Enhanced Intensive Indexing (I2D) De-Duplication for Space Optimization in Private Cloud Storage Backup

M. Shyamala Devi and Steven S. Fernandez

***Abstract*—Cloud Storage provide users with abundant storage space and make user friendly for immediate acquiring of data, which is the foundation of all kinds of cloud applications. However, there is a lack of deep studies on how to optimize cloud storage aiming at improvement of data access performance. With the development of storage and computer technology, digital data has occupied more and more space. According to statistics, 60% of these digital data is redundant, and the traditional data compression can only eliminate the intra-file redundancy. The growth in redundant data will continue, unabated. The issue is how to manage this phenomenon, while operating with the assumption that the growth will likely accelerate. In order to solve these problems, Data De-Duplication has been proposed. Many organizations have set up private clouds for best resource utilization. An organization can built private cloud storage with their unused resources for storing their personal data. Since private cloud storage has a limited amount of hardware resources, they need to optimally utilize the space to accommodate maximum data. Data De-Duplication is an effective technique to optimize the utilization of storage space backup by avoiding the redundancy. In this paper, we are going to discuss the flaws in the existing de-duplication methods and introduce new methods for Data De-Duplication. Our proposed method namely Intensive Indexing (I2D) De-duplication which is the enhanced File level de-duplication that provides dynamic space optimization in private cloud storage backup as well as increase the throughput and de-duplication efficiency.**

***Index Terms*—Cloud backup, cloud computing, constant-size chunking, data de-duplication, full-file chunking, private storage cloud, redundancy.**

I. Introduction

Cloud computing delivers flexible applications, web services and IT infrastructure as a service over the internet using utility pricing model. The Cloud is a cost-effective approach to technology as there is no need to make usage predictions, upfront capital investments or over purchase hardware or software to meet the demands of peak periods. Cloud computing incorporates virtualization, data and application on-demand deployment, internet delivery of services, and open source software. The different forms of cloud design are Public cloud, Private cloud and Hybrid cloud. **Public clouds** are run by third party service providers and applications from different customers are likely to be mixed together on the cloud's servers, storage systems, and networks. Here the computing infrastructure is hosted by the cloud vendor at the vendor's premises. **Private clouds** are built for the exclusive use of one client. Private clouds can also be built and managed by the organization's own administrator. Here the computing infrastructure is dedicated to a particular organization and not shared with other organizations. Private clouds are more secure when compared to public clouds. **Hybrid clouds** combine both public and private cloud models which may be used to handle planned workload spikes, or storage clouds configuration.

A. Cloud Storage

Cloud storage is a service model in which data is maintained, managed and backed up remotely and made available to users over a network. Cloud storage [1] provides users with storage space and make user friendly and timely acquire data, which is foundation of all kinds of cloud applications. There are many companies providing free online storage. The storage cloud provides Storage-as-a-Service. The organization providing storage cloud uses online interface to upload or download files from a user's desktop to the servers on the cloud. Typical usage of these sites is to take a backup of files and data. Storage cloud exists for all the types of cloud. A cloud storage SLA is a service-level agreement between a cloud storage service provider and a client that specifies details of the service, usually in quantifiable terms.

B. Advantages of Cloud Storage

Cloud storage has several advantages over traditional data storage. For example, if we store our data on a cloud storage system, we will be able to get that data from any location that has internet access. There is no need to carry around a physical storage device or use the same computer to save and retrieve our information. Thus cloud storage is convenient and offers more flexibility.

C. Private Cloud Storage

Public cloud storage such as Amazon's Simple Storage Service (S3) [2], provide a multi-tenant storage environment that is most suitable for unstructured data. **Private cloud storage** services provide a dedicated environment protected behind an organization's firewall. Private clouds are appropriate for a user who need customization and more control over their data and is shown in Fig. 1. **Hybrid cloud storage** is a combination of at least one private cloud and one public cloud infrastructure. An organization store actively used and structured data in private cloud and unstructured and archival data in a public cloud.

D. Private Cloud Storage Backup

Cloud storage backup [1] is a strategy for backing up data that involves removing data offsite to a managed service provider for protection. Data moved offsite should be

Manuscript received March 19, 2014; revised May 14, 2014.

M. Shyamala Devi and Steven S. Fernandez are with R.M.D Engineering College, Chennai, India (e-mail: shyamalapmr@gmail.com).

de-duplicated to avoid the redundancy and it is done by Cloud Storage Controller (CSC). The three benefits of CSC are as follows. First, it creates a seamless and highly robust connection to cloud storage, while requiring no changes to applications running in the data center. Applications are able to access the cloud using standard block and file access protocols. Second, it accelerates the performance of applications using cloud storage through advanced WAN techniques including caching, de-duplication, compression, and protocol optimization. Third, the Cloud Storage Controller provides the same features and capabilities expected of local storage arrays

