JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

Kakinada

DATA DUPLICATION REMOVAL USING FILE CHECKSUM

A Project Report submitted to

Jawaharlal Nehru Technological University Kakinada

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING.

By

N Lakshmi Hemalatha(16H71A0517)

M.Gopi Krishna(16H71A0510)

R.Sai Krishna Rakesh(16H71A0543)

K.Honey(16H71A0512)

Under the Guidance of

Ms. K. Vinaya Sree Bai
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Devineni Venkata Ramana & Dr. Hima Sekhar
MIC College of Technology
An Autonomous Institution
Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada | ISO 9001:2015 Certified
NBA Accredited (B.Tech-CSE/ECE/MECH | NAAC Accredited with 'A' Grade
Kanchikacherla-521180, Krishna Dist., A.P., India. Phone: 08678-273535, 273623, Fax: 08678-273569

An ISO 9001:2015
Crtified Institution

2019-20

Devineni Venkata Ramana & Dr. Hima Sekhar
MIC College of Technology
An Autonomous Institution
Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada | ISO 9001:2015 Certified
NBA Accredited (B.Tech-CSE/ECE/MECH | NAAC Accredited with 'A' Grade
Kanchikacherla-521180, Krishna Dist., A.P., India. Phone: 08678-273535, 273623, Fax: 08678-273569

CERTIFICATE

This is to certify that the project work entitled "**Data Duplication Removal Using File Checksum**" is a bonafied work carried out by Ms Lakshmi Hemalatha.N (16H71A0517), Mr. Gopi Krishna. M(16H71A0510), Mr Sai Krishna Rakesh. R(16H71A0543) ,Ms Honey. K(16H71A0512) in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University, Kakinada during the year 2019-2020. It is certified that all corrections/ suggestions indicated for assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the above degree.

Project Guide
(Ms. K. Vinaya Sree Bai)

Head of the Department
(Ms. L. Kanya Kumari)

Examiner

Principal
(Dr. Y. Sudheer Babu)

DECLARATION

Hereby we, who carried out the project on **"Data Deduplication Removal Using File Checksum"**, declare that the matter included in this project report is a genuine work done by us and has not, been submitted to this university or any other university/institute for the fulfilment of the requirement of the degree.

N.Lakshmi Hemalatha(16H71A0517)
M.Gopi Krishna(16H71A0510)
R.Sai Krishna Rakesh(16H71A0543)
K.Honey(16H71A0512)

ACKNOWLEDGEMENT

We are grateful to **Ms. K. Vinaya Sree Bai, Assistant Professor, Department of Computer Science and Engineering,** whose guidance has been immense all throughout the project work.

Our sincere thanks to **Ms. L. Kanya Kumari, Head of the Department, Computer Science and Engineering,** for her valuable guidance and timely advices in the completion of our project.

We express our heartfelt gratitude to **Dr. Y. SUDHEER BABU, Principal** for permitting us to take up our project work and to complete the project successfully.

We would like to express our sincere and heartful thanks to all the lecturers of the department for their continuous cooperation, which has given us the cogency to buildup adamant aspiration over the completion of our project.

Finally we thank one and all who directly and indirectly helped us to complete our project successfully.

ABSTRACT

Data duplication technology usually identifies redundant data quickly and correctly by using file checksum technique. A checksum can determine whether there is redundant data. However, there are the presence of false positives. In order to avoid false positives, we need to compare a new chunk with chunks of data that have been stored. In order to reduce the time to exclude the false positives, current research uses extraction of file data checksum. However, the target file stores multiple attributes such as user id, file name, size, extension, checksum and data-time table. Whenever user uploads a particular file, the system then first calculates the checksum and that checksum is cross verified with the checksum data stored in database. If the file already exists, then it will update the entry else it will make a new entry into the database.

List of Figures

List of Tables

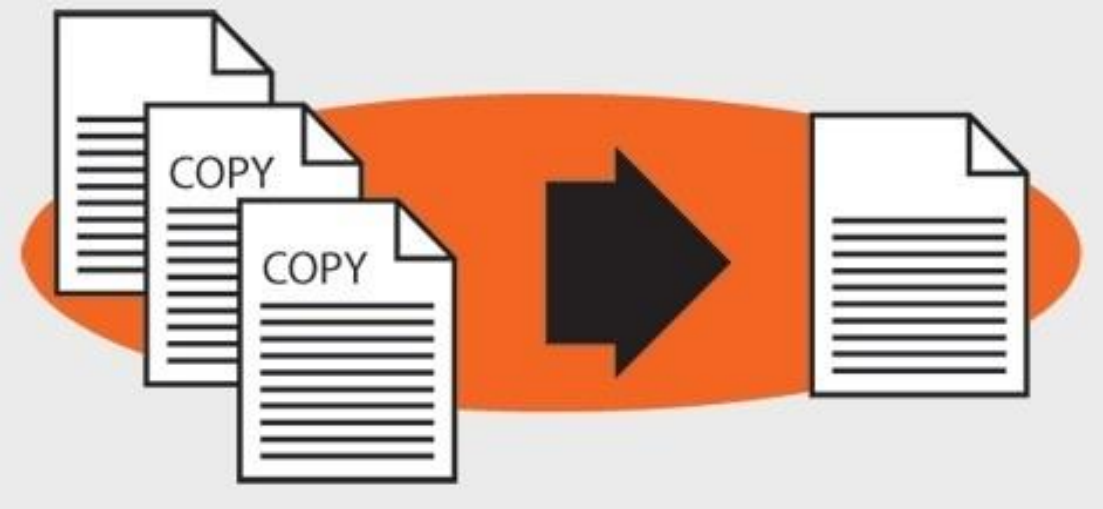
CONTENTS

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND:

Data deduplication -- often called intelligent compression or single-instance storage -- is a process that eliminates redundant copies of data and reduces storage overhead. Data deduplication techniques ensure that only one unique instance of data is retained on storage media, such as disk, flash or tape. Redundant data blocks are replaced with a pointer to the unique data copy. In that way, data deduplication closely aligns with incremental backup, which copies only the data that has changed since the previous backup.

For example, a typical email system might contain 100 instances of the same 1 megabyte (MB) file attachment. If the email platform is backed up or archived, all 100 instances are saved, requiring 100 MB of storage space. With data deduplication, only one instance of the attachment is stored; each subsequent instance is referenced back to the one saved copy. In this example, a 100 MB storage demand drops to 1 MB.



(fig 1.1) email system instances

1.2 PURPOSE:

Data deduplication has an important role in reducing storage consumption to make it affordable to manage in today's explosive data growth. The main goals of this project is, to maximally reduce the amount of duplicates in one type of NoSQL DBs, namely the key-value store, to maximally increase the process performance such that the backup window is marginally affected, and to design with horizontal scaling in mind such that it would run on a Cloud Platform competitively.

While data deduplication is a common concept, not all deduplication techniques are the same. Early breakthroughs in data deduplication were designed for the challenge of the time: reducing storage capacity required and bringing more reliability to data backup to servers and tape. One example is Quantum's use of **file-based or fixed-block-based storage** which focused on reducing storage costs. Appliance vendors like Data Domain further improved on storage savings by using **target-based-** and **variable-block–**based techniques that only required backing up changed data segments rather than all segments, providing yet another layer of efficiency to maximize storage savings.

1.3 PROBLEM STATEMENT:

Data duplication technology usually identifies redundant data quickly and correctly by using file checksum technique. A checksum can determine whether there is redundant data**.** This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times, the amount of data that must be stored or transferred can be greatly reduced.

For example, the same file may be saved in several different places by different users, or two or more files that aren't identical may still include much of the same data. Deduplication eliminates these extra copies by saving just one copy of data and replacing the other copies with pointers that lead back to the original copy.

1.4 SCOPE OF STUDY:

As data deduplication efficiency improved, new challenges arose. How do you backup more and more data across the network, without impacting overall network performance? Avamar addressed this challenge with variable block deduplication and source-based deduplication, compressing data before it ever left the server, thereby reducing network traffic, the amount of data stored on disk, and the time it took to backup. With this step forward, deduplication became more than simply storage savings; it addressed overall performance across networks, ensuring that even in environments with limited bandwidth, data had a chance to be backed up in a reasonable time.Another step function improvement to data deduplication was achieved by Druva when it addressed data redundancies at object level (versus file level) and solved for deduplication across distributed users at a global scale. From this we thought if we can make offline data deduplication that might be very useful.

CHAPTER 2
LITERATURE SURVEY

2.1 DATA FINDING:

Data units are scattered all over the internet as most data items are unorganized, cloud computing tend to bring a definite arrangement pattern to files save on them as all data units on the cloud are assigned a unique index. An advanced search gives users the option of finding files or folders by type, name, title, location, date (taken, modified, or created), size, or property tag. The search locates files and programs stored anywhere in indexed locations, which includes personal folders, e-mail, offline files, and web sites in your History list and ultimately the Cloud. Depending on the file access technology adopted in the cloud storage. Files on the cloud are accessed through a lot of means, for example; files have a Name, Author Name, Size, Keywords all of which are adopted in Search Engine Optimization (SEOs). Searching for a file by the file name is the easiest way of accessing a file over the cloud, then using keywords would help in situations where an external party is searching for a related file and it is done through the use of the Keywords that are defined at the point where the file is stored.

2.2 DATA SHARING:

Data sharing is the practice of distributing or providing access to digital media, such as data, computer programs, multimedia (audio, images and video), documents or electronic books. File sharing may be achieved in a number of ways. Common methods of storage, transmission and dispersion include manual sharing utilizing removable media, centralized servers on computer networks, Cloud Links, World Wide Web-based hyperlinked documents, and the use of distributed peer-to-peer networking. Confidential data are also stored in the cloud using one or more encryption technique. So only the authenticated members who know the key can access the data but everyone can download.

