Journal of Theoretical and Applied Information Technology
30th April 2015. Vol.74 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INTENSIVE FIXED CHUNKING (IFC) DE-DUPLICATION FOR SPACE OPTIMIZATION IN PRIVATE CLOUD STORAGE BACKUP

[1]M.SHYAMALA DEVI, [2]V.VIMAL KHANNA, [3]M.SHAHEEN SHAH

[1]Assistant Professor, Department of CSE, R.M.D. Engineering College, Chennai
[2]FINAL YEAR B.E Student, Department of CSE, R.M.D. Engineering College, Chennai
[3]FINAL YEAR B.E Student, Department of CSE, R.M.D. Engineering College, Chennai
E-mail: [1]shyamalapmr@gmail.com, [2]vimalkhanna93@gmail.com, [3]shaheen0106@gmail.com

ABSTRACT

Cloud Storage provides users with abundant storage space and make user friendly for immediate data access. But there is a lack of analysis on optimizing cloud storage for effective data access. With the development of storage and technology, digital data has occupied more and more space. According to statistics, 60% of digital data is redundant, and the data compression can only eliminate intra-file redundancy. In order to solve these problems, De-Duplication has been proposed. Many organizations have set up private cloud storage with their unused resources for resource utilization. Since private cloud storage has limited amount of hardware resources, they need to optimally utilize the space to hold maximum data. 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 Fixed Chunking (IFC) 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

Keywords: *Cloud Computing, Private Storage Cloud, Cloud Backup, Data De-Duplication, Chunking, Redundancy*

1. INTRODUCTION

Cloud computing delivers flexible applications, web services and IT infrastructure as a service over the internet using utility pricing model. **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. **Private clouds** are built for the exclusive use of one client and can be built and managed by the organization's own administrator. **Hybrid clouds** combine both public and private cloud models.

1.1 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 provides users with storage space and make user friendly and timely acquire data, which is foundation of all kinds of cloud applications [3]. **Public cloud storage** such as Amazon's Simple Storage Service (S3) provides a multi-tenant storage environment [19]. **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. Cloud storage backup [3] is a strategy for backing up data that involves removing data offsite to a managed service provider for protection.

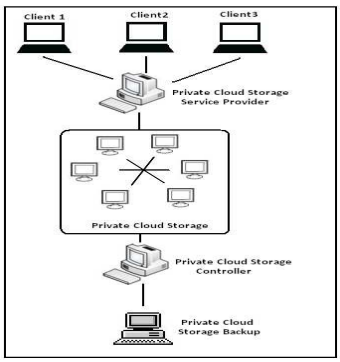

Figure 1: Private Cloud Storage

Journal of Theoretical and Applied Information Technology
30th April 2015. Vol.74 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1.2 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" which is the method of reducing storage needs by eliminating redundant data [10]. 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.

1.3 De-Duplication Techniques

The optimization of backup storage technique is shown in figure 2. The Data de-duplication can operate at the whole file, block (Chunk), and bit level [1, 2, 5]. **Whole file de-duplication** 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. If the new file is having new file index, then it is updated to the storage.

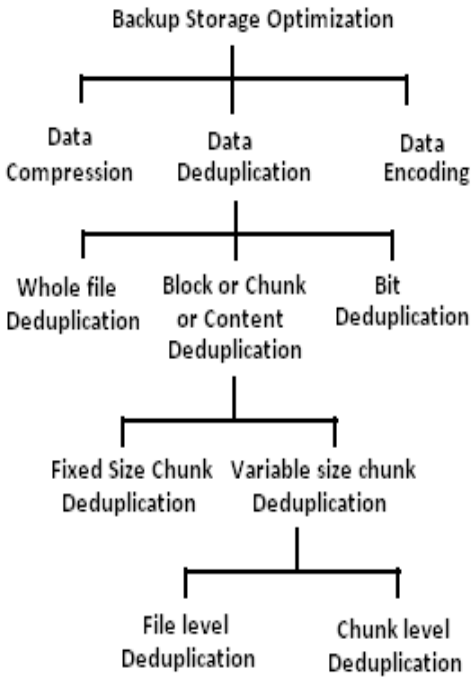

Figure 2: De-Duplication Methods

Block De-duplication [4, 5] 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 or MD5. 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.

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.

2. ANALYSIS OF EXISTING METHODS

2.1 Advantages of Existing Methods

i) 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.

ii) Fixed-size chunking is conceptually simple and fast since it requires less processing power due to the smaller index and reduced number of comparisons.

iii) 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.

iv) Bit De-duplication done exact de-duplication and it is more efficient since it eliminates redundancy.