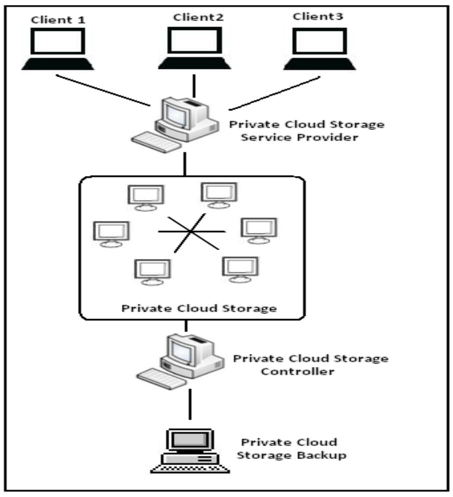

Client 1
Client2
Client3
Private Cloud Storage Service Provider
Private Cloud Storage
Private Cloud Storage Controller
Private Cloud Storage Backup

Fig. 1. Private cloud storage.

E. *Overview of De-Duplication*

Data De-duplication identifies the duplicate data to remove the redundancies and reduces the overall capacity of data transferred and stored. De-duplication often called as "intelligent compression" or "single-instance storage"[3] which is the 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. For example, if an organization webmail system might contain 50 instances of the same one megabyte (MB) file attachment. If the webmail platform is backed up or archived, all 50 instances are saved, requiring 50 MB storage space. With data de-duplication, only one instance of the attachment is actually stored. Each subsequent instance is just referenced back to the one saved copy. In this example, a 50 MB storage demand could be reduced to only one MB. Data de-duplication offers three benefits. First, lower storage space requirements will save money on disk expenditures. Second, efficient use of disk space also allows for longer disk retention periods and reduces the need for tape backups. Third, it also reduces the data that must be sent across a WAN

F. *De-Duplication Techniques*

The optimization of backup storage technique is shown in Fig. 2. The Data de-duplication [4]-[6] can operate at the whole file, block (Chunk), and bit level.

Whole file de-duplication or Single Instance Storage (SIS) finds the hash value for the entire file which is the file index. If the new incoming file matches with the file index, then it is regarded as duplicate and it is made pointer to existing file index. **Block De-duplication** [6], [7] divides the files into fixed-size block or variable-size blocks. For **Fixed-size chunking**, a file is partitioned into fixed size chunks for example each block with 8KB or 16KB. In **Variable size chunking**, a file is partitioned into chunks of different size. Both the fixed size and variable size chunking creates unique ID for each block using a hash algorithm such as MD5 or SHA-1 [8] or MD5 [9]. The unique ID is then compared with a central index. If the ID exists, then that data block has been processed and stored before. Therefore, only a pointer to the previously stored data needs to be saved. If the ID is new, then the block is unique. The unique ID is added to the index and the unique chunk is stored. Block and **Bit de-duplication** looks within a file and saves unique iterations of each block or bit.

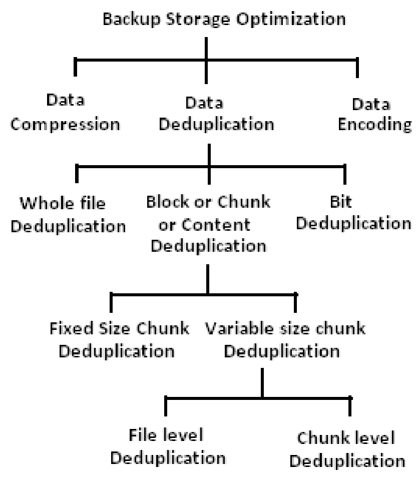

Backup Storage Optimization
Data Compression
Data Deduplication
Data Encoding
Whole file Deduplication
Block or Chunk or Content Deduplication
Bit Deduplication
Fixed Size Chunk Deduplication
Variable size chunk Deduplication
File level Deduplication
Chunk level Deduplication

Fig. 2. De-duplication methods.

The rest of the paper is organized as follows. In Section II, we analyze the existing methods of de-duplication with its advantages and disadvantages. In Section III, we discuss about our proposed system and its functions. In Section IV, we conclude our design of DWFD and prove that our scheme greatly increases the de-duplication efficiency. We show our implementation analysis in Section V.

II. Analysis of Existing Methods

In this section, we describe the advantages and disadvantages of each de-duplication method.

A. *Advantages of Existing Methods*

1) Indexes for whole file de-duplication are significantly smaller, which takes less computational time and space when duplicates are being determined. Backup performance is less affected by the de-duplication process.
2) Fixed-size chunking is conceptually simple and fast since it requires less processing power due to the smaller index and reduced number of comparisons.
3) In variable size chunking, the impact on the systems performing the inspection and recovery time is less. The efficiency of identifying the duplicate is high.
4) Bit De-duplication done exact de-duplication and it is more efficient since it eliminates redundancy.