Types of Data Sharing

Peer-To-Peer File Sharing:

Peer-to-peer file sharing is based on the peer-to-peer (P2P) application architecture. Shared files on the computers of other users are indexed on directory servers. P2P technology was used by

popular services like Napster, Spotify, and Infinit. The most popular protocol for P2P sharing is BitTorrent. File Hosting Service File hosting services are a simple alternative to peer-to-peer software. These are sometimes used together with Internet collaboration tools such as email, forums, blogs, or any other medium.

File hosting service, cloud storage service, online file storage provider is an internet hosting service specifically designed to host user files. It allows users to upload files that could then be accessed over the internet after a user name and password or another authentication is provided. Typically, the services allow HTTP access, and sometimes FTP access. Related services are content-displaying hosting services (i.e. video and image), virtual storage, and remote backup.

File Sync and Sharing Services Cloud-based file syncing and sharing services implement automated file transfers by updating files from a dedicated sharing directory on each user's networked devices. Files placed in this folder also are typically accessible through a website and mobile app, and can be easily shared with other users for viewing or collaboration. Such services have become popular via consumer-oriented file hosting services such as Dropbox and Google Drive. Data synchronization in general can use other approaches to share files, such as distributed file systems and version control.

Data Duplication Cloud Service Providers move large amounts of data over a network and provide access to that data as a service, for example, a basic requirement for any cloud-based data protection solution needs to be the ability to reduce the overall costs of providing the same service that clients could do themselves in their own data centers. One method being used to achieve this goal is data deduplication across multiple end user clients, where the costs to provide the service is amortized over the number of paying clients. There are multiple methods to remove duplicate data, so service providers and their customers need to be cognizant of the differences between the available solutions and the impact they may have on security and the ability to efficiently move, protect and store data in the cloud in a cost-effective manner. More storage is not the best answer as storage cost money and the increasing number and size of files eventually burdens the company's backup and disaster recovery (DR) plans. Rather than finding ways Data Finding, Sharing and Duplication Removal in the Cloud Using File Checksum

Algorithm International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) Page 28 to store more data, companies are turning to data reduction technologies that can store less data. Data deduplication emerged as an important part of any data reduction scheme .

Data Deduplication Demystified Data deduplication is basically a means of reducing storage space. It works by eliminating redundant data and ensuring that 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. Data deduplication, sometimes called intelligent compression or single-instance storage, is often used in conjunction with other forms of data reduction. Traditional compression has been around for about three decades, applying mathematical algorithms to data in order to simplify large or repetitious parts of a file effectively making a file smaller.

The business benefits of data de-duplication include;

- Reduced hardware costs; cloud computing gives businesses the ability to enable employees to gain access to data and applications from anywhere, making them more productive when on the go without the need to adopt high performance computers of their own.
- Reduced backup costs; cloud computing environments enable businesses to scale their compute and storage needs up on as-needed basis, which can keep costs low. Additionally, the cloud architecture moves IT spending from capital to operating expenditures, which makes it easier on the books and simpler to justify. Costs are directly aligned with a business usage so they are easy to predict.
- Automation provides agility; an agile business is a successful business and agility is gained from high levels of automation. Cloud computing services are designed to be heavily automated and self provisioning , giving end-users the ability to quickie address their needs. Businesses have the ability to more quickly attend to customer demands, which improves service and responsiveness.
- New Business Models: Cloud computing has made it easier to start business innovation initiatives, often enabled by readily available cloud services.

Data deduplication primarily operates at the file, block and even the bit level. File deduplication is relatively easy to understand if two files are exactly alike, one copy of the file is stored and subsequent iterations receive pointers to the saved file. However, file deduplication is not very efficient because the change of even a single bit results in a totally different copy of the entire file being stored. By comparison, block and bit deduplication looks within a file and saves unique iterations of each block. If a file is updated, only the changed data is saved. This behavior makes block and bit deduplication far more efficient.

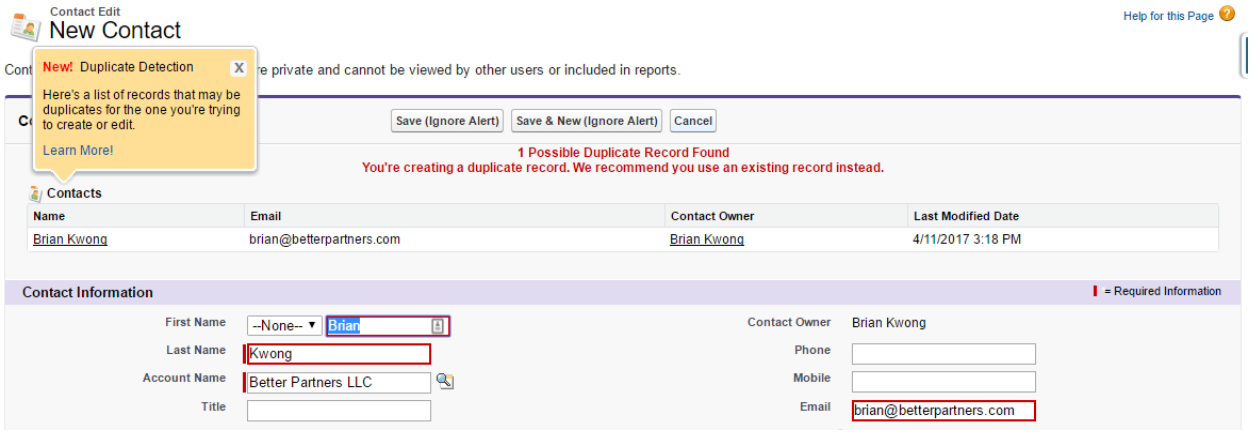
CHAPTER 3

DATA DEDUPLICTION

3.1 EXISTING SYSTEM:

Data Deduplication Application is one of the most common forms of data deduplication implementations work by comparing chunks of data to detect duplicates. For that to happen, each chunk of data is assigned an identification, calculated by the software, typically using cryptographic hash functions.

In our project we implemented an algorithm in such a way so that it can easily detect and remove the duplicate data by comparing chunks of data.



(fig 3.1)Duplicate detection

3.2 PROPOSED SYSTEM:

In the project we are going to use a new method with hash functions where Deduplication may occur "in-line", as data is flowing, or "post-process" after it has been written. Both in-line and post-process architectures may offer bit-for-bit validation of original data for guaranteed data integrity. The hash functions used include standards such as SHA 256, MD-5 and others. The chunks of data in the system are compared by using these algorithm to remove the duplicate data that present in the primary or secondary storage of the system

3.3 ADVANTAGES:

Advantage is the interaction of compression and encryption. The goal of encryption is to eliminate any discernible patterns in the data. Thus encrypted data cannot be deduplicated, even though the underlying data may be redundant.

- Helps reduce the storage place on the drive or server.
- Overcomes the complexity of redundant data.
- Ease of uploading and downloading a file.

3.4 DISADVANTAGES:

The computational resource intensity of the process can be a drawback of data deduplication. To improve performance, some systems utilize both weak and strong hashes. Weak hashes are much faster to calculate but there is a greater risk of a hash collision.

- Takes more time for finding duplicates

3.5 SOFTWARE REQUIREMENTS:

- Windows 7 or higher
- Microsoft SQL Server 2008
- Command prompt or Terminal(For Mac)

3.6 HARDWARE COMPONENTS:

- Processor –Core i3
- Hard Disk – 160 GB
- Memory – 1GB RAM
- Monitor

3.7 FILE CHECKSUM:

File checksum shows you how to evaluate checksums or hashes to remove duplicate from your collection. Data duplication removal involves searching storage devices e.g. hard drive, server, etc. for redundant instances of files and selectively deleting them. While the text which follows refers to image files, the same process can be used for any file type on a computer. As image files take up substantial space, being able to eliminate all but a single instance could significantly decrease the need for storage, especially in cases where there is no strict file handling process when first adding files. File checksum adopts the concepts of file verification (process of using an algorithm for verifying the integrity of a computer file. This can be done by comparing two files bit by bit, but requires two copies of the same file, and may miss systematic corruptions which might occur to both files). The features that can be included in the data duplication removal using file checksum application are as follows:

- **Storage space:** This application can help in reducing the storage space if there are duplicate data.
- **Saves time:** This application can help in saving time since there is no need to go the database to check for the data redundancy.
- **Complexity:** The complexity of the redundant data can be overcome through the use of this application.
- **Upload:** Through this application, uploading and downloading of files will be easier.
- **Easy access:** This application can be accessed anytime and anywhere from the world.
- **User friendly:** This application will be user friendly since the user interface will be simple and easy to understand even by the common man.

3.8 CHECKSUM ALGORITHM:

Parity byte or parity word: The simplest checksum algorithm is the so-called longitudinal parity check, which breaks the data into "words" with a fixed number n of bits, and then computes the exclusive or (XOR) of all those words. The result is appended to the message as an extra word. To check the integrity of a message, the receiver computes the exclusive or of all its words, including the checksum; if the result is not a word consisting of n zeros, the receiver knows a transmission error occurred. With this checksum, any transmission error which flips a single bit of the message, or an odd number of bits, will be detected as an incorrect checksum. However, an error which affects two bits will not be detected if those bits lie at the same position in two distinct words. Also swapping of two or more words will not be detected. If the affected bits are independently chosen at random, the probability of a two-bit error being undetected is 1/ n .

Modular Sum: A variant of the previous algorithm is to add all the "words" as unsigned binary numbers, discarding any overflow bits, and append the two's complement of the total as the checksum. To validate a message, the receiver adds all the words in the same manner, including the checksum; if the result is not a word full of zeros, an error must have occurred. This variant too detects any singlebit error, but does not do well in more than one bit.

