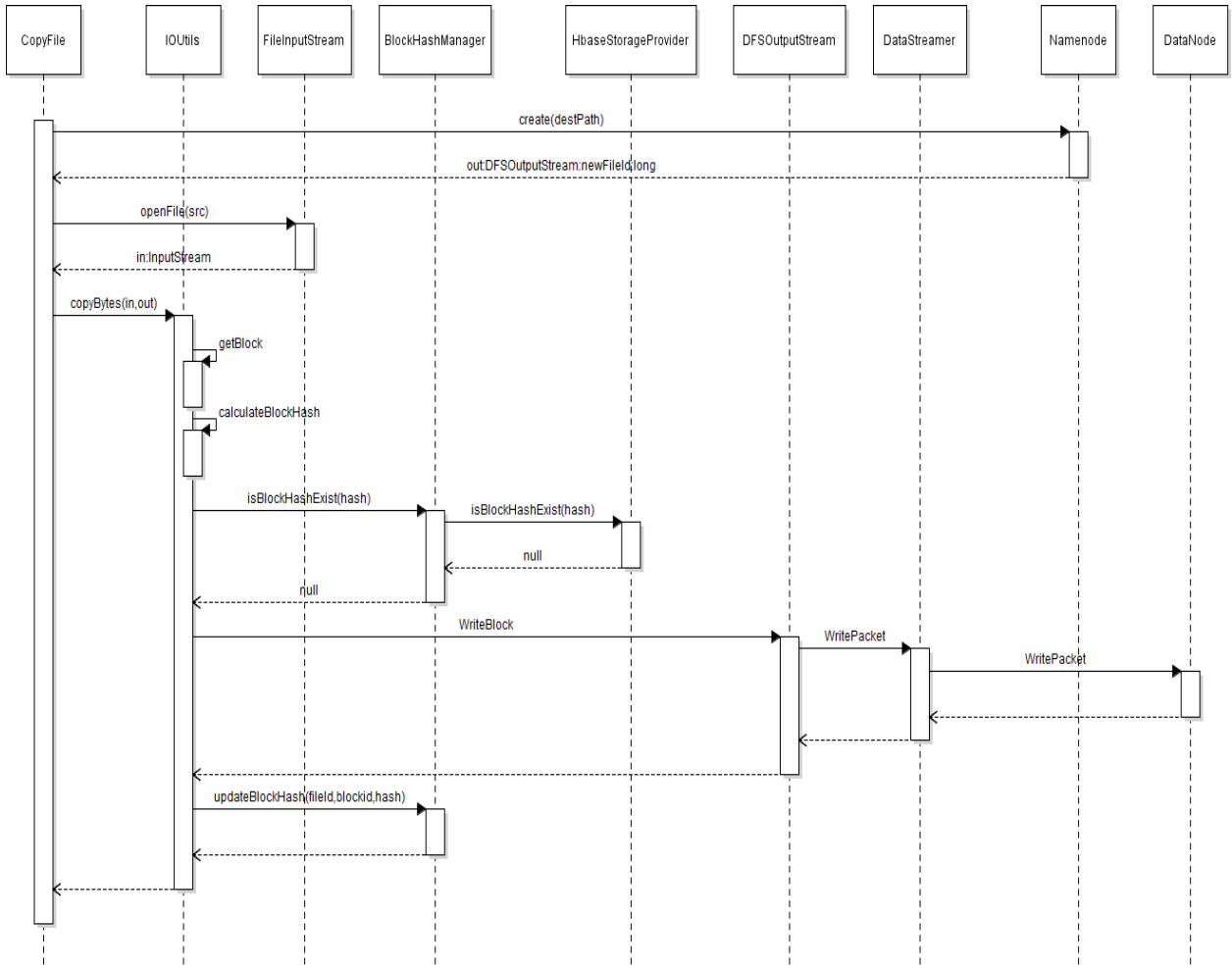

Fig. 4 Sequence diagram for Hash does not exist.

In Fig 4 When a file is ready to be copied it first creates a namespace reference for the file in the name node. DFS output stream will be created and returned to write a file. Then source file (which has to be copied from source file system to HDFS) will be read using input stream and each block will be created based on the block size. Once the block is created Hash for that block will be generated. isBlockHashExist(hash) will be called on the Block Hash Manager and in turn it calls the getBlockHashDetails(hash) will be called to get the details of that generated hash. If that hash is not present or not stored then the null will be returned saying no such block exists. Then the procedure of writing a block will be initiated by writing a block to DFS output stream Then DFS output stream divides that block into packets and stores those packets into queue. And Data Stream reads one by one packet from queue sequentially and writes to the data node.

VIII. Conclusion

The de-duplication framework based on HDFS is proposed in this project. The details of design and implementation of Block level de-duplication are shared and evaluated. This de-duplicated HDFS framework is very well suited for backup recovery, sync and share, Enterprise Content Management type of applications that store enormous amount of data is suitable for applications. The experiments and results demonstrate that the replicated information space can be spared. That is, the recommended framework indeed reduces the storage space consumption by removing redundant files for HDFS.

References

1. Kapil Bakshi, “Considerations for Big Data : Architecture and Approach”, in Aerospace Conference, 2012 IEEE conference.
2. B. Hong and D.D.E Long “Duplicate data elimination in SAN file system” – in Proceedings of the 21st IEEE Conference on Mass Storage Systems and Technologies, 2004.
3. Q.Sean and D.Sean. “Venti: A New Approach to Archival Data Storage”, USENIX Conference on File and Storage technologies.