B. Disadvantages of Existing Methods

1) Whole File de-duplication is not a very efficient, because a little change within the file causes the whole file to be saved again. For example, if 500 identical attachments are sent by a insurance coordinator, this method will find all those 500 attachments that are exactly the same, but it would not find the exact duplicate copies that we have saved (i.e) Insure.Aug,Insure.Sep,Insure.Oct etc. This de-duplication checks only the size of the file regardless of the file content.
2) In Fixed-size chunking, when a small amount of data is inserted into a file or deleted from a file, an entirely different set of chunks is generated from the updated file.
3) The indexes for both fixed and variable size chunking are large which leads to larger index table and more number of comparisons which leads to low throughput.
4) Bit de-duplication takes more processing time to identify the duplicate bit.

C. Methods of Block Level De-Duplication.

The block level de-duplication divides the incoming file into fixed size chunks or variable size chunks. Depending on the duplicate detection of incoming chunk, the variable size chunk de-duplication can be divided into **Chunk level de-duplication** and **File level de-duplication.**

D. Chunk Level De-Duplication – DDDFS

When a file has to be written, then every chunk of that file is checked for duplicate with chunks of all files. This method of detecting duplicates is **Chunk level de-duplication**. Data Domain De-duplication File System [10] DDDFS is a file system which performs chunk level de-duplication. It supports multiple access protocols. Whenever a file to be stored, it is managed by the interfaces such as Network File System (NFS), Common Internet File System (CIFS) or Virtual Tape Library (VTL) to a generic file service layer. **File service layer** manages the file metadata using **Namespace index** and forwards the file to the content store. **Content store** divides the file into variable sized chunks. Secure Hash Algorithm SHA-1 [8] or MD5 [9] finds the hash value for each variable size chunk, which is ChunkID. Content store maintains the **File Reference Index** (FRI) which contains the sequence of ChunkID constituting that file. **Chunk store** maintains a chunk index for duplicate chunk detection. In this chunk level de-duplication, the efficiency of duplicate detection is high but the throughput of the de-duplication is low. So this method can be used for the applications with locality of reference between the data streams in the cloud storage.

E. File level De-Duplication – Extreme Binning

When a file has to be written, then every chunk of that file is checked for duplicate with all the chunks of the similar files. This method of detecting duplicates is **File level de-duplication**. Extreme Binning [7] uses this approach by dividing the chunk index into two tiers namely Primary index and Bin [11], [12]. ***Primary Index*** contains the representative ChunkID, Whole file hash and pointer to bin. The disk contains **bin**, Data chunks and the File recipes. The file recipes [13], [14] contain the sequence of chunked for that file. Fig. 3 shows the structure of a backup node in extreme binning de-duplication. When a file has to be backed up, it performs variable size chunking and finds the representative ChunkID and the hash value for the entire file [15], [16]. The Representative ChunkID is checked in the primary index and if it is not there, then the incoming file is new one and a new bin is created with all ChunkID, chunksize and a pointer to the actual chunks are added to the disk. Then Representative ChunkID, file hash value and the pointer to bin of a newly created bin are added to the primary index [17]. If the representative ChunkID, file hash of the incoming file is already present in the primary index, then the file is a duplicate [18] and it is not loaded into disk and the bin is not updated. If the representative ChunkID of the incoming file is already present in the primary index [19] but the hash value of the whole file does not match, then the incoming file is considered to be nearly similar to the one that is already on the disk. Most of the chunks of this file will be available in the disk. The corresponding bin is loaded to RAM from the disk, and now searches for the matching chunks of the incoming file. If the ChunkID is not found in the bin, then its metadata of the chunk is added to the bin and the corresponding chunk is written to the disk. The whole file hash value is not modified in the primary index and the updated bin is written back to the disk. Here every incoming chunk is checked only against the indices of similar files, this approach achieves better throughput compared to the chunk level de-duplication. Since non-traditional backup workload demands better de-duplication throughput, file level de-duplication approach is more suited in this case.

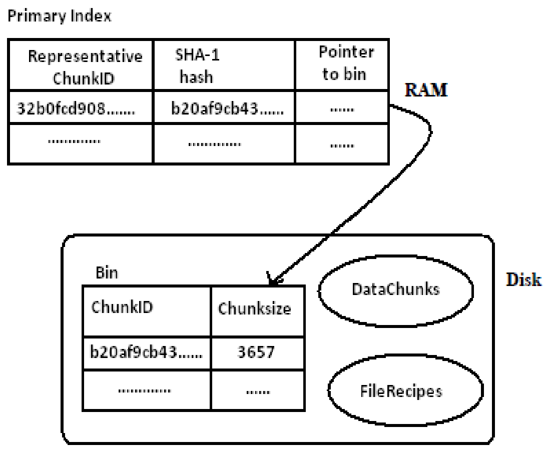


Fig. 3. Backup node in extreme binning.

III. Our Contribution

Cloud computing is used for better utilization of available resources [20]. The unused materials of an organization can be used to build up a private cloud. Here we try to optimize the private cloud storage backup in order to provide high throughput to the users of the organization by increasing the de-duplication efficiency.

