International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

Cloud Based Deduplication Using Middleware Approach

Sujit Tilak[1], Shrey Jakhmola[2], Arvind Hariharan Nair[3], Bhabya Mishra[4] , Isha Dwivedi[5]

[1]*Professor, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India*
[2]*B.E. Student, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India*
[3]*B.E. Student, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India*
[4]*B.E. Student, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India*
[5]*B.E. Student, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India*

Abstract – *Data Deduplication is an integral part of cloud storage. It efficiently reduces redundant data and thus reducing storage space. Data deduplication can be done in the server side or in the client side location. But, providing a middleware will be beneficial for the users, as they can have their preference for the storage solution. The benefit of implementing through client side is a reduction in the use of bandwidth and storage which would in turn results in user satisfaction, but it induces load on the user system, synchronization problem with the server and also increases difficulties in the technical aspect of the overall architecture. Data deduplication is done by splitting the file into small chunks in the server side and further generating a hash value which will be unique to the file content. The hash values are stored in the database. The hashing can be effectively performed by the SHA1 algorithm, since it has no noticeable collision thereby maintaining security. If the same hash value has been found in the storage server database then the file is not stored again but a pointer is created that points to the original file which will be used when a user requests for the file. Here, the entire deduplication process is performed in a middleware which connects the user and the server.*

Key Words: Cloud storage, deduplication, middleware, chunking, hashing, SHA1,

1.INTRODUCTION

Cloud storage is a cloud computing model that is used to store data online which can be accessed anytime by the user with active Internet connectivity. It can provide strong protection for our data and can be beneficial for cases like data backup and archival. It is also reliable during natural disasters, as the data can be stored in a remote server. The overall cost of using a cloud storage device is low as there is no need to purchase hardware; therefore reducing maintenance cost drastically compared to the traditional storage systems. Cloud not only store data. It is also known for being the shared pool of resources. In cloud computing, duplication of data is the main problem. Data duplication, as the name suggests is duplication of file while storing in the cloud. Removing the duplicate file from the storage space is termed as deduplication.

Data deduplication can be performed at various locations and levels using several methodologies. Deduplication can be either be implemented in a client side or server side location. But implementing in either side has its' advantages and disadvantages. Performing the process in the client side will increase the load on the client system. Software is required in the client side to proceed for deduplication. While on the server side, high bandwidth is utilized by the client. To overcome some of these issues in the client and server side, the middleware approach comes in to picture. The middleware acts as a bridge for the user and the storage. It allows the software to be isolated from the storage solution. It also helps reduce storage space. Instead, the user can rely on existing storage like Google Drive, Dropbox or any other user preference as this application is platform independent. Middleware is the place where the core deduplication procedure takes place that involves fixed size chunking and hashing using the SHA1 algorithm.

2. LITERATURE SURVEY

Many researchers have done commendable work on the subject. This section contains the overview of the work done by some of the researchers on deduplication.

1) Data Deduplication Cluster Based on Similarity-Locality Approach [2] .

Data deduplication can be done in file level, block level and byte level. The block level deduplication will reduce the expenses of the system and detection of segment within a file which is redundant can be detected. The similarity and locality concept is combined and used in this deduplication. The similarity deduplication take the similar kind of characteristics from backup stream. While on the other hand, locality deduplication means the order of the backup stream chunks will remain same in other high probability backup streams. The file is segmented and distributed to several nodes. These incoming segments are checked with respect to the local finger print. The Bloom filters are used to store these fingerprints. The incoming segments are identified by using MD5 or SHA1 algorithm by the fingerprints. If the fingerprint is similar then the segment isn't stored. But if it's not, then based on the degree of similarity comparison is done with the fingerprint summary or corresponding nodes.

2) A Hybrid Cloud Approach for Secure Authorized Deduplication [3] .

Hybrid Cloud Approach is a combination of both private and public cloud. The private cloud is responsible here for the sharing the file token to the user on request and providing with the private key. Through this way the private cloud does the authentication of the user so that only authorized user can check for deduplication of data, upload or download the file. Once the user gets hold of the token, it then shares it with the public cloud for upload or download purpose. The file token is received by the public cloud and results are given accordingly as per the request.

3) Try Managing Your Deduplication Fine-grained-ly: A Multi-tiered and Dynamic SLA-driven Deduplication Framework for Primary Storage [4].