2.2 Disadvantages of Existing Methods

i) 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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.
ii) 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.
iii) 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 and takes more processing time to identify the duplicate

2.3 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.**

2.4 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 DDDFS is a file system which performs chunk level de-duplication [5]. 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-1or MD5 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. Chunk index is the metadata that includes ChunkID and the address of actual chunks in storage. Unique chunks will be compressed and stored in the container.

2.5 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 uses this approach by dividing the chunk index into two tiers namely Primary index and Bin [4]. ***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 contain the sequence of chunked for that file. Refer [4] for knowing 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. 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. If the representative ChunkID, file hash of the incoming file is already present in the primary index, then the file is a duplicate 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 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.

3. OUR CONTRIBUTION

3.1 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

Journal of Theoretical and Applied Information Technology
30th April 2015. Vol.74 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 Fixed Chunking (IFC) File De-duplication which is the modified File Level de-duplication that provides grouping of files of individual users.

3.2 Intensive Fixed Chunking(IFC) File De-duplication

The existing File Level de-duplication (Extreme Binning) is shown in figure 3.

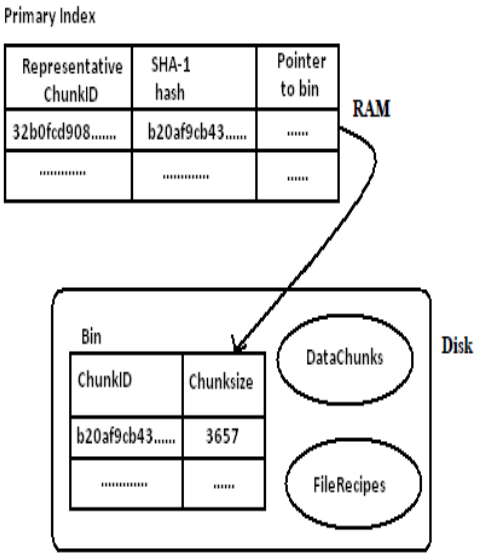

Figure 3: Backup Node in Extreme Binning

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 Fixed Chunking (IFC) 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, Chunk Index and Bin. We create separate file index, Chunk Index and bin for each user and each file belonging to an individual user is grouped with their folders and is shown in figure 4. With this method, it is possible to group the files of each users of the organization.

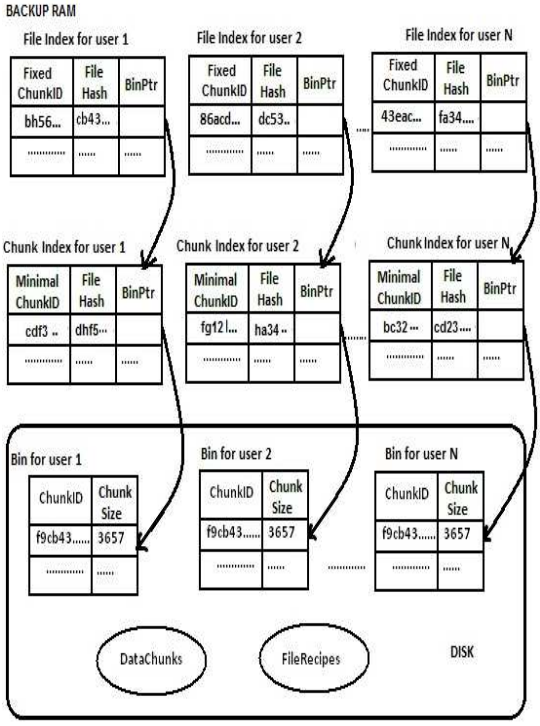

Figure 4: Intensive Fixed Chunking File De-duplication

4. DESIGN OF INTENSIVE FIXED CHUNKING (IFC) FILE DE-DUPLICATION

Before we start our design, we have the following assumptions: i) Users of 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 Fixed Chunking (IFC) File De-duplication scheme has the following modules,

i) Cloud Service Providing Module
ii) Cloud Storage Initiation Module
iii) Cloud Storage Controller Module
iv) Intensive Fixed Chunking Module.
v) Cloud Backup De-duplication Module

4.1. 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 figure 5.