A. Proposed System

Generally the backup of the private storage cloud belongs to the non-traditional backup. Traditional backup contains data streams with locality of reference. But the non-traditional backup contains the individual files that owns by the individual users of the organization with no locality of reference. The storage of the private cloud should be optimized as there is physical limitation on the storage space. Here we try to enhance the File level de-duplication since it provides high de-duplication throughput. However a single primary index is used for de-duplication that takes more time in merely checking the representative ChunkID of the file. This leads to low de-duplication throughput. So we try to refine file level de-duplication further to increase the throughput and de-duplication efficiency. So we propose a new method for de-duplication namely Intensive Indexing (I2D) File De-duplication which is the modified File Level de-duplication that provides grouping of files of individual users.

B. *Intensive Indexing (I2D) File De-Duplication*

The single Primary index contains representative ChunkID, whole file hash and bin pointer which points to the bin of the backup node which is used for finding the duplication regardless of the users of the private cloud which leads to low de-duplication throughput. Private storage cloud consists of personal documents of the individual users belonging to organization. If we use Extreme Binning, then there will be only one primary index for all user files. So all the incoming files that belong to the different user's merely waste time for checking the representative chunkID of the single primary index that reduce the throughput and de-duplication efficiency. In our Intensive Indexing (I2D) File De- duplication, the users accessing the private cloud storage are identified by their unique user-id. Here the chunk index is divided into File Index and Bin. We create separate file index and bin for each user and each file belonging to an individual user is grouped with their folders and is shown in Fig. 4. With this method, it is possible to group the files of each users of the organization.

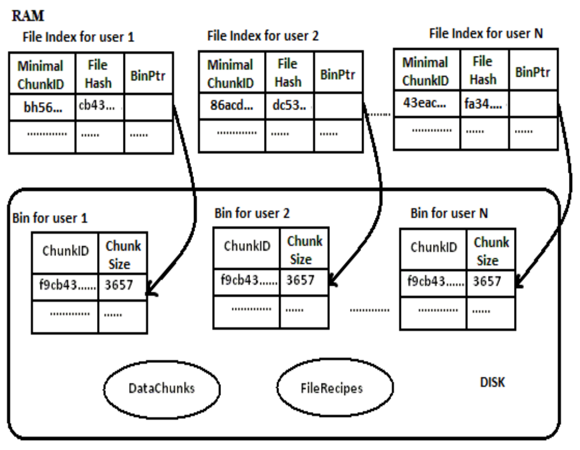


Fig. 4. Intensive indexing file de-duplication.

IV. Design of Intensive Indexing (I2D) File De-Duplication

Before we start our design, we have the following assumptions:

i) The users of the private cloud are provided with separate user id. ii) The files of the individual users are collected in separate folders in the cloud backup. Our new Intensive Indexing (I2D) File De-duplication scheme has the following modules,

1) Cloud Service Providing Module
2) Cloud Storage Initiation Module
3) Cloud Storage Controller Module
4) Intensive Indexing Module.
5) Cloud Backup De-duplication Module.

A. *Cloud Service Providing Module*

The user authentication is done in this module. If the user is new, then the registration process is done in this module and is shown in Fig. 5.

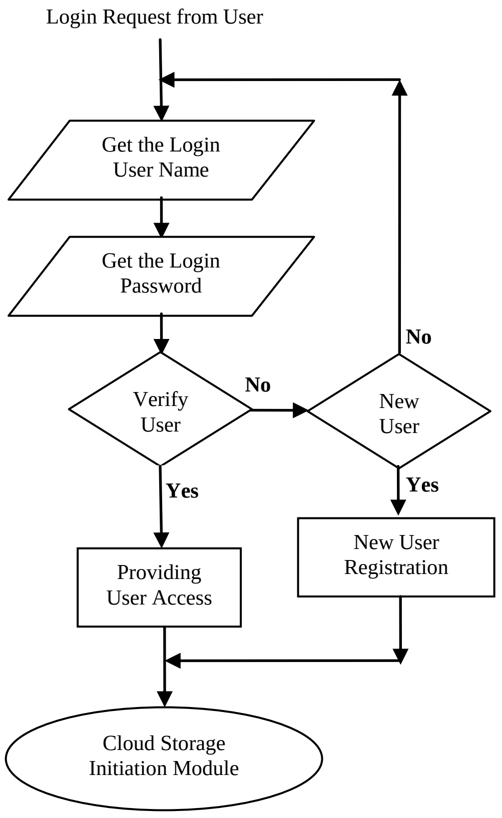


Fig. 5. Cloud service providing module.

B. *Cloud Storage Initiation Module*

After the user authentication is done in the private cloud, then he / she can start viewing, editing and saving their personal files into their folders and it is shown in Fig. 6. In this module, the authenticated user can perform their own work and they may also try to upload the files from online.

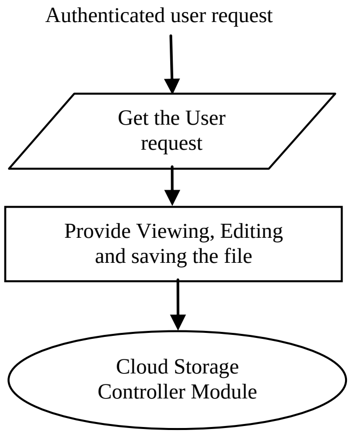

Authenticated user request
Get the User request
Provide Viewing, Editing and saving the file
Cloud Storage Controller Module

Fig. 6. Cloud storage initiation module.