The primary storage system is made up by two main modules: proxy server and object storage cluster (OSC). The client side requests are sent to the OSC storage node by the proxy server. The newly entered files meta data is present in the buffer zone present in the proxy server. OSC is a distributed storage cluster that contains approximately hundreds of storage nodes made up of various chunks. These storage nodes maintain the fingerprint in the index table for all the chunks. Fingerprint is nothing but hash of the content that is done by using SHA1 technique. Deduplication doesn't interfere with the Mudder on going operations, instead it performs the task in the background itslef. DCL-Module, SLA-Handler, Deduplication Executor(Deduper) and Workload Monitor (Monitor) are the main modules of the Mudder. When new files meta data are filles in the buffer zone and the entries reach a threshold value N, DCL module fetches all these entries and generate a Target list. DCL module is operable inside the proxy server. Initially the entry in the list is predefined as dirty, meaning not duplicate. The DCL module then calculated for deduplication. DCL-Module pushes the list to Deduper which is a channel where deduplication between storage nodes and proxy takes place. Procedure of removing redundant data and index update is similar to other systems.

4) Deduplication in cloud storage using Hashing technique for encrypted data [6].

The data to be sent for uploading purpose is encrypted first before sending it to the Cloud Service Provider. The encryption is done using the AES 128 bit encryption that is protected from various attacks, one being Birthday brute force attack. The data is encrypted in the data owner machine by using a secure key. In this way the Cloud Service Provider doesn't have access to the plain text of the file to be uploaded because it receives a encrypted copy of the file. The encrypted file received undergoes chunking and hashing. The hashing is performed using the MD5 algorithm. If the hash value of the data is same as that already present in the cloud, then the data isn't stored again. If the hash isn't present in the cloud then the file is stored in the cloud and the location is updated in theindex table of Cloud Service Provider.

5) Improving the Performance of System in Cloud By Using Selective Deduplication [7].

The user of the system can request for upload. This upload takes place from LAN or WAN or MAN to the cloud. It depend on the users whether they want to check the deduplication of the data or not. The selective deduplication is applied on the data to find the duplicate data if any. If any duplicate data is found then the feedback is sent to user and the redundant data is deleted from the cloud storage. If in case, the data is unique, that is, not redundant, then user is asked for encryption. Security key for the file is generated and encryption is performend on the file. After encryption, the file is stored in the cloud storage. When the user requires, decryption can be performed on the file and thus a key is generated for decrypting file from cloud.

6) PFP: Improving the Reliability of Deduplication-based Storage Systems with Per-File Parity [8].

During the process of deduplication, the reliability gets affected because of the removal of redundant chunks. These identified duplicate chunks are removed because they are common to other files too. At the end, there is just one chunk that is shared by all the files after the deduplication. This removal of chunks renders the sharing files unavailable. This paper gave an idea of Per File Parity(PFP) to improve the reliability. PFP performs the XOR parity within the parity groups of data chunks which are part of each file after the chunks are created but before these chunks are deduplicated. PFP performs XOR parity within the data chunks parity group. Therefore, PFP provides security from the parity redundancy. The PFP doesn't hinder the flow of data duplication.The fingerprint calculation, chunking, detection of duplicate chunks and removal of these duplicate chunks, nothing changes.Parity generation is implemented on every file to prevent deduplication of the data. The resulting parity metadata is stored as the metadata of file. The incoming file data chunks are created and the fingerprints are checked. PFP then divides these chunks into groups as per predefined group size of N. For N data chunk group the parity is calculated by using XOR operation. This scheme can tolerate a single chunk failures.

3. PROPOSED SYSTEM

The proposed system comprises of components like the Client, Middleware and the Server. Middleware is the core component where all the processing takes place.

Fig -1: System Architecture

A middleware is a software that acts as a bridge between two endpoints, generally a client and a server. The middleware gives the users an interface to upload the files. The user has to create an account on the application in order to perform functions like upload or download. Once the user has an account, they can add files to their storage using the Middleware. The Middleware will be handling the process of data deduplication.