Position-dependent: The simple checksums described above fail to detect some common errors which affect many bits at once, such as changing the order of data words, or inserting or deleting words with all bits set to zero. The checksum algorithms most used in practice, such as Fletcher's checksum, Adler32, and cyclic redundancy checks (CRCs), address these weaknesses by considering not only the value of each word but also its position in the sequence. This feature generally increases the cost of computing the checksum.

Message Digest: A utility installed which is capable of creating a Checksum file, such as File verification using MD5 Checksums will be used, a spreadsheet application like Microsoft Excel. Its goal is to gather all checksums together for the image collection we want to remove duplicate, sort them by their checksum values, and then use a spreadsheet calculation formula to help us quickly identify the duplicates if any exist. If there are duplicates, then we can use the spreadsheet as our checklist for deleting the files. Summarily, creating a master checksum text file for the image collection, import that text file into a spreadsheet application, locate recurring

checksum values within the spreadsheet, find corresponding digital files and verify they are the same and delete all but one instance of the file.

The MD5 algorithm is designed to be fast on 32-bit machines. Additionally, the MD5 algorithm does not need any large substitution tables; the algorithm code can be written quite compactly.

3.8.1 MD5 CHECKSUM ALGORITHM:

MD5 checksum algorithm which is known as MD5 message-digest is an algorithm that takes as input a message of random length and produces as output a 128-bit fingerprint or message digest of the input. It is estimated that it is computationally impossible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is proposed for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem. We begin by assuming that we have a b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

m_0 m_1 ... m_{b-1}

The following five steps are performed to compute the message digest of the message.

Step 1:Append Padding Bits- The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2: Append Length-A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the loworder 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32- bit)words. Let M[0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

Step 3:Initialize MD Buffer-A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32- bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

Step 4:Process Message in 16-Word Blocks-We first state four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X)Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z) will be independent and unbiased.

The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased. Note that the function H is the bit-wise "xor" or "parity" function of its inputs.

MD5 is implemented in 4 rounds with each round consists of 16 steps or iterations.

Round 0: Step 0 – Step 15, uses F function

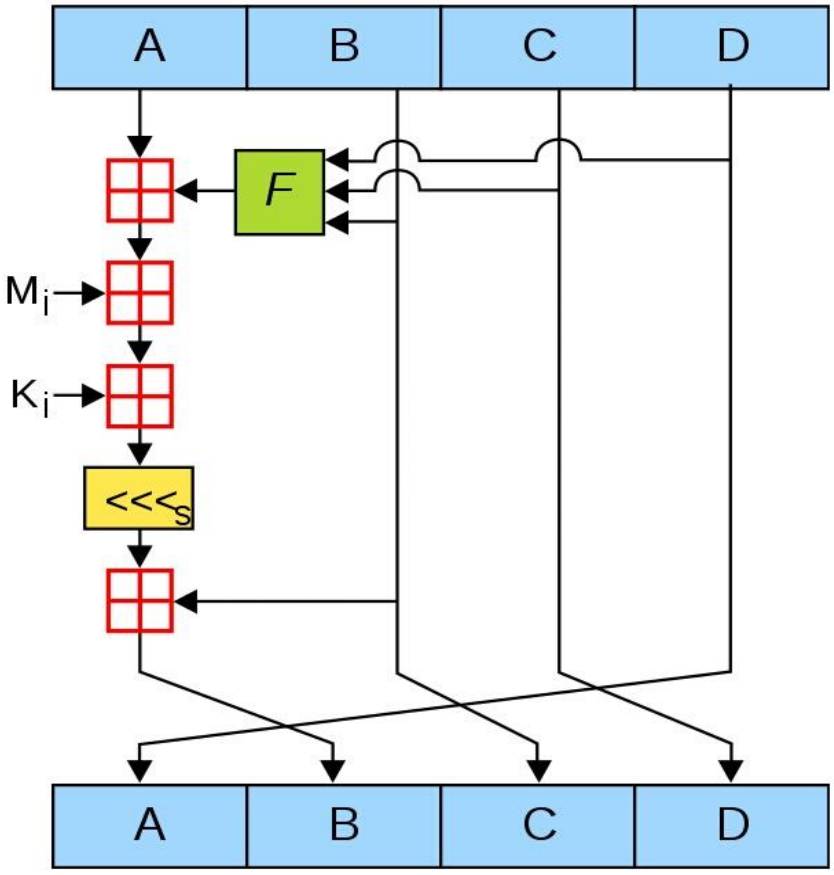
Round 1: Step 16 – Step 31, uses G function

Round 2: Step 32 – Step 47, uses H function

Round 0: Step 48 – Step 63, uses I function

Save AA=A, BB=B, CC=C, DD=D.

The process of each step in all the three rounds is as follows:



(Fig 3.8.1) One MD5 Operation

The figure shows how the auxiliary function F is applied to the four buffers (A, B, C, D), using message word Mi and constant Ki. The term "<<<s" denotes a binary left shift by s bits.
In this process we divide the message into blocks of 512-bits and then each 512-bits block to 16 words of 32-bits. Those 16 words blocks are denoted as M[0….N-1]. Now for each word block we perform 4 rounds where each round has 16 operations.

Each of rounds uses one auxiliary function for each of its operation. Round 1 uses function F, round 2 uses function G, round 3 uses function H, round 4 uses function I.

Round 1:

A=B+((A+F(B, C, D)+Mi+Ki)<<<s), when 0<= i <=15.

Round 2:

A=B+((A+G(B, C, D)+Mi+Ki)<<<s), when 16<= i <=31.

Round 3:

A=B+((A+H(B, C, D)+Mi+Ki)<<<s), when 32<= i <=47.

Round 4:

A=B+((A+F(B, C, D)+Mi+Ki)<<<s), when 48<= i <=63.

After each round is over, then perform the following addition.

A=A+AA

B=B+BB

C=C+CC

D=D+DD

Step 5:Output- After all operations and all 4 previous steps are done. The buffer A, B, C, D contains MD5 digest of our input message.

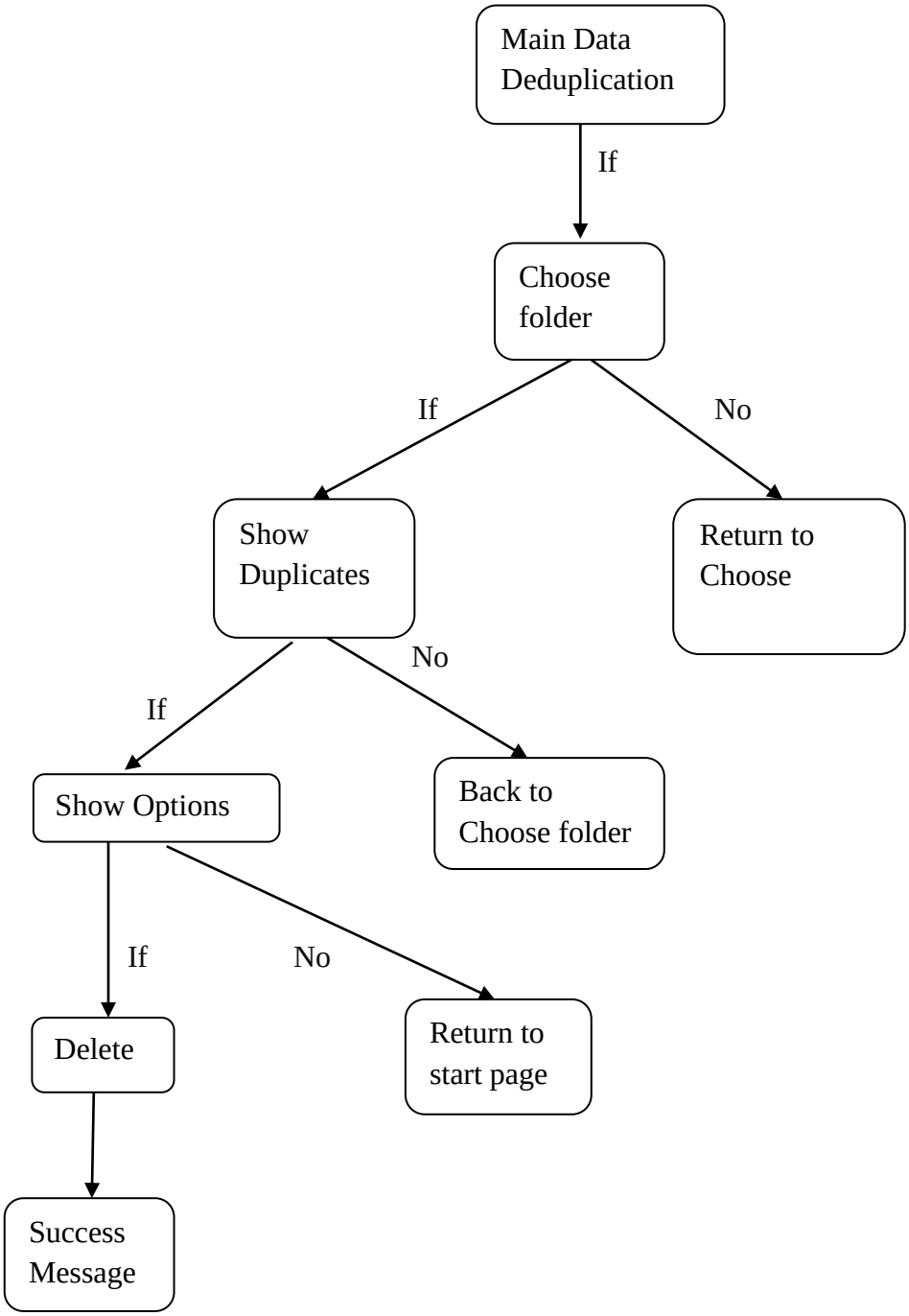
CHAPTER 4
WORKING

4.1 HOW IT WORKS:

- Firstly open a Command prompt or a Terminal and then run the software file.
- It asks for “Choose a folder or Drag and Drop files” after adding the files which contain the duplicate files or data that we want to merge or delete.
- Then the command takes you to another line asking whether you want to delete the data or return to start page so choose the right option that you want to perform the action like which action do you need….?
- 1.DELETE or 2.RETURN
- Then the action completed menu and success message will be displayed.