C. Cloud Storage Controller Module (CSCM)

This module performs the function of integrating the files of the individual users. This module groups the files of all users and is shown in Fig. 7.

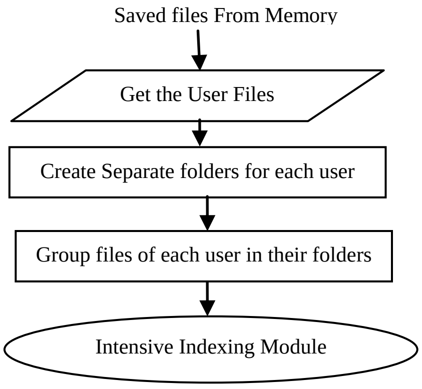

Saved files From Memory
Get the User Files
Create Separate folders for each user
Group files of each user in their folders
Intensive Indexing Module

Fig. 7. Cloud Storage controller module.

D. Intensive Indexing Module

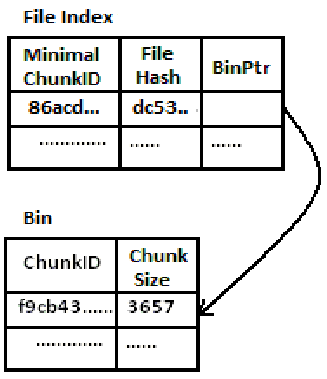


Fig. 8. File Index in I2D.

The chunk index for each user is created in this module. The chunk index is divided into file index and bin and it is shown in Fig. 8. The chunk index for each user is created in this module and it is shown in Fig. 9. The file index has three field's namely minimal chunkID, File hash and Binptr. The minimal chunkID is found for all the files. First the file is divided into variable sized chunks. Then the minimal chunkID is found using Broder's theorem [21]. The hash value is found for all the different sized chunks. The ID with minimum hash value is choosen to be the minimal ChunkID for the file index. The purpose of finding the minimal ChunkID is that according to Broder's theorem, the probability that the two sets S1 and S2 have the same minimum hash element is the same as their Jaccard similarity coefficient [22]. In other words, if two files are highly similar they share many chunks and hence their minimum chunk ID is the same with high probability. The file hash is found by SHA-1 [8] or MD5 [9]. The Binptr provides pointer to the corresponding bin. Each bin contains two fields as chunkID and the chunksize. The hash value of the chunk is found and it named as ChunkID

E. Cloud Backup De-Duplication Module

This module performs the function of de-duplication detection by comparing the incoming file index with the backup node file index. It starts by checking the whole file hash. If the match is found with the hash value along with the file type, then the file is a duplicate one. If the file is identified as duplicate, then it is not saved into the disk. If the match is not found with the hash value, then the file assumed as new file and it is updated into backup node So here the file is assumed to be duplicate if and only if both the hash value and the file type matches thereby increasing the de-duplication efficiency and it is shown in Fig. 10.

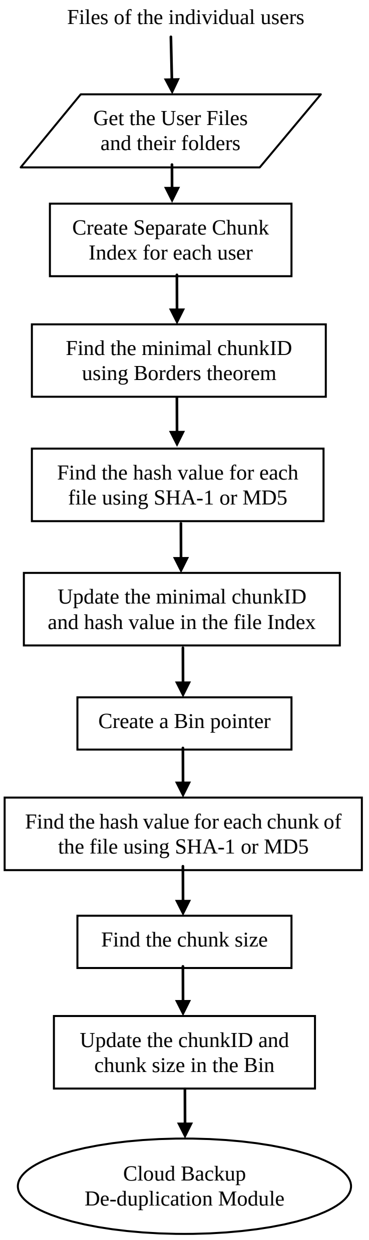

Files of the individual users
Get the User Files and their folders
Create Separate Chunk Index for each user
Find the minimal chunkID using Borders theorem
Find the hash value for each file using SHA-1 or MD5
Update the minimal chunkID and hash value in the file Index
Create a Bin pointer
Find the hash value for each chunk of the file using SHA-1 or MD5
Find the chunk size
Update the chunkID and chunk size in the Bin
Cloud Backup De-duplication Module

Fig. 9. Intensive indexing module.