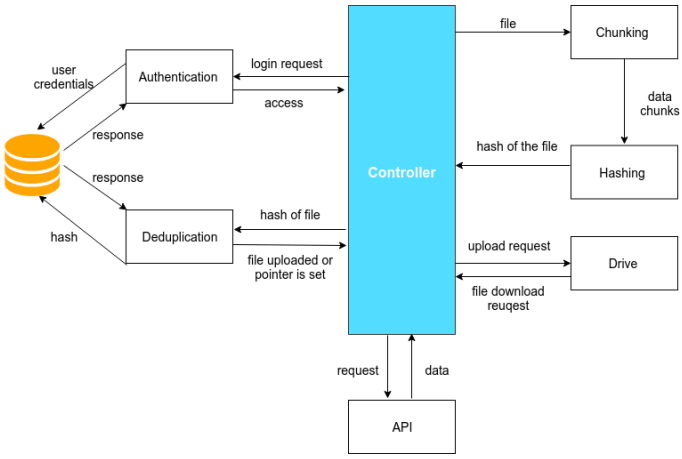

Fig -2: Middleware and associated modules

The middleware consists of the the following modules:

1. Authentication Module : It is used to authenticate the user so that the user can access the middleware dashboard. The user needs to supply their credentials in the login form to get access to their account. The authentication module also provided a service to sign up for the account of the middleware.

2. Chunking Module: The chunking module is invoked by an API call which chunks the file into small fixed size pieces in a synchronized manner which is then passed on to the hashing module.

3. Hashing Module: The hashing module takes the chunked file input from the chunking module and generates SHA1 hash of each chunk and stores it in an array until the the chunks of the current file being processed is completely supplied to the module. After the individual hashing of the chunks, the SHA1 of the whole array which consists of hash of the chunk is hashed and returned to the API.

4. Drive Module: The drive module is used for uploading the user file to the user's drive storage. It is also used by the cron job process to sync drive storage content information with the middleware by using the metadata of the files present on the cloud of the user and also to download the file from users cloud for deduplication processing.

5. API: The API Module is used by other modules to interact to other middleware or to pass the data throughout the entire middleware. It handles the incoming requests and data response for other modules using HTTP requests. The data is wrapped on the response body.

6. Cron Module: The cron module runs automatically everyday once to sync with the users cloud storage. The job of the cron module is to check whether the user has uploaded any file directly to the cloud storage by fetching the file's metadata and checking in the database. If the file has never been processed by the middleware, then it fetches the file from the cloud storage using the API and drive module for the process of deduplication.

The processing of the file can take place based on the two scenarios with respect to the method of file upload. In the first scenario, the user uploads the file to the middleware via the uploader. In the second scenario, if the user uploads the file directly to the cloud storage or for the old files on cloud storage need to go through the process of deduplication which is done using the cron job module.

The file which is uploaded to the middleware or downloaded from the user's cloud storage will go through chunking and hashing. Based on the value of hash generated it is determined whether duplication is present or not by checking in the database with respect to that individual user. If the hash of the file exists in the database then a pointer is added to the location where the original file exists and the file at hand is discarded. But, if the hash is not present in the database then the current file information along with its hash value is added in the database. While syncing with the cloud storage, using file ID from the metadata of the file, we check whether the file has been ever processed or not. If not processed then it goes through the above process otherwise it is sent for batch processing.

3.1. Techniques Used

The system focuses on removal of the duplicate data once the file is uploaded by the user. The major techniques used in the system are chunking and hashing that is performed in the middleware. The chunking and hashing are used for the deduplication process. The deduplication is done in order to remove the redundancy. If the user uploads a file, and the same is present in the storage server then a pointer is assigned to the original file instead of saving the new file again. Deduplication reduces use of storage space.

This is performed in two steps:

1. Chunking: This is a technique that breaks a large piece of information say file into multiple pieces or chunks. These chunks are helpful for further data deduplication. Chunking can be done by either using fixed size chunking or file level chunking.

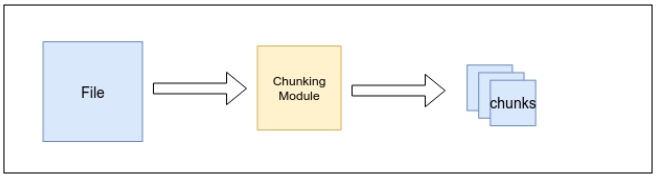

Fig -3: Chunking Process

2. Hashing: This is a method used to generate a value of the incoming data of a file through a mathematical function. This is performed during the deduplication procedure in the middleware. The hash values can be generated by either using SHA1 or MD5 algorithm. SHA1 is preferred over MD5 as the collision experience is less. Also, SHA1 is faster with respect to it's successors.