4.2 FLOW CHART:

A flowchart is a graphical representation of steps. It originated from computer science as a tool for representing algorithms and programming logic but had extended to use in all other kinds of processes. Nowadays, flowcharts play an extremely important role in displaying information and assisting reasoning. They help us visualize complex processes, or make explicit the structure of problems and tasks. A flowchart can also be used to define a process or project to be implemented.



(fig 4.2) graphical representation

4.3 DATA FLOW DIAGRAM(DFD'S):

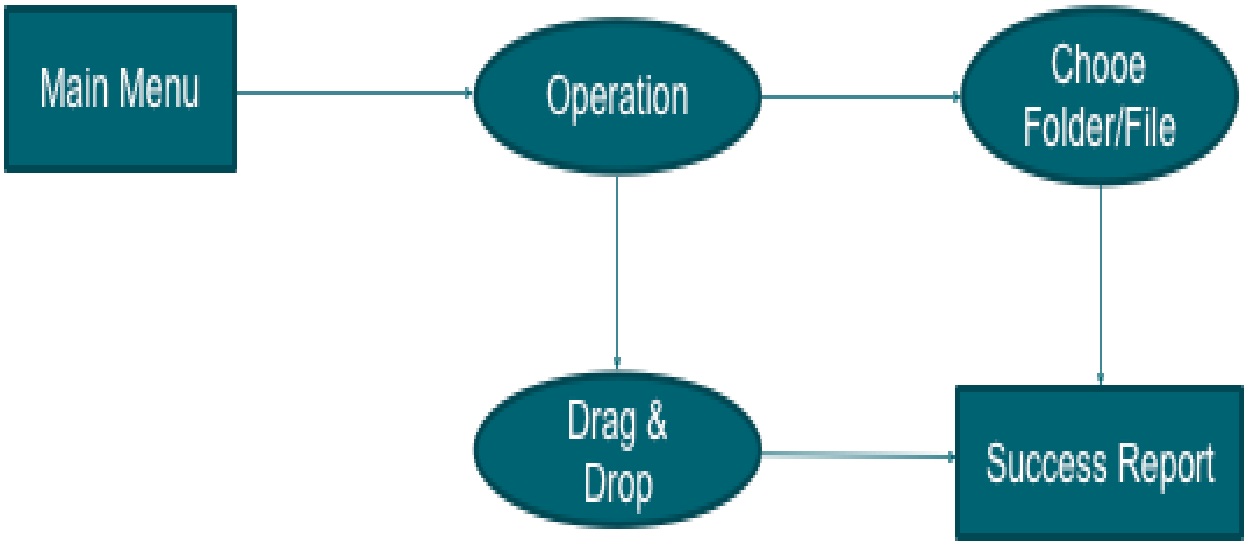
Data flow diagram also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

4.3.1 LEVEL 0 DFD:

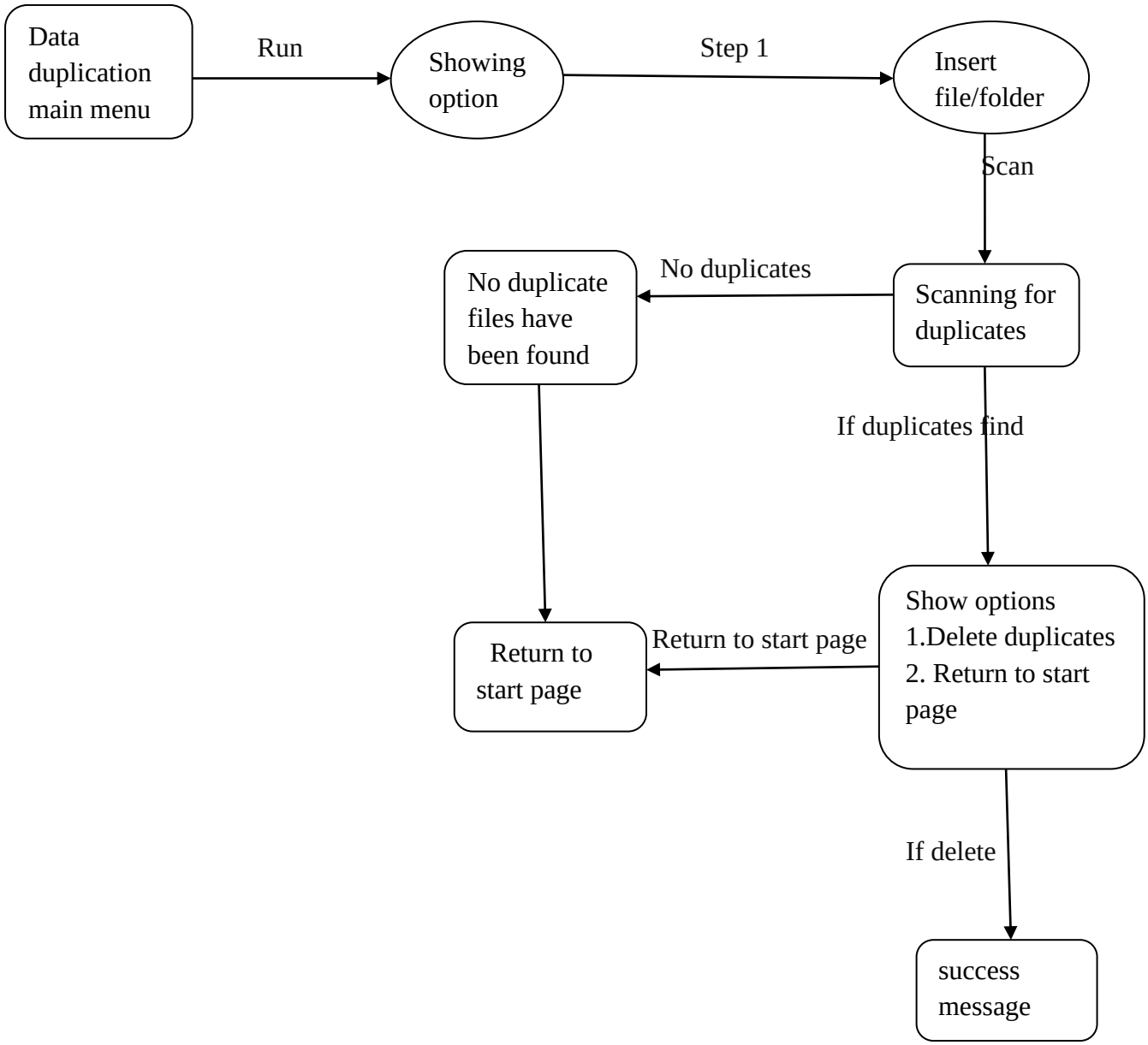
(fig 4.3.1)level 0 dfd

4.3.2 LEVEL 1 DFD:



(fig 4.3.2) level 1 dfd

4.3.3 LEVEL 2 DFD:



(fig 4.3.3) level 2 dfd

CHAPTER 5
APPROACHES AND TECHNIQUES

5.1 DATA DEDUPLICATION APPROACHES:

Data deduplication can be applied in various forms as follows:

5.1.1 FILE LEVEL DEDUPLICATION:

File-level data deduplication compares a file to be backed up or archived with copies that are already stored. This is done by checking its attributes against an index. If the file is unique, it is stored and the index is updated; if not, only a pointer to the existing file is stored. The result is that only one instance of the file is saved, and subsequent copies are replaced with a stub that points to the original file.

5.1.2 BLOCK LEVEL DEDUPLICATION:

Block-level deduplication looks within a file and saves unique iterations of each block. All the blocks are broken into chunks with the same fixed length. Each chunk of data is processed using a hash algorithm, such as MD5 or SHA-256.

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way. The 256-bit key makes it a good partner-function for AES. It is defined in the NIST (National Institute of Standards and Technology) standard ‘FIPS 180-4’. NIST also provides a number of test vectors to verify correctness of implementation.

This process generates a unique number for each piece, which is then stored in an index. If a file is updated, only the changed data is saved, even if only a few bytes of the document or presentation have changed. The changes don't constitute an entirely new file. This behavior makes block deduplication far more efficient. However, block deduplication takes more processing power and uses a much larger index to track the individual pieces.

5.1.3 INLINE DEDUPLICATION:

Deduplicating the data before it is written to disk thus it reduces the storage requirement. The inline deduplication only checks the incoming raw blocks and it does not have any knowledge of the files. This forces it to use the fixed-length block approach. Extent of deduplication is less, and only fixed-length block deduplication approach can be used.

5.1.4 POST PROCESSING DEDUPLICATION:

Post-processing deduplication is an asynchronous backup process that removes redundant data after it is written to storage. It can be applied on file-level or sun-file levels. Whole file data checksum can be easily compared with the existing checksums of previous backup files and thus full file level duplicates can be eliminated easily. Duplicate data is removed and replaced with a pointer to the first iteration of the block. The post-processing approach gives users the flexibility to deduplication specific workloads and to quickly recover the most recent backup without hydration. The tradeoff is a larger backup storage capacity than is required with inline deduplication.

5.1.5 SOURCE BASED DEDUPLICATION:

Deduplication is applied when data is on the source i.e. when data is created. Then the non-deduplicate data is backup to the cloud. It helps better and optimized utilization of resources. It is also helpful in incremental backup of new blocks in the user's instances.