V. Implementation

We have implemented this by creating the cloud server, cloud controller and multiple clients on WINDOWS platform. Any number of clients can be registered to the cloud server. The coding is done by using visual studio.Net and back end as Microsoft SQL server. The cloud server node is executed followed by the users registration. All the users can have their individual username and password. They can upload any type of files. Our proposed method namely Intensive Indexing

(I2D) File De-duplication is compared with the Extreme Binning file de-duplication. Our analysis is showing that our proposed system will have efficiency based on the number of files being stored in the backup node. Our result analysis is shown in the Fig. 11, 12 and 13.

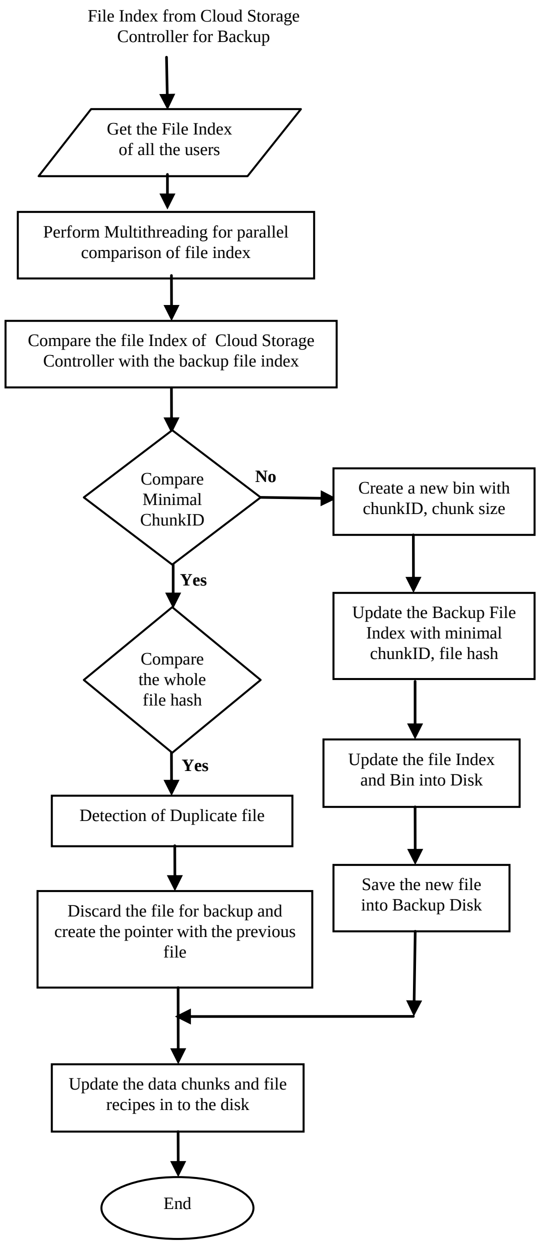
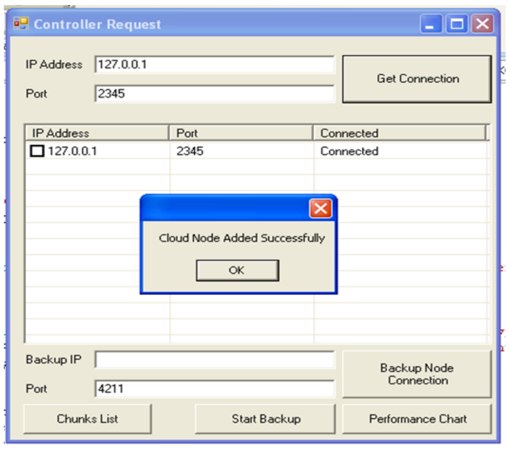

File Index from Cloud Storage Controller for Backup
Get the File Index of all the users
Perform Multithreading for parallel comparison of file index
Compare the file Index of Cloud Storage Controller with the backup file index
Compare Minimal ChunkID
No
Create a new bin with chunkID, chunk size
Yes
Update the Backup File Index with minimal chunkID, file hash
Compare the whole file hash
Yes
Update the file Index and Bin into Disk
Detection of Duplicate file
Save the new file into Backup Disk
Discard the file for backup and create the pointer with the previous file
Update the data chunks and file recipes in to the disk
End

Fig. 10. Cloud backup de-duplication module.