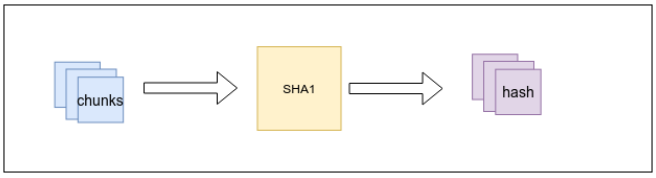

Fig -4: Hashing Process

The chunks created from the chunking process are hashed and pushed into an array. Finally after all the chunks hash are pushed into an array, the arrays' hash is generated that is used for deduplication. The middleware runs a service which is used to sync the data already present on the user's cloud storage. The sync checks whether the file present on the cloud storage is processed by the middleware or not using the fileID from metadata. The file is fetched and processed from the cloud server and same deduplication process is done if the file is not processed ever otherwise the files are sent for batch processing which is optional or not required if the cloud is only used for storage and not for editing files.

3.2 Proposed System Procedure Flow

The user's file upload command is examined by the middleware. The middleware fetches the file from the user for chunking and hashing. The chunking process first takes place which gives small chunks of the file whose hash is generated. The number of chunks depend on the size of the file with respect to the chunk size. One by one the hash of the chunks are generated and pushed into an array. Once all the chunks of the file are created, the hashing of the complete array is generated. This generated hash of the file is now compared with the hash values present in the database.

As shown in Fig 5, there are two scenarios as stated below:

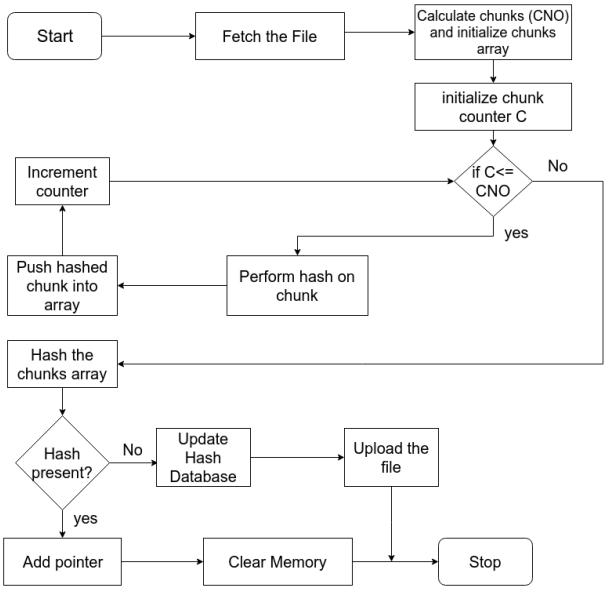

Fig -5: Procedural Flow

Case I: Hash generated of the file is not present in the database. In this case the database is updated along with the new hash value. The file in this case is uploaded in the storage server.

Case II: Hash generated of the file is present in the database. In this case the file is not saved again. Rather a pointer is added that will point to the original file present in the storage server.

3.3 Use Case Diagram

There are three actors involved in the system as shown in Fig 6. Those can be described as:

1. User: The user is the one who will use the deduplication application.

2. Middleware: The middleware is the system which will check for deduplication and ensure that there is no duplicate file in the storage server.

3. Storage: Storage is a system where the file of the users is stored and retrieved.

The system validates the user credentials and logs in them into the system. The middleware fetches the file from the storage server using the appropriate credentials. The middleware uploads the processed file to the storage server using appropriate credentials. The user uploads the file to the middleware using the dashboard. The middleware create chunks of the file which was uploaded by the user. The chunks created will be hashed and stored in a hash digest. The middleware will now store the hash generated to the database.