5.1.6 TARGET BASED DEDUPLICATION:

Deduplication occurs after data is been stored. Process of removing deduplication occurs when data was not generated at the location. User is unaware of the deduplication process occurrence. Thus this approach helps in storage utilization but does not helps in saving upload bandwidth.

5.2 DEDUPLICATION METHODS:

One of the most common forms of data deduplication implementations works by comparing chunks of data to detect duplicates. For that to happen, each chunk of data is assigned an identification, calculated by the software, typically using cryptographic hash functions. In many implementations, the assumption is made that if the is identical, the data is identical, even though

this cannot be true in all cases due to the pigeonhole principle; other implementations do not assume that two blocks of data with the same identifier are identical, but actually verify that data with the same identification is identical. If the software either assumes that a given identification already exists in deduplication namespace or actually verifies the identity of the two blocks of data, depending on the implementation, then it will replace that duplicate chunk with a link.

Once the data has been deduplicated, upon read back of the file, wherever a link is found, the system simply replaces that link with the referenced data chunk. The deduplication process is intended to be transparent to end users and applications.

Commercial deduplication implementations differ by their chunking methods and architectures.

- **Chunking:** In some systems, chunks are defined by physical layer constraints(e.g. 4 KB block size in WALF). In some systems only complete files are compared, which is called single-instance storage or SIS. The most intelligent(but CPU intensive) method to chunking is generally considered to be sliding-block. In sliding block, a window is passed along the file stream to seek out more naturally occurring internal file boundaries.
- **Client backup deduplication:** This is the process where the deduplication hash calculations are initially created on the source machines. Files that have identical hashes to files already in the target device are not sent, the target device just creates appropriate internal links to reference the duplicated data. The benefit of this is that it avoids data being unnecessarily sent across the network thereby reducing traffic load.
- **Primary storage and Secondary storage:** By definition, primary storage systems are designed for optimal performance, rather than lowest possible cost. The design criteria for these systems is to increase performance, at the expense of other considerations. Moreover, primary storage systems are much less tolerant of any operation that can negatively impact performance. Also by definition, secondary storage systems contain primarily duplicate, or secondarily copies of data. These copies of data are typically not used for actual production operations and as a result are more tolerant of some performance degradation, in exchange for increased efficiency.

Data deduplication has predominantly been used with secondary storage systems. The reasons for this are two-fold. First, data deduplication requires overhead to discover and remove the deduplicate data. In primary storage systems, this overhead may impact performance. The second reason why deduplication is applied to secondary data, is that secondary data tends to

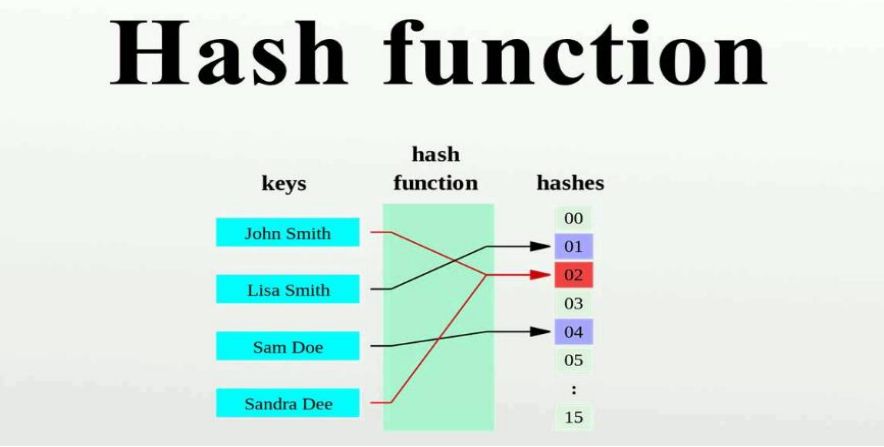
have more duplicate data. Backup applications in particular commonly generate significant od duplicate data over time.

Data deduplication has been deployed successfully with primary storage in some cases where the system designed does not require significant overhead, or impact performance.

5.3 HASH FUNCTION:

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing or scatter storage addressing.

Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval, and storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space efficient form of data access which avoids the non-linear access time of ordered and unordered lists and structured trees, and the often exponential storage requirements of direct access of state spaces of large or variable-length keys.



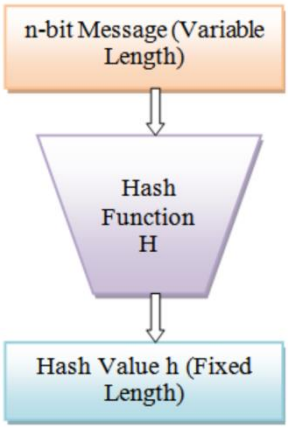
(fig 5.3) Hash function

5.4 HASH VALUE:

A hash value is a numeric value of a fixed length that uniquely identifies data. Hash values represent large amounts of data as much smaller numeric values, so they are used with digital signatures. You can sign a hash value more efficiently than signing the larger value.

Data can be compared to a hash value to determine its integrity. Usually, data is hashed at a certain time and the hash value is protected in some way. At a later time, the data can be hashed again and compared to the protected value. If the hash values match, the data has not been altered. If the values do not match, the data has been corrupted. For this system to work, the protected hash must be encrypted or kept secret from all untrusted parties.

Hash collisions are a potential problem with deduplication. When a piece of data receives a hash number, that number is then compared with the index of other existing hash numbers. If that hash number is already in the index, the piece of data is considered a duplicate and does not need to be stored again. Otherwise, the new hash number is added to the index and the new data is stored. In rare cases, the hash algorithm may produce the same hash number for two different chunks of data. When a hash collision occurs, the system won't store the new data because it sees that its hash number already exists in the index. This is called a false positive, and it can result in data loss. Some vendors combine hash algorithms to reduce the possibility of a hash collision.



(fig 5.4) Hashing algorithm

CHAPTER 6
IMPLEMENTATION

In order to implement data deduplication, you must make several decisions that are based on the outcome that you want to achieve.

6.1 ABOUT THIS WORK:

- Determine database capacity requirements. When you use data deduplication, considerably more database space is required as a result of storing the metadata that is related to duplicate data.
- Determine database log size requirements. It is essential that you properly size the storage capacity for the database active log and archive log.
- Determine which client nodes have data that you want to deduplicate.
- Determine whether you want to implement server-side data deduplication, client-side data deduplication, or a combination of both. To help you make that determination, consider the following factors:
    - Server-side data deduplication is a two-step process in which duplicate data is identified and then storage space is reclaimed to remove the duplicate data. Client-side data deduplication stores the data directly in a deduplicated format.
    - Data deduplication and data compression can be combined on the backup-archive client to reduce data storage. This reduction is typically more than you can achieve by using server-side data deduplication alone.
    - If bandwidth is not restrictive, client-side data deduplication processing typically causes an increase in time for backup operations to complete. Consider doubling the time that you allow for backups when you use client-side data deduplication in an environment that is not limited by the network. If you are creating a secondary copy by using storage pool backup, where the copy storage pool is not using data deduplication, it takes longer for data to be moved because of the extra processing that is required to reconstruct the deduplicated data.

- Duplicate identification processing is handled by client systems when client-side deduplication is used. However, the Tivoli® Storage Manager server still requires processing to handle the lookup requests from clients and to store data deduplication metadata that is produced by client systems.
- Client-side data deduplication cannot be combined with LAN-free data movement that uses the Tivoli Storage Manager for Storage Area Networks feature. If you are implementing a Tivoli Storage Manager supported LAN-free to disk solution, consider server-side data deduplication.

- If you choose client-side data deduplication, decide what, if any, security precautions to take.
- Decide whether you want to define a new storage pool exclusively for data deduplication or update an existing storage pool. The storage pool must be a sequential-access disk (FILE) pool. Data deduplication occurs at the storage-pool level, and all data within a storage pool, except encrypted data, is deduplicated.
- If you want to implement server-side data deduplication, decide how best to control duplicate-identification processes. For example, you might want to run duplicate-identification processes automatically all the time. Alternatively, you might want to start and stop duplicate-identification processes manually. You can also start duplicate-identification processes automatically and then increase or decrease the number of processes depending on your server workload. Whatever you decide, you can change the settings later, after the initial setup, to meet the requirements of your operations.

The following table lists the options that you can use to control duplicate identification processes.

If you create a storage pool for data deduplication...	If you update an existing storage pool...
• You can specify 1 - 20 duplicate-identification processes to start automatically. The Tivoli Storage Manager server does not start any processes if you specify zero. • If you are creating a primary sequential-access storage pool and you do not specify a value, the server starts one process automatically. If you are creating a copy storage pool or an active-data pool and you do not specify a value, the server does not start any processes automatically. • After the storage pool is created, you can increase and decrease the number of duplicate-identification processes manually. You can also start, stop, and restart duplicate-identification processes manually.	• You can specify 0 - 20 duplicate-identification processes to start automatically. If you do not specify any duplicate-identification processes, you must start and stop processes manually. • The Tivoli Storage Manager server does not start any duplicate-identification processes automatically by default.

- Decide whether to define or update a storage pool for data deduplication, but not actually perform data deduplication. For example, suppose that you have a primary sequential-access disk storage pool and a copy sequential-access disk storage pool. Both pools are set up for data deduplication. You might want to run duplicate-identification processes for only the primary storage pool. In this way, only the primary storage pool reads and deduplicates data. However, when the data is moved to the copy storage pool, the data deduplication is preserved, and no duplicate identification is required.
- Determine the best time to use data deduplication for the storage pool. The duplicate identification (IDENTIFY) processes can increase the workload on the processor and system memory. Schedule duplicate identification processes at the following times:
    - When the process does not conflict with other processes such as reclamation, migration, and storage pool backup
    - Before node replication (if node replication is being used) so that node replication can be used in combination with deduplication

6.2 CHECKLIST FOR DATA DEDUPLICATION:

Data deduplication requires additional processing resources on the server or client. Use the checklist to verify that hardware and your Tivoli® Storage Manager configuration have characteristics that are key to good performance.