IP Address 127.0.0.1

Port 2345

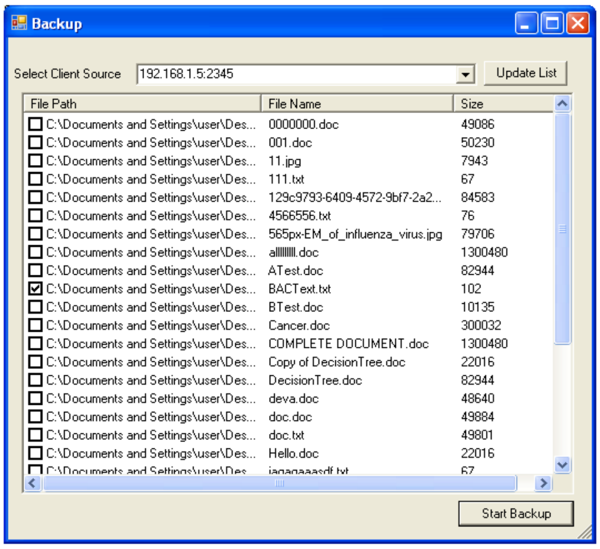
IP Address	Port	Connected
☐ 127.0.0.1	2345	Connected

Cloud Node Added Successfully

Backup IP

Port 4211

Fig. 11. Registering client node to the cloud controller.



Select Client Source 192.168.1.5:2345

File Path	File Name	Size
☐ C:\Documents and Settings\user\Des...	0000000.doc	49086
☐ C:\Documents and Settings\user\Des...	001.doc	50230
☐ C:\Documents and Settings\user\Des...	11.jpg	7943
☐ C:\Documents and Settings\user\Des...	111.txt	67
☐ C:\Documents and Settings\user\Des...	129c9793-6409-4572-9bf7-2a2...	84583
☐ C:\Documents and Settings\user\Des...	4566556.txt	76
☐ C:\Documents and Settings\user\Des...	565px-EM_of_influenza_virus.jpg	79706
☐ C:\Documents and Settings\user\Des...	allllllll.doc	1300480
☐ C:\Documents and Settings\user\Des...	ATest.doc	82944
✓ C:\Documents and Settings\user\Des...	BACText.txt	102
☐ C:\Documents and Settings\user\Des...	BTest.doc	10135
☐ C:\Documents and Settings\user\Des...	Cancer.doc	300032
☐ C:\Documents and Settings\user\Des...	COMPLETE DOCUMENT.doc	1300480
☐ C:\Documents and Settings\user\Des...	Copy of DecisionTree.doc	22016
☐ C:\Documents and Settings\user\Des...	DecisionTree.doc	82944
☐ C:\Documents and Settings\user\Des...	deva.doc	48640
☐ C:\Documents and Settings\user\Des...	doc.doc	49884
☐ C:\Documents and Settings\user\Des...	doc.txt	49801
☐ C:\Documents and Settings\user\Des...	Hello.doc	22016
☐ C:\Documents and Settings\user\Des	iagagaaasdf.txt	67

Fig. 12. Making backup for the particular client.

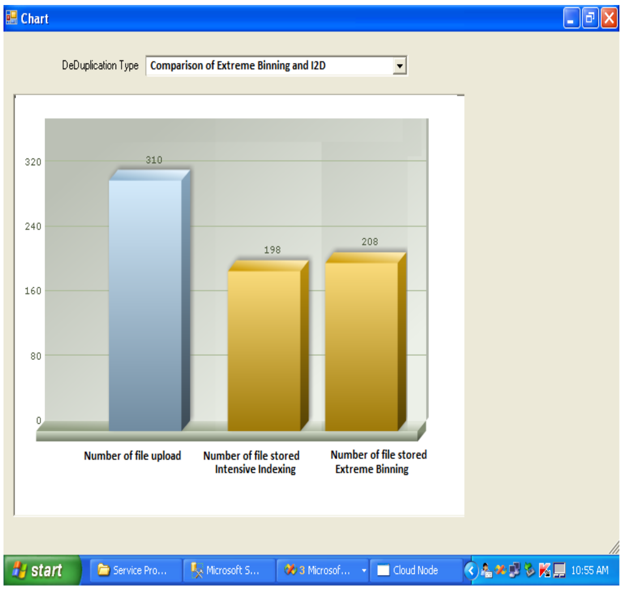


Fig. 13. Performance analysis.

VI. Conclusion

In this paper, we have designed our new scheme namely Intensive Indexing (I2D) De-duplication that effectively removes duplication. It is highly desirable to improve the private cloud backup storage efficiency by reducing the de-duplication time. Our future enhancement is to use chunk level de-duplication in the private cloud storage by overcoming the negative factors in its efficiency.

References

[1] Y. Abe and G. Gibson, "PWalrus: Towards better integration of parallel file systems into cloud storage," in *Proc. IEEE International Conference on Cluster Computing Workshops and Posters*, Heraklion, Crete, 2010, pp. 1–7.

[2] Amazon Web Services LLC. (2009). Amazon simple storage service. [Online]. Available: http://aws.amazon.com/s3/

[3] W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur, "Single instance storage in windows 2000," in *Proc. Fourth USENIX Windows Systems Symp*., 2000, pp. 13-24.

[4] J. Min, D. Yoon, and Y. Won, "Efficient De-duplication techniques in modern backup operation," *IEEE Transactions on Computers*, vol. 60, no. 6, June 2011.

[5] J. Wei, H. Jiang, K. Zhou, and D. Feng, "Mad2: A scalable high-throughput exact de-duplication approach for network backup

services," in *Proc. IEEE / NASA Goddard Conference on Mass Storage Systems and Technologies*, NV, USA, May 2010.