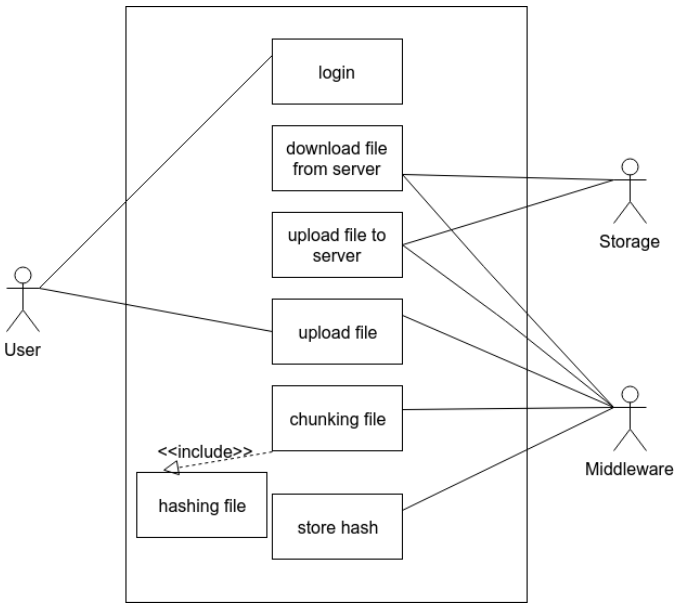

Fig -6: Use Case Diagram

4. PERFORMANCE EVALUATION

4.1 Datasets Used

The datasets used in this section are iso files. An ISO file is a disc image of an optical disc and contains everything that can be written to the optical drive. In earlier days ISO was the format used to burn CDs and DVDs whereas, in the present days an ISO file is used to transfer a larger amount of files compressed into a single ISO file which is then shared over the internet using peer to peer or FTP servers. The name ISO is taken from the ISO 9660 file system. The datasets taken are ISO image of Linux distributions that are most widely used. These datasets are relatively large, usually, the size ranges in Gigabytes. The large dataset will ensure harmonious and distinct results between fixed size hashing and file level hashing.

Table -1: Datasets Involved

Datasets	Dataset Name	Type	Provided By	Size (GiB)
Dataset 1	Antergos-18.12-x86_64.is	ISO File	Antergos.com	2.1
Dataset 2	openSUSE-Tubleweed-DVD-x86_64Snapshot20181128-Media.iso	ISO File	Suse Linux	4.1
Dataset 3	Popos_18.10_amd64_intel_11.iso	ISO File	System 76 inc.	2
Dataset 4	Ubuntu-18.04.1-desktop-amd64.iso	ISO File	Canonical inc.	1.8

4.2 Tools Involved

The performance analysis done on the proposed system involves the use of following tools:

1. Time command linux: The time command runs the specified program with the provided arguments. When command finishes the execution time writes back a message to standard error giving timing and processing statistics about the program that was runned.

2. htop for cpu usage: Htop is an interactive system viewer and process managing tool. It is based on UNIX tool top. Htop allows the user to view process or tasks statistics like CPU usage, ram consumption, and the process uptime.

3. console.time() and console.timeEnd() for nodejs function: These inbuilt methods are used to benchmark javascript applications for response and average computation time required to complete the given task. console.time(label) is used to set the initial reference from where the timer would begin. console.timeEnd(label) is used as the closing statement the tasks in between console.time() and console.timeEnd is bench marked for time.

4.3. Evaluation Metrics

The metrics used for benchmarking the systems performance with respect to the defined constraints is the Evaluation metrics.

1. Hashing Time: The Hashing Time is the time required (in Seconds) to produce the hash of a given input. The hash function encodes each byte as two hexadecimal characters. The hash time should be lower in order to increase the efficiency of the application. Lower hash time also helps in scaling the application if needed.

Table -2: Hashing Time in seconds

Datasets	File Level	Fixed Size
Dataset 1	67.3	24.19
Dataset 2	132.66	46.1
Dataset 3	82.59	22.6
Dataset 4	34.5	25

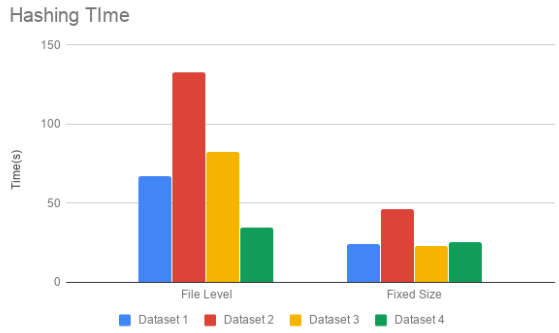

Fig -7. Time consumption graph for hashing

2. Chunking Time: The time required to divide a file into chunks is termed as chunking time . It is usually measured in milliseconds (ms). The lesser the chunking time, the faster the file data will be processed and analyzed. It obtained results won't be better than file level processing, as the chunking time increase.

