# FiLeD: File Level Deduplication Approach

Jyoti Malhotra[*], Jagdish Bakal

*Department of Computer Science & Engineering, G.H. Raisoni College of Engineering,*
*RTM Nagpur University, Nagpur, Maharashtra-440016, India.*

**Abstract:** *In the digital era, uncontrolled data growth is a huge problem. This paper intends to cover the various data storage medium and their backup patterns adopted by end users for their personal data. With respect to an individual concern; the rate of increase in personal data is directly proportional to storage space issues; we focus on an implementation of file-level deduplication, which keeps away the duplicate files. This increases the storage capacity making a room for new data. It also illustrates the comparison of compression, deduplication, and deduplication with compression. We conclude that data will continue to grow and users should seek intelligent methods to shrink the storage space.*

**Keywords**—*Compression, Deduplication, Post-Process, Backup, Fingerprints*

## I. Introduction

Considering the paperless future, everyone is moving towards go green and go paperless strategy. We are investing our documents electronically. These documents create a large data corpus which is easily scattered all over the online storage, personal computers, and laptops. According to the Analytical and expert study of Gartner [1], 90% of data is unstructured data; which is growing fast every year. Once the data is stored, we never delete anything until we are short of the storing space. The easiest way to create a space on our PCs/Laptops is to take a backup of all the files onto some online drives, external hard disks or pen drives. This results in duplication of files and documents as we store our files to multiple sources. On our personal computers or laptops; at any moment, when we are downloading files from any of the online resources, duplicate files may get stored with some suffix "(1)" to the same location; creating multiple instances of the same file(s). In this journey of copy and move of files, we don't realize the amount of doubling the files that eat the storage space. We can have some strategy to get rid of this duplication process. The solution is, either compress the files or de-duplicate them. Compression being a traditional method of shrinking the space helps to reduce the data volume by removing redundancies within the files. De-duplication is another well turned-out compression technique which squashes the data volume by removing redundancies across the files as shown in Figure 1.

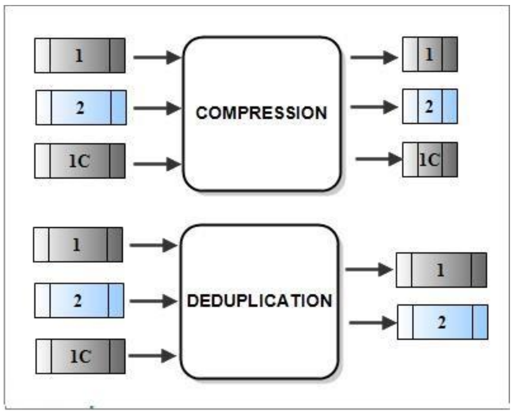

**Fig. 1: Compression Vs De-duplication**

There are two categories of de-duplication [2]: Inline and Post-process. In the first category, duplicates are removed while storing the files. In the later process, duplicates are removed from the stored files.

With the personal data corpus; there is a large probability of finding duplicate files. This paper contributes towards:

1. Post Process de-duplication, i.e. duplicate elimination process is applied on the stored data.
2. Duplicate Elimination process goes through File Level elimination pipeline. Files F, are read to eliminate redundant copies and the results are updated into the Metadata.
3. With a private collection of various files; we showed that the significant amount of space is saved with the De-duplication process.

## II. Related Work

We did a survey to understand the lay of the land related to de-duplication. An article presented in [3] illustrates the popularity of usage of laptop and desktop data, need of taking a backup. The author also covers the difference between the process of laptop backups and enterprise backups; this motivated us to remove the duplicates before moving data for the backup. In [4], the author focuses on various de-duplication design considerations such as chunking, metadata processing, scalability and throughput with respect to text files. Parimala Devi et.al [5] shows the importance of de-duplication in structured records from multiple databases. The author highlights the problems raised due to dirty data i.e. duplicate data. These duplicates are grouped into one cluster and are eliminated effortlessly. In order to remove identical files, Daehee Kim et. al [6] cements the understanding of hybrid approach for storing the data in the cloud systems at the source site. Authors use de-duplication approach on files and emails based on the header and text semantics of files and structure information of emails. Ho Min Jung et. al. [7] represents the file transfer scheme from client to server, enabling energy efficient de-duplication on the basis of file similarity to identify redundancy percentage between the files. The author concludes with the fact that, storage space and computation energy is minimized using file similarity percentage. In [8], the author illustrates the File-level de-duplication on Hadoop.