Journal of Theoretical and Applied Information Technology
30th April 2015. Vol.74 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

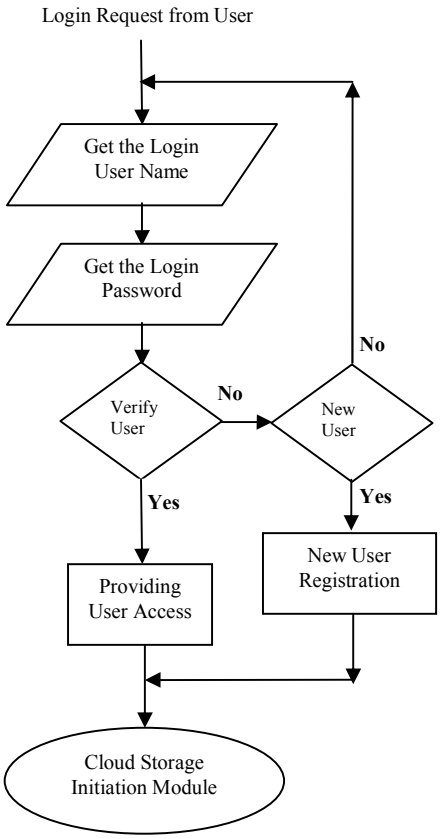

Figure 5: Cloud Service Providing Module

4.2. 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 figure 6. In this module, the authenticated user can perform their own work and they may also try to upload the files from online.

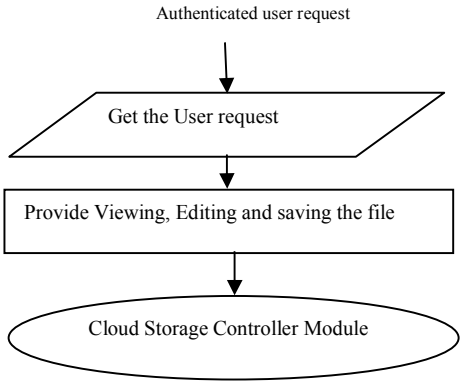

Figure 6: Cloud Storage Initiation Module

4.3. Cloud Storage Controller Module

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 figure 7.

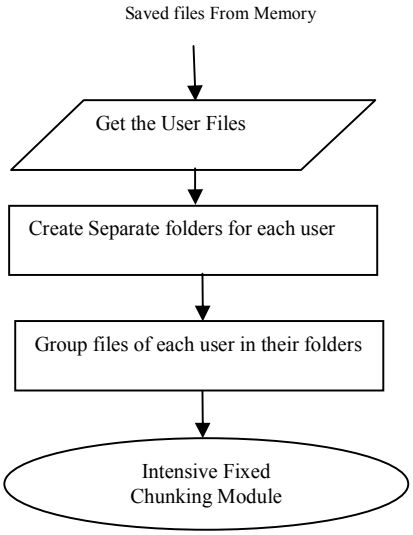

Figure 7: Cloud Storage Controller Module

4.4. Intensive Fixed Chunking Module

The chunk index for each user is created in this module and it is shown in figure 8. The file index and chunk index for each user is created in this module. The file index has three field's namely Fixed ChunkID, File hash and Binptr. First the files for each user are divided into fixed sized chunks. Then each fixed chunkID is palced in the File Index with their size. Now each fixed chunk is again divided into variable sized chunks. The hash value is found for all the different sized chunks. The minimal chunkID of those variable sized chunks is found for all the files. The ID with minimum hash value is choosen to be the minimal ChunkID for the file index. The minimal chunkID is found using Broder's theorem [16]. 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 [17]. 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 or MD5. 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.

Journal of Theoretical and Applied Information Technology
30th April 2015. Vol.74 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

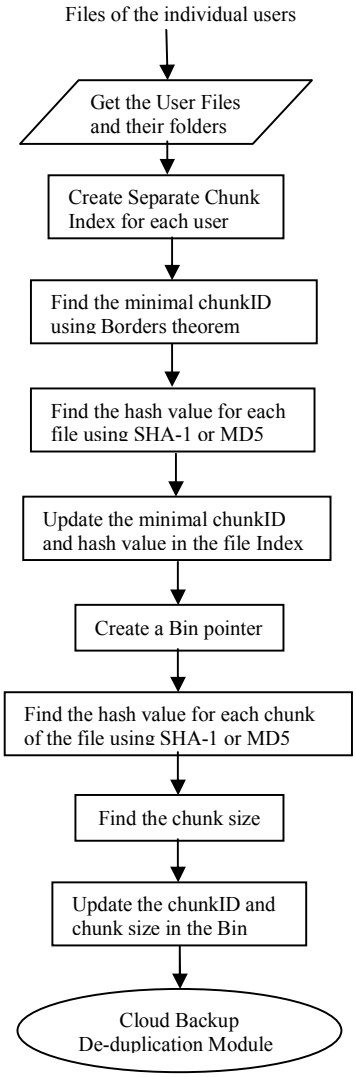

Figure 8: Intensive Fixed Chunking Module

4.5. 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 figure 9.