Question	Tasks, characteristics, options, or settings	More information
• Are you using fast disk storage for the Tivoli Storage Manager database as measured in terms of input/output operations per second (IOPS)?	• Use a high-performance disk for the Tivoli Storage Manager database. At a minimum, use 10000-rpm drives for smaller databases that are 200 GB or less. For databases over 500 GB, use 15000-rpm drives or solid-state drives. • Tivoli Storage Manager database should have a minimum capability of 3,000 IOPS. For each TB of data that is backed up daily (before data deduplication), include an additional 1,000 IOPS to this minimum. • For example, a Tivoli Storage Manager server that is ingesting 3 TB of data per day would need 6,000 IOPS for the database disks: o 3,000 IOPS minimum + 3,000 (3 o TB x 1,000 IOPS) = 6,000 IOPS	• See the checklist for server database disks.

• Do you have enough memory for the size of your database?	• Use a minimum of 64 GB of system memory for Tivoli Storage Manager servers that are deduplicating data. If the retained capacity of backup data grows, the memory requirement might need to be as high as 128 GB. • Monitor memory usage regularly to determine whether more memory is required.	
• Have you properly size your disk space for the database, logs, and storage pools?	• For a rough estimate, plan for 150 GB of database storage for every 10 TB of data that is to be protected in deduplicated storage pools. Protected data is the amount of data before deduplication, including all versions of objects stored. • Configure the server to have the maximum active log size of 128 GB by setting the ACTIVELOGSIZE server option to a value of 131072. • Use a directory for the database archive logs with an initial free capacity of at least 500 GB. Specify the directory by using the ARCHLOG DIRECTORY server option. • Define space for the archive failover log by using the ARCH FAILOVER LOG DIRECTORY server option.	

• Are the Tivoli Storage Manager database and logs on separate disk volumes (LUNs)? • Is the disk that is used for the database configured according to best practices for a transactional database?	• The Tivoli Storage Manager database must not share disk volumes with Tivoli Storage Manager database logs or storage pools, or with any other application or file system.	• See server database and recovery log configuration and tuning.
• Are you using a minimum of 8 (2.2 GHz or equivalent) processor cores for each Tivoli Storage Manager server that you plan to use with data deduplication?	• If you are planning to use client-side data deduplication, verify that client systems have adequate resources available during a backup operation to perform data deduplication processing. Use a processor that is at least the minimum equivalent of one 2.2 GHz processor core per backup process with client-side data deduplication.	

• Have you estimated storage pool capacity to configure enough space for the size of your environment?	• You can estimate storage pool capacity requirements for a deduplicated storage pool by using the following technique: • Estimate the base size of the source data. • Estimate the daily backup size by using an estimated change and growth rate. • Determine retention requirements. • Estimate the total amount of source data by factoring in the base size, daily backup size, and retention requirements. • Apply the deduplication ratio factor. • Round up the estimate to consider transient storage pool usage.	• For an example of using this technique, see Effective Planning and Use of IBM® Tivoli Storage Manager V6 and V7 Deduplication.
• Have you distributed disk I/O over many disk devices and controllers?	• Use arrays that consist of as many disks as possible, which is sometimes referred to as wide striping. • Specify 8 or more file systems for the deduplicated storage pool device class so that I/O is distributed across as many LUNs and physical devices as possible.	• See the checklist for storage pools on disk.
• Do you have adequate resources to cover client-side	• If you are planning to use client-side data deduplication, verify that client systems have adequate resources	

data deduplication requirements?	available during a backup operation to perform data deduplication processing. Use a processor that is at least the minimum equivalent of one 2.2 GHz processor core per backup process with client-side data deduplication.	

CHAPTER 7
TESTING

We, as testers are aware of the various types of Software Testing such as Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their subtypes, etc. Each of us would have come across several types of testing in our testing journey. We might have heard some and we might have worked on some, but not everyone has knowledge about all the testing types. Each type of testing has its own features, advantages, and disadvantages as well. However, in this article, I have covered mostly each and every type of software testing which we usually use in our day to day testing life.

There are two types of testing. They are:

7.1 FUNCTIONAL TESTING:

FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements/specifications.

Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions.

During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester.

Functional testing is normally performed during the levels of System Testing and Acceptance Testing.

Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function’s specifications.
- Determine the output based on the function’s specifications.
- Execute the test case.
- Compare the actual and expected outputs.

Functional testing is more effective when the test conditions are created directly from user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users' wrath when they finally use the software.

Functional Testing types include:

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing

1)**Unit testing** is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

2)Integration testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

3)Performance testing checks the speed, response time, reliability, resource usage, scalability of a software program under their expected workload. The purpose of Performance Testing is not to find functional defects but to eliminate performance bottlenecks in the software or device.

The focus of Performance Testing is checking a software program's

- Speed - Determines whether the application responds quickly.
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads.

4)Regression testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

7.2 NON FUNCTIONAL TESTING:

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of a non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

Non-functional Testing types include:

- Performance Testing
- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing

- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

1)Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

2) Reliability:

The extent to which any software system continuously performs the specified functions without failure. This is tested by Reliability Testing

3) Survivability:

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by Recovery Testing

4) Availability:

The parameter determines the degree to which the user can depend on the system during its operation. This is checked by Stability Testing.

5) Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by Usability Testing

6) Scalability:

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by Scalability Testing

7) Interoperability:

This non-functional parameter checks whether a software system interfaces with other software systems. This is checked by Interoperability Testing

8) Efficiency:

The extent to which any software system can handle capacity, quantity and response time.

9) Flexibility:

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

10) Portability:

The flexibility of software to transfer from its current hardware or software environment.

11) Reusability:

It refers to a portion of the software system that can be converted for use in another application.

CHAPTER 8

CODE

8.1 CODE:

8.1.1 FINDING DUPLICATES:

```
import sys
import os
import hashlib

def chunk_reader(fobj, chunk_size=1024):
    while True:
        chunk = fobj.read(chunk_size)
        if not chunk:
            return
        yield chunk

def get_hash(filename, first_chunk_only=False, hash=hashlib.sha1):
    hashobj = hash()
    file_object = open(filename, 'rb')
    if first_chunk_only:
        hashobj.update(file_object.read(1024))
    else:
        for chunk in chunk_reader(file_object):
            hashobj.update(chunk)
    hashed = hashobj.digest()
    file_object.close()
    return hashed

def check_for_duplicates(paths, hash=hashlib.sha1):
    hashes_by_size = {}
    hashes_on_1k = {}
    hashes_full = {}
```

```
for path in paths:
    for dirpath, dirnames, filenames in os.walk(path):
        for filename in filenames:
            full_path = os.path.join(dirpath, filename)
            try:
                full_path = os.path.realpath(full_path)
                file_size = os.path.getsize(full_path)
            except (OSError,):
                continue
            duplicate = hashes_by_size.get(file_size)
            if duplicate:
                hashes_by_size[file_size].append(full_path)
            else:
                hashes_by_size[file_size] = []
                hashes_by_size[file_size].append(full_path)

for __, files in hashes_by_size.items():
    if len(files) < 2:
        continue

    for filename in files:
        try:
            small_hash = get_hash(filename, first_chunk_only=True)
        except (OSError,):
            continue
        duplicate = hashes_on_1k.get(small_hash)
        if duplicate:
            hashes_on_1k[small_hash].append(filename)
        else:
            hashes_on_1k[small_hash] = []
```

```
            hashes_on_1k[small_hash].append(filename)

    for __, files in hashes_on_1k.items():
        if len(files) < 2:
            continue

        for filename in files:
            try:
                full_hash = get_hash(filename, first_chunk_only=False)
            except (OSError,):
                continue
            duplicate = hashes_full.get(full_hash)
            if duplicate:
                print(filename," & ", duplicate)
            else:
                hashes_full[full_hash] = filename

if sys.argv[1:]:
    check_for_duplicates(sys.argv[1:])
else:
   print("Please pass the paths to check as parameters to the script")
```

8.1.2 HASH FILE:

```
import hashlib
import sys
filename = sys.argv[1]
sha256_hash = hashlib.sha256()
with open(filename,"rb") as f:
   for byte_block in iter(lambda: f.read(4096),b""):
```

```
    sha256_hash.update(byte_block)
  print(sha256_hash.hexdigest())
```

8.1.3 LIST FILES:

```
import os
import sys
path = sys.argv[1]
for root,d_names,f_names in os.walk(path):
  for f in f_names:
    print(os.path.join(root, f))
```

8.1.4 MINIMATCH:

```
var path = { sep: '/' }
try {
 path = require('path')
} catch (er) {}

var GLOBSTAR = minimatch.GLOBSTAR = Minimatch.GLOBSTAR = {}
var expand = require('brace-expansion')

var plTypes = {
 '!': { open: '(?:(?!(?:', close: '))[^/]*?)'},
 '?': { open: '(?:', close: ')?' },
 '+': { open: '(?:', close: ')+' },
 '*': { open: '(?:', close: ')*' },
 '@': { open: '(?:', close: ')' }
}

// any single thing other than /
// don't need to escape / when using new RegExp()
var qmark = '[^/]'
```

```
// * => any number of characters
var star = qmark + '*?'

// ** when dots are allowed.  Anything goes, except .. and .
// not (^ or / followed by one or two dots followed by $ or /),
// followed by anything, any number of times.
var twoStarDot = '(?:(?!(?:\\\/|^)(?:\\.{1,2})($|\\\/)).)*?'

// not a ^ or / followed by a dot,
// followed by anything, any number of times.
var twoStarNoDot = '(?:(?!(?:\\\/|^)\\.).)*?'

// characters that need to be escaped in RegExp.
var reSpecials = charSet('().*{}+?[]^$\\!')

// "abc" -> { a:true, b:true, c:true }
function charSet (s) {
  return s.split('').reduce(function (set, c) {
    set[c] = true
    return set
  }, {})
}

// normalizes slashes.
var slashSplit = /\/+/

minimatch.filter = filter
function filter (pattern, options) {
  options = options || {}
  return function (p, i, list) {
```

```
    return minimatch(p, pattern, options)
  }
}

function ext (a, b) {
  a = a || {}
  b = b || {}
  var t = {}
  Object.keys(b).forEach(function (k) {
    t[k] = b[k]
  })
  Object.keys(a).forEach(function (k) {
    t[k] = a[k]
  })
  return t
}
```