Table -3: Chunking Time in milliseconds

Datasets	Fixed Size
Dataset 1	1.7
Dataset 2	2.47
Dataset 3	2.93
Dataset 4	2.41

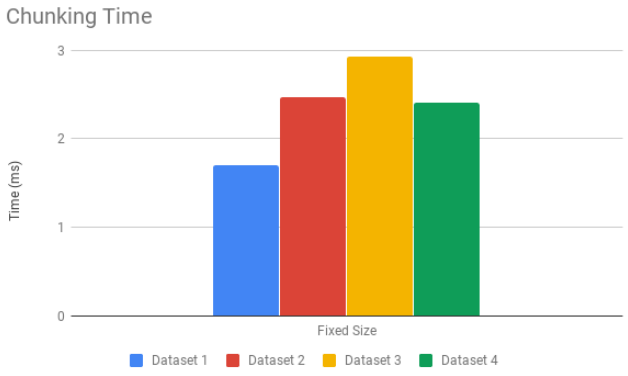

Fig -8. Time consumption graph for chunking

3. CPU load: CPU load measures the amount of computational work in percentage that the system performs in order chunk and hash the input data. The load on the CPU should be neutral and consistent. The more the load on the CPU the more the chances of the application getting halt. Also, less CPU load ensures less cost on dedicated hardware and increased scalability of the application.

Table -4: CPU Load in Percentage

Datasets	File Level	Fixed Size
Dataset 1	40	23
Dataset 2	66	40
Dataset 3	38	37
Dataset 4	40	33

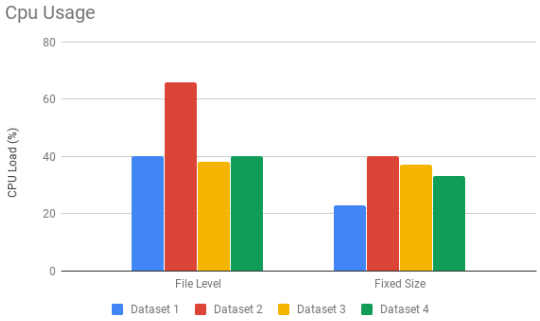

Fig -9: Tme consumption graph for CPU

5. CONCLUSION

In effect of all the results that were obtained in our proposed system, the evaluation of the performance of our project, as well as the extensive Literature survey conducted by us in this project it can be concluded that the the middleware approach for performing the deduplication is more effiecient and cost effective. The workload on the client and server side is reduced. The performance evaluation reveals that chunking is the most efficient form of file processing and hence it must be used in data dedupication. The hashing is performed using SHA1 algorithm. We discovered that, SHA1 is significantly more secure than MD5 as it is prone to less collisions and it is significantly faster than SHA256 and SHA512. Hence, it provides the best balance between security and performance.

REFERENCES

[1] W. Litwin, Thomas Schwarz,"Combining Chunk Boundary and Chunk Signature Calculations for Deduplication ", IEEE, 2012.

[2] Jian Zhang, Xingyu Zhang, "Data Deduplication Cluster Based on Similarity-Locality Approach",IEEE, 2013.

[3] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication", IEEE, 2014.

[4] Yan Tang, Jianwei Yin, and Zhaohui Wu, "Try Managing Your Deduplication Fine-grained-ly: A Multi-tiered and Dynamic SLA-driven Deduplication Framework for Primary Storage", IEEE, 2016.

[5] Nancy Digra1 , Sandeep Sharma, "A Noval Approach of Enhancing Security in Cloud Using Diffie Hellman Algorithm",IJSRM, 2017.

[6] Vaishnavi Moorthy, Arpit Parwal and Udit Rout,"Deduplication in Cloud Storage Using Hashing Technique For Encrypted Data",IEEE, 2018.

[7] Mr. Nishant N.Pachpor, Dr. Prakash S.Prasad, "Improving the Performance of System in Cloud by Using Selective Deduplication", IEEE, 2018.

[8] Suzhen Wu, Bo Mao, Hong Jiang, Huagao Luan, Jindong Zhou,"PFP: Improving the Reliability of Deduplication-based Storage Systems with Per-File Parity",IEEE, 2019.