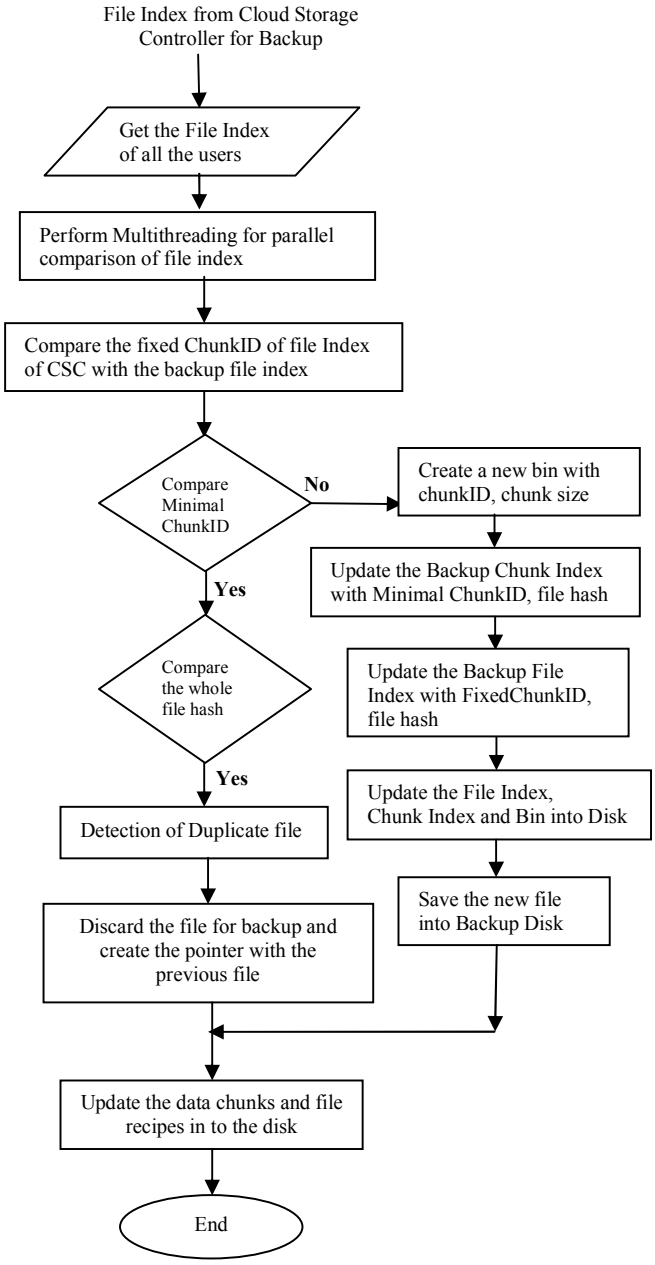

Figure 9: Cloud Backup De-Duplication Module

5. 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Intensive Fixed Chunking 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 figure 10, 11 and 12.