[6] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain de-duplication file system," in *Proc. the 6th USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, 2008, pp. 18:1–18:14.

[7] D. Bhagwat, K. Eshghi, D. D. E. Long, and M. Lillibridge, "Extreme binning: Scalable, parallel de-duplication for chunk-based file backup," in *Proc. IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, 2009, pp. 1-9.

[8] Secure Hash Standard, National Institute of Standards and Technology, FIPS 180-1. (Apr. 1995). [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[9] R. Rivest. (Apr. 1992). The MD5 message-digest algorithm. *IETF, Request For Comments (RFC) 1321*. [Online]. Available: http: //www.ietf.org/rfc/rfc1321.txt

[10] P. Kulkarni, F. Douglis, J. LaVoie, and J. Tracey, "Redundancy elimination within large collections of files," in *Proc. USENIX Ann. Technical Conf. on General Track, 2004,* pp. 59-72.

[11] B. Hong and D. D. E. Long, "Duplicate data elimination in a san file system," in *Proc. 21st IEEE / 12th NASA Goddard Conf. on Mass Storage Systems and Technologies (MSST)*, 2004, pp. 301-314.

[12] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "HYDRAstor: A scalable secondary storage," in *Proc. the 7th USENIX Conference on File and Storage Technologies (FAST)*, San Francisco, CA, USA, Feb. 2009.

[13] C. Policroniades and I. Pratt, "Alternatives for detecting redundancy in storage systems data," in *Proc. Conf. USEXNIX* '04, June 2004.

[14] L. L. You, K. T. Pollack, and D. D. E. Long, "Deep store: An archival storage system architecture," in *Proc. Int'l Conf. Data Engineering,* 2005, pp. 804-8015.

[15] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse indexing: large scale, inline deduplication using sampling and locality," in *Proc. Seventh USENIX Conf. File and Storage Technologies*, 2009.

[16] D. R. Bobbarjung, S. Jagannathan, and C. Dubnicki, "Improving duplicate elimination in storage systems*," ACM Trans. Storage,* vol. 2, no. 4, pp. 424-448, 2006.

[17] W. Zeng, Y. Zhao, K. Ou, and W. Song, "Research on cloud storage architecture and key technologies," in *Proc. the Second International Conference on Interaction Sciences,* 2009, pp. 1044-1048.

[18] C. Policroniades and I. Pratt, "Alternatives for detecting redundancy in storage systems data," in *Proc. ATEC '04,* 2004, pp. 1-15.

[19] G. Forman, K. Eshghi, and S. Chiocchetti, "Finding similar files in large document repositories," in *Proc. the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 394–400.

[20] Amazon's elastic block storage, elastic block storage. [Online]. Available: http://aws.amazon.com/ebs/.

[21] A. Z. Broder, "On the resemblance and containment of documents," *in Proc. SEQUENCES '97*: *the Compression and Complexity of Sequences 1997,* 1997, pp. 21–29.

[22] P. Jaccard, "Etude comparative de la distribution orale dans une portion des Alpes et des Jura," *Bulletin del la Soci ́et ́e Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

M. Shyamala Devi was born in Madurai in 1984. She has completed B.E degree in computer science and engineering at P.S.N.A College of Engineering and Technology, Dindigul, TN, India in 2005. She completed her M.E computer science and engineering at P.S.N.A College of Engineering and Technology, Dindigul, TN, India in 2009. She completed her M.B.A systems area at Madurai Kamaraj University, Madurai, TN, India. She is now pursuing her Ph.D. in Anna University, Chennai.

She worked as a lecturer at P.S.N.A College of Engineering and Technology, Dindigul, TN, India from 2005 to 2009. Then she joined as an assistant professor at R.M.D Engineering College, Chennai, TN, India from 2009 to till date. She has authored 7 engineering books titled Theory of Computation, Principles of Compiler Design, Data Structures and Algorithm Analysis, Graphics and Multimedia, Fundamentals of Computer Programming, Digital Computer Fundamentals and Visual Programming, by Shri Krishna HiTech Publishing Pvt Ltd, Chennai, TN, India. She have published three papers in IEEEXplore.

Mrs. M. Shyamala Devi is an active life member of CSI, ISTE, ICST, IAEST and IACSIT. She has received a funded project from CSI on March 2010 for 'Web based speech recognition for visually challenged Users'.

S. Steven Fernandez was born in Chennai in 1994. He has completed his SSLC at Don Bosco Matric Higher Secondary School, Chennai, TamilNadu, India in April 2009. He completed his HSC at Don Bosco Matric Higher Secondary School, Chennai, TamilNadu, India in April 2011. Currently he is a third-year student pursuing B.E degree in computer science and engineering at R.M.D Engineering College, Chennai, TamilNadu, India. He is online certified professional in artificial intelligence. He is an active student member of CSI. He is a BEC certified Professional. He is an member of Entrepreneurial Development Cell of R.M.D Engineering college, Chennai, TamilNadu, India.