## III. File level deduplication Approach

### A. *FILE–TO-FILE REDUNDANCY ELIMINATION DESIGN*

Entire De-duplication process can be viewed in two layers; namely Data Source Layer (DSL) and LDSU Layer as shown in Figure 2. The main objective is to eradicate exact identical files and increasing the storage space.

**Data Source Layer** – All type of data (files) is acquired for the backup process.

**LDSU (Load; De-dupe; Store; Update) Layer** - *Load* process access files from the Data Source Layer and Loads/Redirects it for De-dupe. *De-dupe* performs de-duplication; a hash value lookup process of checking duplicates using file hash values. *Store* the next process, which keeps the unique files in the device, and removes the duplicate files. *The update* is the last process; where metadata is updated with a new entry for unique file or pointer for the duplicate file.

Design assesses an overhead of multiple copies and downloads. Duplicate elimination is based on indexing and a hash value (i.e. fingerprints) check.

Duplicate Elimination process goes through File Level elimination pipeline; where Files F, are read to apply the Hashing function <H>, which is given to Storage Layer, ST for applying De-duplication <DD> to eliminate redundant copies and to update the results into the Metadata, MD as defined in equation (1).

$$F \xrightarrow{<H>} ST \xrightarrow{<DD>} MD \qquad (1)$$

The process follows a de-duplication rank order - 0 for Non-duplicates and 1 for Duplicates as shown in equation (2).

$$\text{Rank order} = \begin{cases} 0 & \text{(Non Duplicates)} \\ 1 & \text{(Duplicates)} \end{cases} \qquad (2)$$

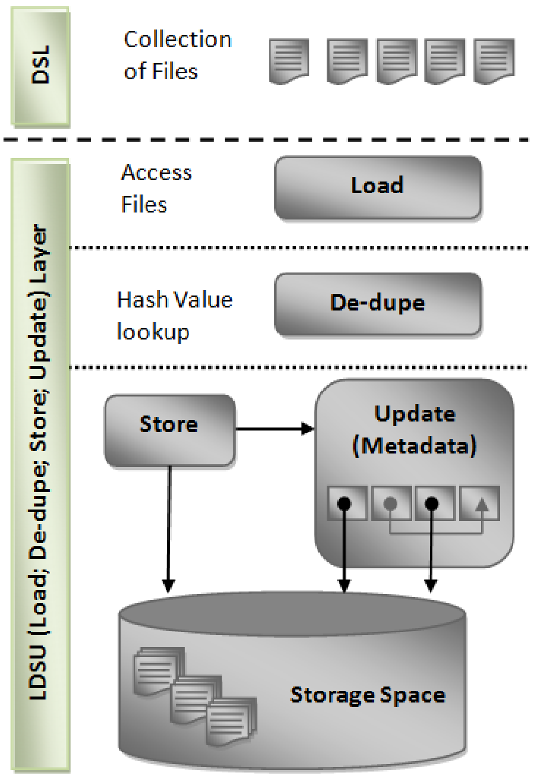

**Fig.2. De-duplication Process**

### B. A Mathematical Model & Algorithm

In this section, we define the mathematical variables used and algorithm for File Level De-duplication.