8.2 COMMANDS:

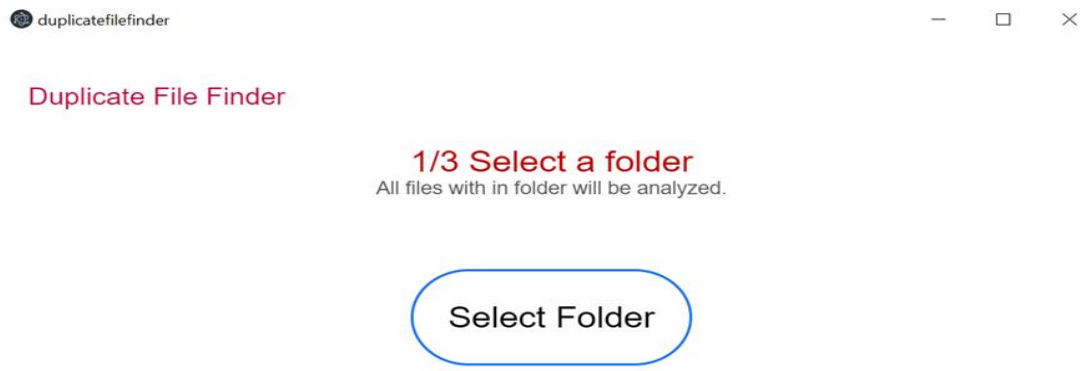
```
# duplicatefilefinder
## Project setup
```
npm install
```
### Compiles and hot-reloads for development
```
npm run serve
```
### Compiles and minifies for production
```
npm run build
```
```

```


### Run your tests
```
npm run test
```
### Lints and fixes files
```
npm run lint
```
### Customize configuration
See [Configuration Reference](https://cli.vuejs.org/config/).
```

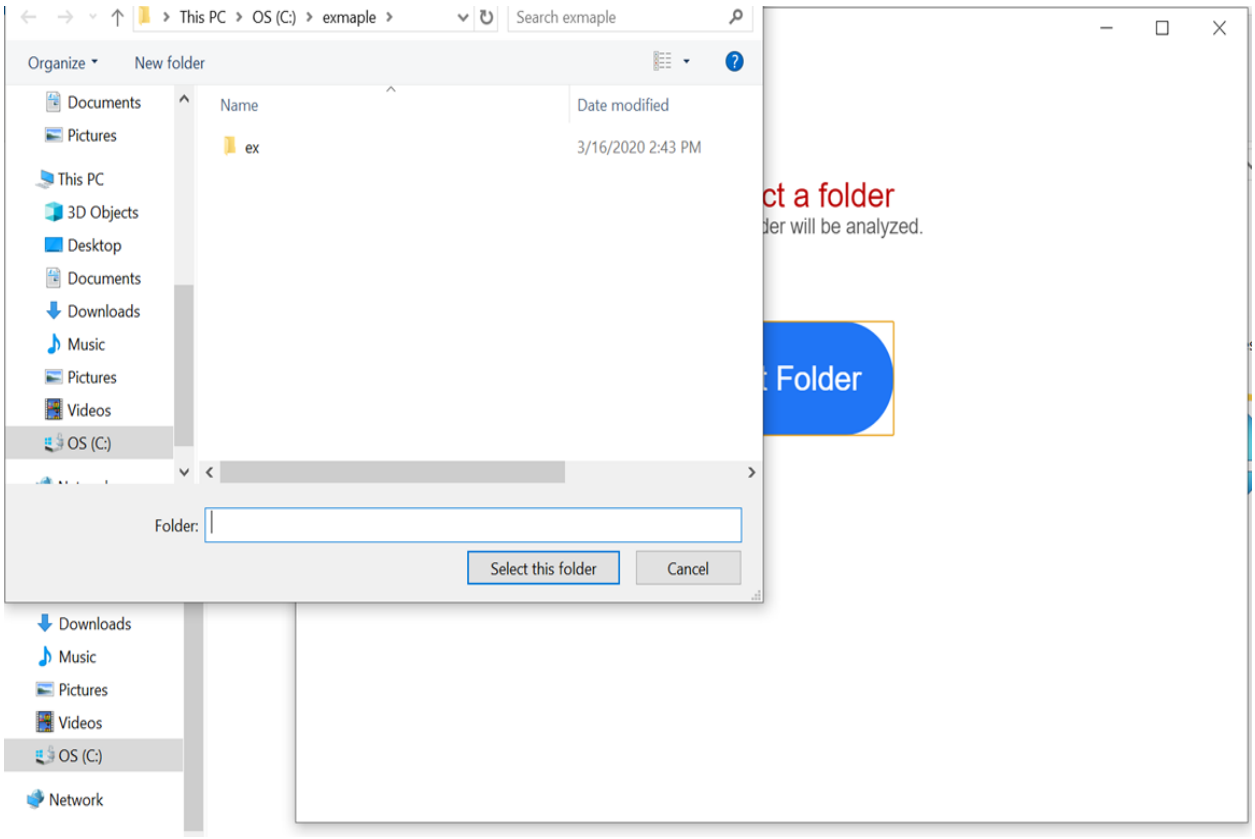
CHAPTER 9

RESULT



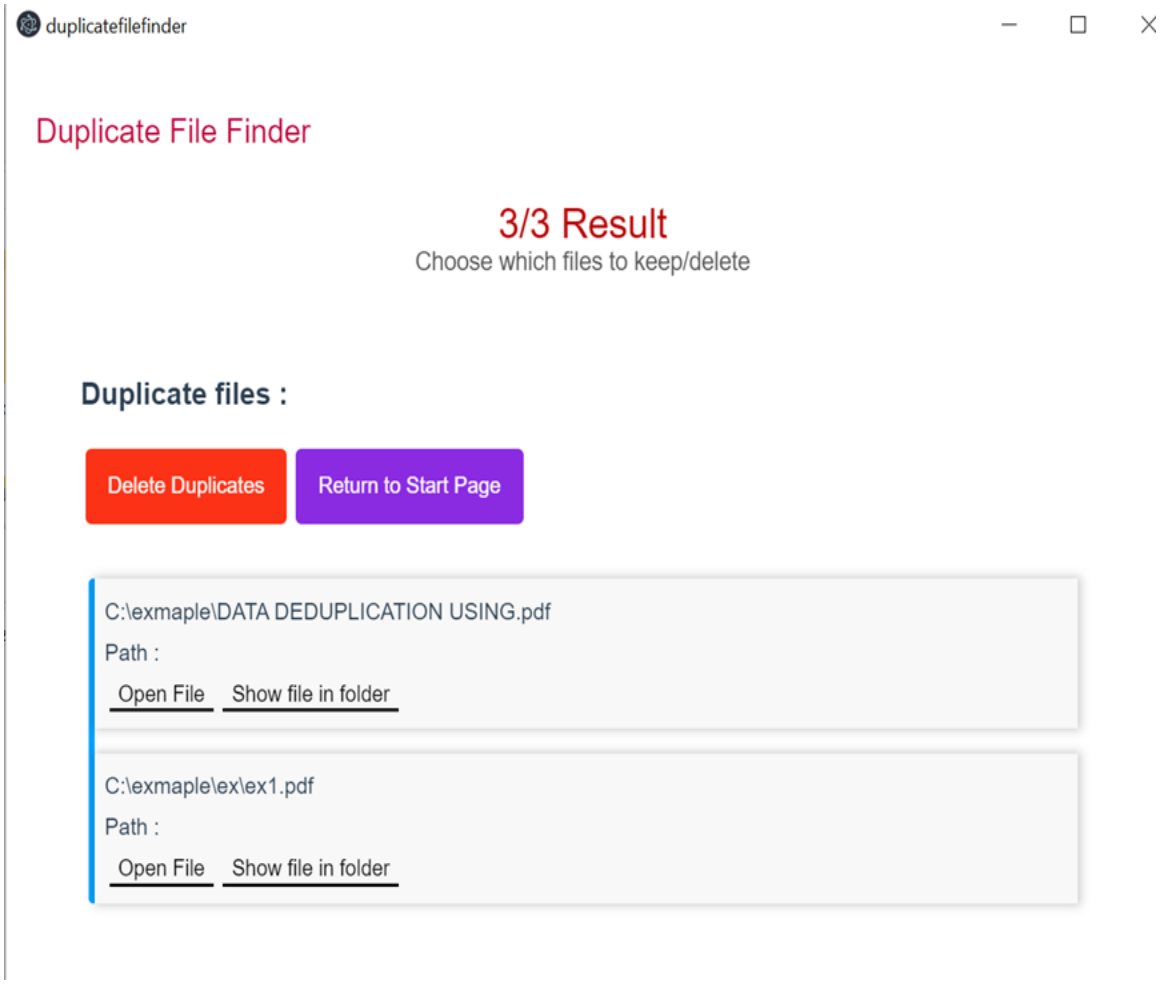
(fig 9.1)run software file

Firstly run the software file. After running that software file new window will be opened. In that window it asks “select folder”.