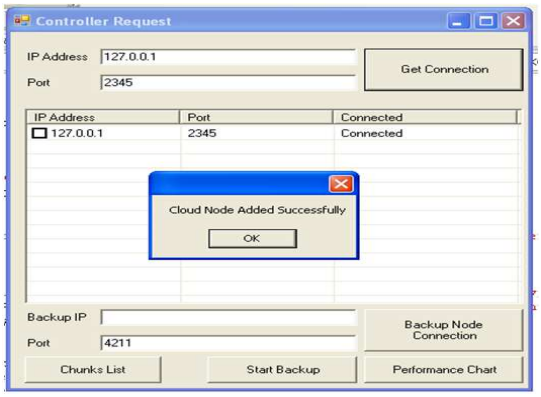

Figure 10: Registering Client To CC

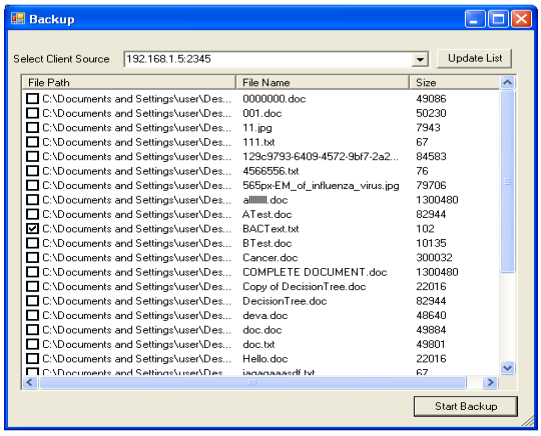

Figure 11: Making Backup For Client

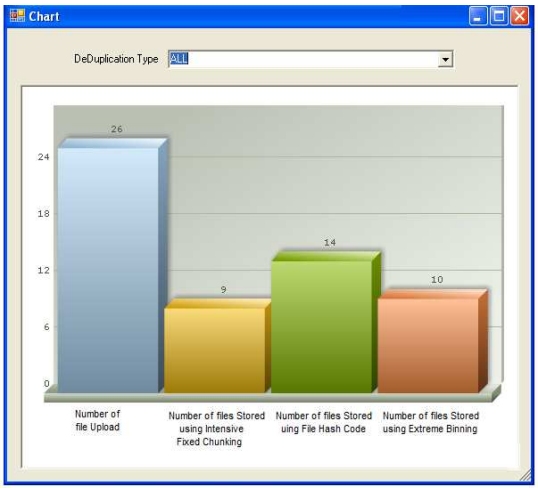

Figure 12: Performance Analysis

6. CONCLUSION

In this paper, we have designed our new scheme namely Intensive Fixed Chunking (IFC) File 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

REFRENCES:

[1] Jaehong Min, Daeyoung Yoon, and Youjip Won," Efficient De-duplication techniques in modern backup operation" IEEE TRANSACTIONS ON COMPUTERS, VOL. 60, NO. 6, JUNE 2011

[2] Wei, et al,"Mad2: A scalable High-throughput exact de-duplication approach for network backup services, Mass Storage Systems and Technologies, IEEE / NASA Goddard Conference.

[3] Abeet al,"Towards better integration of parallel file systems into cloud storage. In Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), IEEE International Conference, pp. 1 –7.

[4] Bhagwat, D., Eshghi, K., and Lillibridge, M. 2009. Extreme binning: Scalable, parallel de-duplication for chunk-based file backup

[5] Zhu, B., Li, K., and Patterson, H. 2008. "Avoiding the disk bottleneck in the data domain de-duplication file system". In Proceedings of the 6th USENIX Conference on File and Storage Technologies, FAST'08, Berkeley, CA, USA. USENIX Association, pp. 18:1–18:14,

[6] P. Kulkarni, F. Douglis, J. LaVoie, and J. Tracey, Redundancy Elimination within Large Collections of Files," Proc. USENIX Ann.Technical Conf., General Track, pp. 59-72, 2004.

[7] B. Hong and D.D.E. Long, "Duplicate Data Elimination in a San File System," Proc. 21st IEEE / 12th NASA Goddard Conf. Mass Storage Systems and Technologies (MSST), pp. 301-314, Apr. 2004.

[8] C. Dubnicki, L. Gryz et al, "HYDRAstor: a Scalable Secondary Storage," in Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST), San Francisco, CA, USA, Feb. 2009.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[9] C. Policroniades and I. Pratt, "Alternatives for Detecting Redundancy in Storage Systems Data," Proc. Conf. USEXNIX '04, June 2004.
[10] W.J. Bolosky et al, "Single Instance Storage in Windows 2000," Proc. Fourth USENIX Windows Systems Symp., pp. 13-24, 2000.
[11] L.L. You, K.T. Pollack, and D.D.E. Long, "Deep Store: An Archival Storage System Architecture," Proc. Int'l Conf. Data Engineering (ICDE '05), pp. 804-8015, 2005.
[12] M. Lillibridge et al,"Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality," Proc. Seventh USENIX Conf. File and Storage Technologies (FAST '09), 2009.
[13] 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.
[14] Zeng W, Zhao Y, Ou K and Song W, 2009, Research on cloud storage architecture and key technologies, ICIS '09: Proceedings of the second International Conference on Interaction Sciences, pp.1044-1048.
[15] Policroniades, C. and Pratt, I. 2004. Alternatives for detecting redundancy in storage systems data, ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference, pp. 1-15.
[16] A. Z. Broder, "On the resemblance and containment of documents," in SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997, pp. 21–29.
[17] P. Jaccard, " Etude comparative de la distribution orale dans une portion des Alpes et des Jura," In Bulletin del la Soci´et´e Vaudoise des Sciences Naturelles, vol. 37, pp. 547–579, 1901.
[18] G. Forman, K. Eshghi, and S. Chiocchetti, "Finding similar files in large document repositories," in KDD '05: Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005, pp. 394–400.
[19] Amazon Web Services LLC, "Amazon Simple Storage Service," http://aws.amazon.com/s3/, 2009.
[20] Amazon's Elastic Block Storage. Elastic Block Storage,[Online] Available:http://aws.amazon.com/ebs/.