Considering,

| | |
|---|---|
| $\Omega_f$ | - Set of Files |
| $\Omega_{fD}$ | - Set of Duplicate Files |
| $\Omega_{f\,ND}$ | - Set of Non Duplicate Files |

Entire Sample space can be mapped to equation 3.

$$\Omega_f = \Omega_{fD} + \Omega_{fND} \qquad (3)$$

Files $\Omega_f$, are distributed having two possible outcomes, labelled as Ω=0 and Ω=1, in which Ω=1 i.e. duplicates occurs with probability D; and Ω=0 i.e. non duplicates occurs with probability ND = 1 – D, where 0 < D < 1. Probability of the sample space can be written as shown in equation 4.

$$P(\Omega) = \begin{cases} 1 - D, & \Omega = 0 \\ D, & \Omega = 1 \end{cases} \qquad (4)$$

i.e.

$$P(\Omega) = D^{\Omega}(1 - D)^{1-\Omega}$$

$\Omega_f$ is grouped into a set of duplicates; if files $f$ have equal hash fingerprints. Files are clustered into a set of non duplicates; if files have distinct hash fingerprints.

The De-duplication process is run on a private collection of stored files, to remove the duplicates. The procedure is as explained in Algorithm 1.

**Algorithm 1**: File-to-File De-duplication Process

```
Data: $\Omega_f$; stored fingerPrints[]
Result: A resultant decision for keeping unique
files and eliminating duplicate files.

Begin De-duplicate
  1.  Files[] = access_Directory($\Omega_f$);
  2.  foreach (file in Files[]) do
  3.      fingerprint = getHashFingerPrints (file)
  4.      if ( fingerprint == fingerPrints[] ) then
  5.          Delete this file; File is duplicate
  6.      else
  7.          Save(file)
  8.          Update Metadata
  9.      end if
  10. End foreach
End De-duplicate
```

The algorithmic operational flow is as follows: Inputs are a set of files and stored fingerprints. A Directory is accessed and files are collected in a File vector. For every file f, hash values are calculated. On a fingerprint match; the file is removed from the storage, and metadata is updated by passing the reference pointer of the existing file.

## C. Performance Analysis

This section illustrates the performance analysis based on the following points-

1. Categorical Behavior on the data storage patterns and
2. Clustering of files as duplicates / non-duplicates.

**Categorical Behavior on the Data Storage Patterns:**

A survey was conducted on a public domain where a questionnaire was framed to know from an individual about the data they deal with and its storage patterns. Some of the important questions were as listed below:-

a. What type of data you frequently deal with?
b. Where do you store your data?
c. How often is it that multiple copies of the same data get stored?
d. If you get a product, which would reduce your storage space, how likely would you use that product?
e. How do you save your important data to recover on damage/delete/misplacement?
f. How do you increase your storage space in case of exhaustion of storage space?

Figure 3 (a, b, c, d) shows the diverse responses received for the above questions respectively from the survey.

Figure 3-a illustrates that nearly 74% individual handles unstructured data i.e. information stored in a word document, power point presentation, text file, pdf, XML, log files, HTML, etc; which shows the necessity of applying de-duplication on unstructured files.

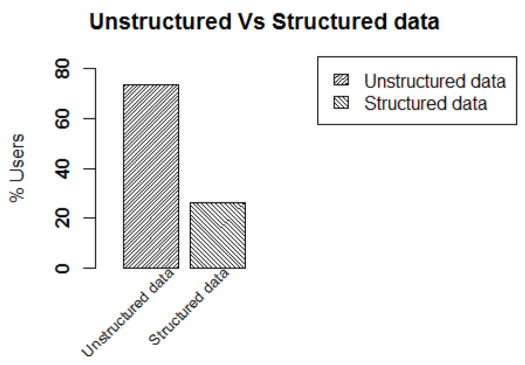

**Fig. 3-a: Type of data frequently used**

It can be seen from figure 3-b, even if cloud is one of the emerging storage eras; many individuals prefer to store their data on local storage; which can be improved, on eliminating duplicate files; creating more space.

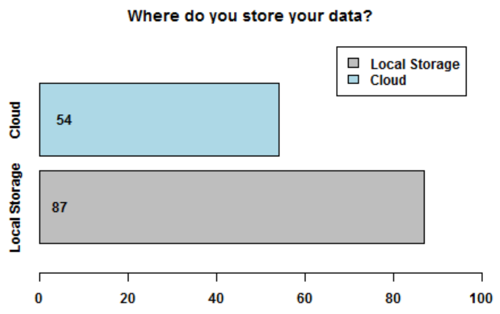

**Fig. 3-b: Data Storage**

Figure 3-c illustrates that many users often creates multiple copies of the same data when their data gets stored on the device. These unwanted exact duplicate copies can be removed using file-level de-duplication.

Figure 3-d shows the interest of individuals towards getting a product, which would reduce the required storage space. Considering "very likely and somewhat likely" to be positive values and "neutral and somewhat unlikely" to be negative values, we observe a significant positive difference in user wanting a product for improving storage space. It was observed; the probability that an individual is inclined towards getting a product is 67%. Seeing these results, it encouraged us furthermore to give the benefits of de-duplication in everyday data handling.

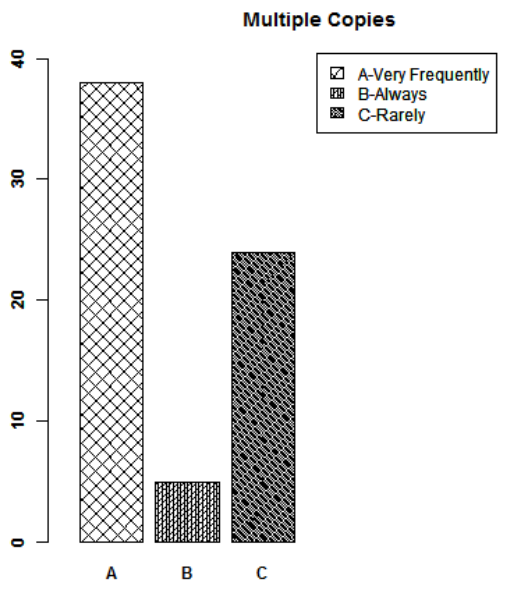

**Fig. 3-c: Multiple copies of same data**

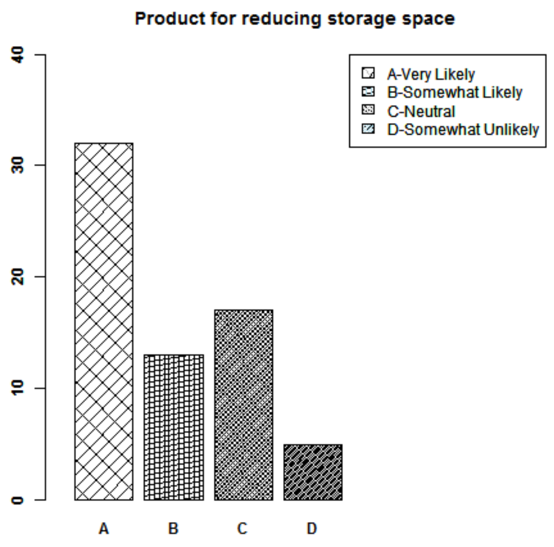

**Fig. 3-d: Product for reducing the storage space**

In the survey, it is also observed that; users save their important data in order to recover it from damage or misplacement by creating multiple copies and storing them at different locations on the local storage; which is later backed up on the cloud storage. On exhaustion of storage space, users either compress the data or increase the storage capacity.

After observing this categorical behavior of individuals on their data storage patterns; a file-level de-duplication was implemented and files were clustered as duplicates and non-duplicates; later duplicate files were deleted successfully without any data loss.

**Clustering of files as duplicates/non-duplicates:**

File level experimentation was performed on various files from different drive location on the local machine. In the storage location there was no particular file type rather it had all formats of file such as docs, pptx, rar,mp3,mp4,flv,html,xml,txt, etc.

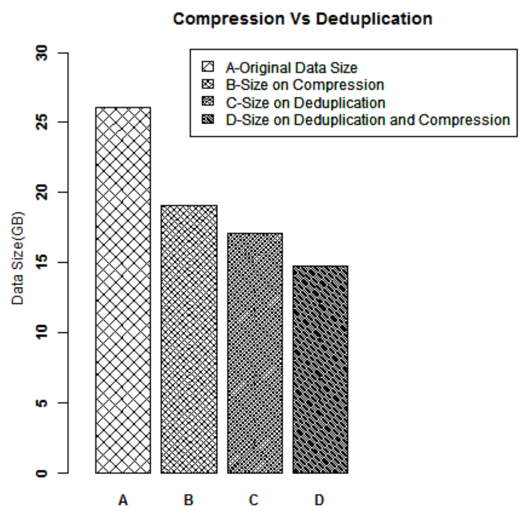

**Fig. 4: Comparison of compression and deduplication**

The original total size of the files for which the experiment was executed was 21.6 GB. Under compression operation, original size got reduced to 19.1GB, whereas when de-duplication was operated, it resulted in a significantly improved change to 17.1GB. But when both the operation was performed concurrently it showed a drastic result in a change to 14.8 GB as shown in figure 4.

Figure 5 shows the variation in the result of CPU usage against compression and de-duplication respectively. While performing the compression on a local machine, all other processes were put to halt. Even then there was a heavy consumption of CPU resources and it went up to inconsiderable 70% CPU usage. Whereas, when we performed de-duplication, as we know apart from de-duplication it simultaneously performs metadata lookup. Even though when two processes were functioning, it only consumed a notable amount of 30% CPU usage.

From figure 6, it can easily seen that the most optimum way to deal with the duplicate data is de-duplication. The valid reason behind it is, it shows the indicative change in the storage saved (primary y axis) from 26% in case of Compression to 34% along with the drastic dip in the time consumption (secondary y axis) when compared again with the compression operation. There was a change from 20 minutes to 8 minutes. These results show that with more duplicate files, deduplication is more powerful than compression.

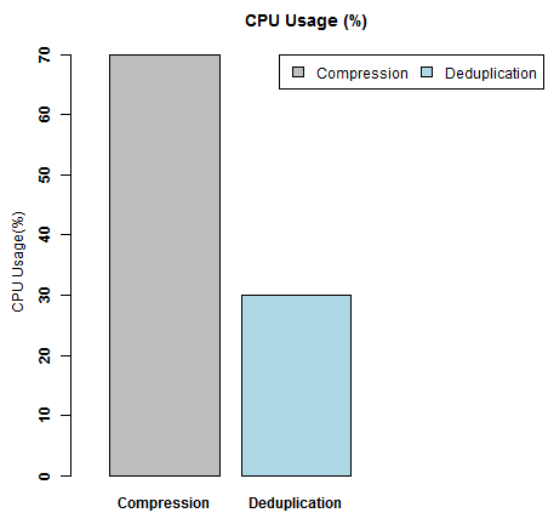

**Fig. 5: CPU utilization**

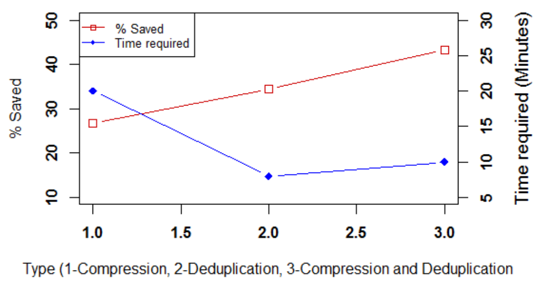

**Fig. 6: Space saved percentage and time required for deduplication**

## D. Concluding Remarks

The problem overhead of multiple downloads and multiple copies of the same data, was addressed using an intelligent and efficient technique of removing the duplicate files. This paper aims at File-level deduplication to identify duplicates; which requires less read throughput. The experiment was done on a private collection of various file types from local drives. The deduplication performance is impacted by the number and the percentage of duplicate content in the dataset; due to which broad remarks cannot be made; here it is observed that good deduplication ratio is achieved in a significant amount of time. Disk space was reduced to approximately 35%. It is also observed the CPU usage percentage for deduplication is less as compared to usage percentage for compression.

## Acknowledgment

We extend a great thank you to everyone who has given us their precious and motivational suggestions and help for achieving the reported work.

## References

[1] http://www.gartner.com/technology/home.jsp
[2] Nagapramod Mandagere, Pin Zhou, Mark A Smith, Sandeep Uttamchandani, “Demystifying data de-duplication”, Proceedings of the ACM/IFIP/USENIX Middleware '08 `
[3] http://searchdatabackup.techtarget.com/feature/Laptop-data-backup-and-desktop-data-backup-best-practices.
[4] Jyoti Malhotra, Jagdish Bakal, “A survey and comparative study of data deduplication techniques”, Pervasive Computing (ICPC), 2015 International Conference on, IEEE ICPC 2015, Pages:1-5
[5] Devi, R. Parimala, and V. Thigarasu. "A Semantic Deduplication of Temporal Dynamic Records from Multiple Web Databases." Indian Journal of Science and Technology 8.34 (2015).
[6] Kim, Daehee, Sejun Song, and Baek-Young Choi. "SAFE: Structure-aware file and email deduplication for cloud-based storage systems." Data Deduplication for Data Optimization for Storage and Network Systems. Springer International Publishing, 2017. 97-115.
[7] Jung, Ho Min, et al. "Energy Efficient Deduplication System Exploiting Similarity Information." Future Information Technology, Application, and Service. Springer Netherlands, 2012. 67-74.
[8] Malhotra, Jyoti, Jagdish Bakal, and L. G. Malik. "Caching: QoS Enabled Metadata Processing Scheme for Data Deduplication." Proceedings of the International Congress on Information and Communication Technology. Springer Singapore, 2016.
[9] Guohua Wang School of Software Engineering, Yuelong Zhao,Xiaoling Xie, and Lin Liu School of Computer Science & Engineering China University of Technology Guangzhou, China,” Research on a clustering data de-duplication mechanism based on Bloom Filter”,2010 IEEE.
[10] Kaiser, J. Meister, D. ; Brinkmann, A. ; Effert, S. “Design of an exact data deduplication cluster” Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on Computing and Processing, Pages(s) 1-12

Jyoti J. Malhotra is B.E. and M.E. in Computer Science and Engineering from the Dr. Babasaheb Ambedkar Marathwada University, Aurangabad and SPPU university Pune in 2001 and 2005, respectively. Presently, she is working as an Assistant Professor in the Department of Information Technology, MIT College of Engineering Pune since December 2004. She has 15+ years of teaching experience. She is pursuing her Ph.D. degree in Computer Science and Engineering; RTM Nagpur University under the guidance of Dr. J. W. Bakal. Her research interest lies in Data Storage patterns, Big Data, Software Testing and Theory of Computation. She has worked in C, Java, and Linux Programming. She is the life member of Computer Society of India. She has delivered expert talks in FDPs and workshops on “Hadoop and Business Intelligence” and “Software Testing”. She is the author of a book on – “Software Testing and Quality Assurance”. She has publications in National, International conferences and Journals like IEEE conference, Springer Conferences, etc.

Dr. J. W. Bakal received M. Tech. (EDT), from Dr. Babasaheb Ambedkar Marathwada University, Aurangabad. Later, He completed his Ph.D. in the field of Computer Engineering from Bharati Vidyapeeth University, Pune. He is presently working as Principal at the S.S. Jondhale College of Engineering, Dombivali (East) Thane, India. In the University of Mumbai, he was on honorary assignment as a chairman, board of studies in Information Technology and Computer Engineering. He is also associated as chairman or member with Govt. committees, University faculty interview committees, for interviews, LIC or various approval works of institutes. He has more than 27 years of academics experience including HOD, Director in earlier Engineering Colleges in India. His research interests are Telecomm Networking, Mobile Computing, Information Security, Sensor Networks and Soft Computing. He has publications in journals, conference proceedings in his credit. During his academic tenure, he has attended, organized and conducted training programs in Computer, Electronics & Telecomm branches. He is a Professional member of IEEE. He is also a life member of professional societies such as IETE, ISTE INDIA, CSI INDIA. He has prominently contributed in the governing council of IETE, New Delhi India.