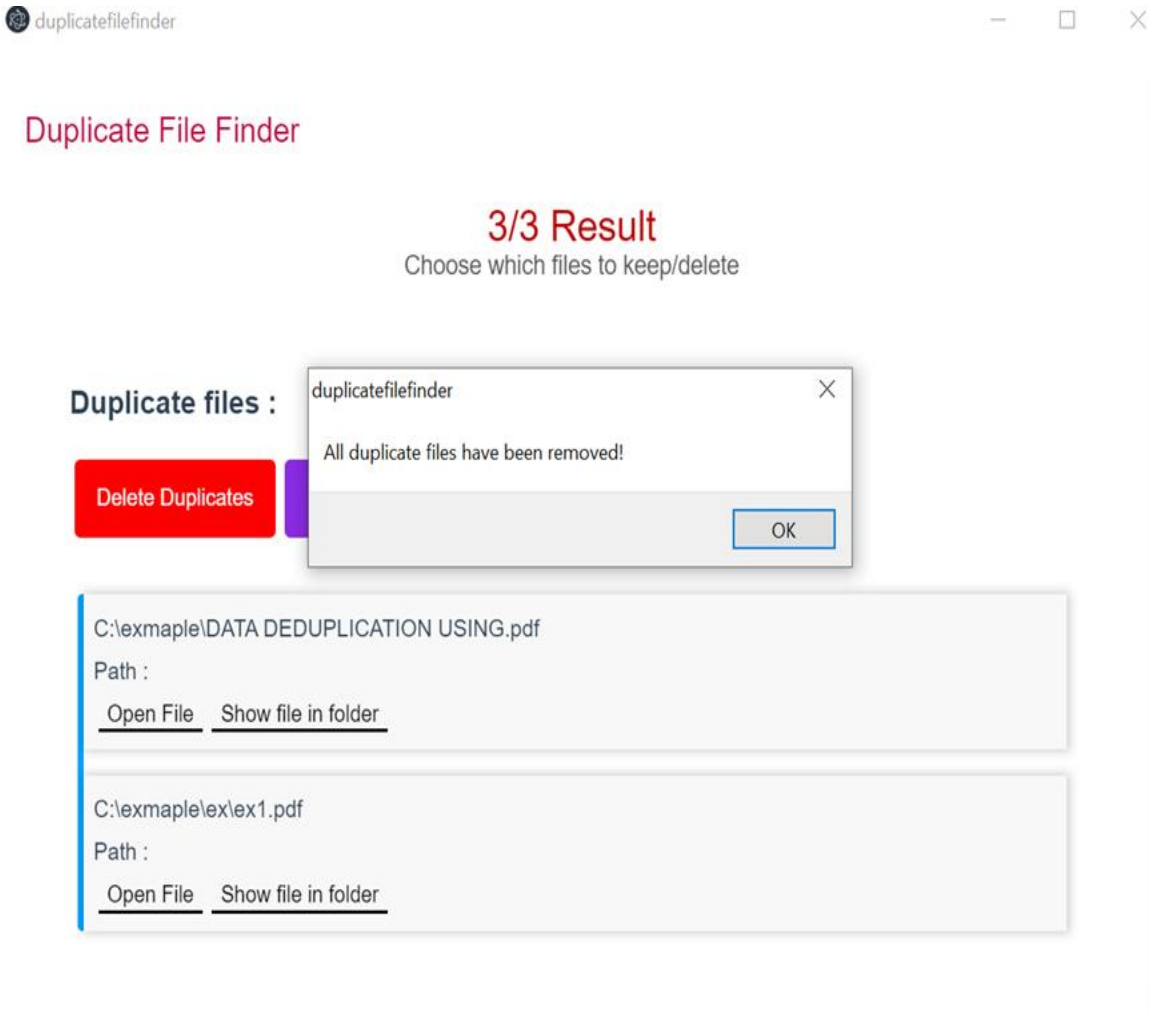
(fig 9.2) selecting folder

After selecting the folder, then add the file which contain duplicate files or data that we want to delete.



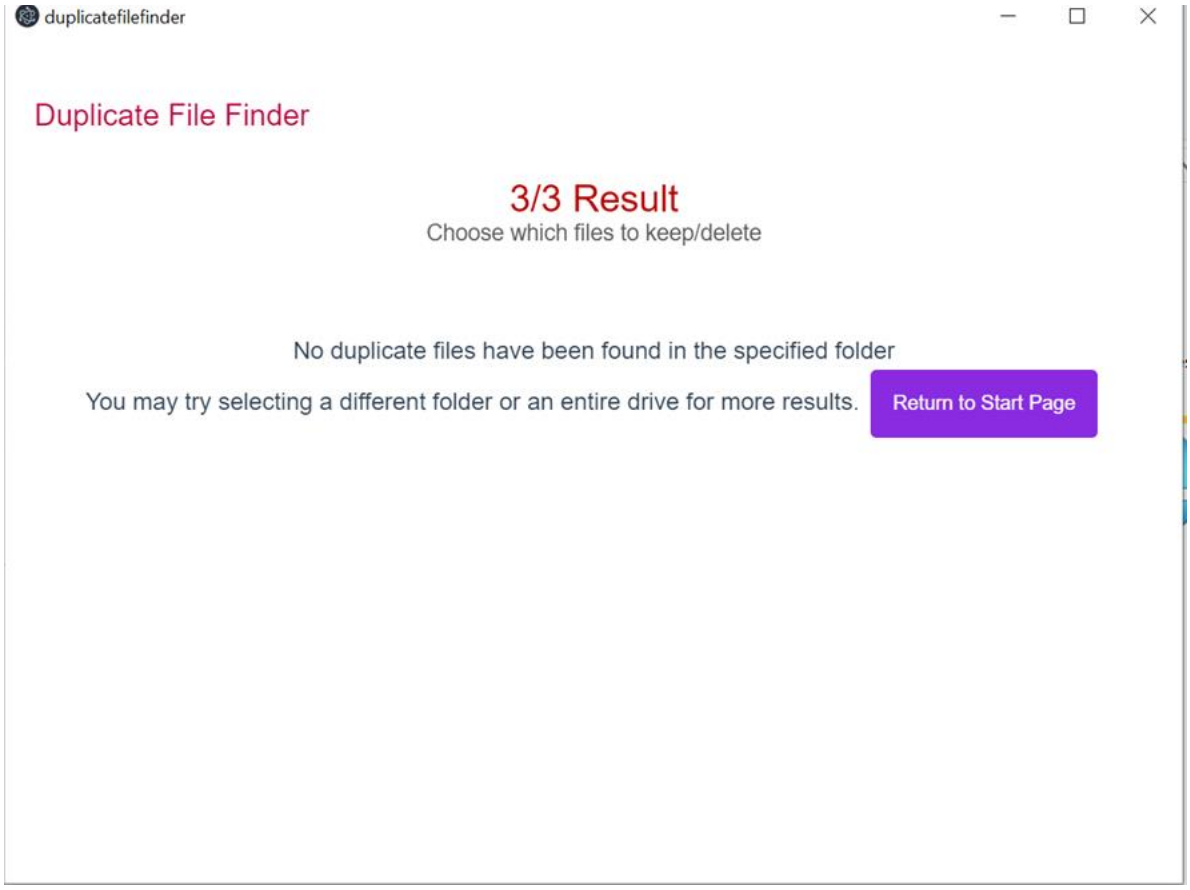
(fig 9.3) find duplicate files

After adding the file which contain the duplicates files or data that we want to delete. Then it asks whether you want to delete the data or return to start page so choose the right option that you want to perform the right action.



(fig 9.4) display success message

If we select delete duplicates option, it will delete the duplicate files and success message will be displayed. Then returns to start page.



(fig 9.5) display success message

If there is no duplicates it will display there is no duplicate files have been found in the specified folder and return to start page.

CHAPTER 10
CONCLUSION

Data duplication technology usually identifies redundant data quickly and correctly by using file checksum technique.MD5 algorithm is used to delete the duplicate files. The chunks of the data in the system are compared by using these algorithm to remove the duplicate data that present in the primary or secondary storage of the system. Deduplication reduces the amount of storage needed for given set of files. Data duplicate removal using file checksum run an analysis and eliminates these sets of duplicate data and keeps only what is unique and essential, thus significantly clearing storage space.

This paper compress the data by removing the duplicate copies of identical data. The core concept involves eliminating the duplicate copies of the repeated data by using hashing algorithms for improving the speed of storing data and security. The proposed system calculates hash values using hashing algorithms and stores them in effective manner for better searching of index. The paper describes for storing , accessing and deleting the files. In our study we have explained the complete framework of the deduplication. The data partitioning and extraction is very crucial steps in the deduplication process, so the choosing of chunking algorithm can decide the entire deduplication performance as well as the correct chunk size can improve the deduplication ratio and the throughput.

Future Work:

This approach is made based on deduplication hash function which helps provide good throughput as well as decreases the space on occupied by repetitive file on system. It worked fruitfully while uploading file on system. It can be applied for heavy multimedia data like video, audio, image files etc. It is much simple to understand and it’s throughput can be merged with another computation to get better result in feature. For future work, the file uploaded to the system can be chunked for integrity with the file that already exists in system. This system work on plane text files it means that this system will not work on the encrypted files. If user uploads the encrypted file then this system will not work for it. But it can be implemented in future as complex logic is required for it.

REFERENCES

[1]. Alok Kumar Kasgar, A Review Paper of Message Digest 5 (MD5), ISSN: 2320-9984 (online), Volume 1, Issue 4, December 2013.

[2]. H. B. Pethe, Dr. S. R. Pande., An Overview of Cryptographic Hash Functions MD5 and SHA.International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), e-ISSN:2278-0661, p-ISSN: 2278-8727.

[3]. Implementation of New Modified MD5-512 bit Algorithm for Cryptography, International Jounal of Innovative Research in Advanced Engineering (IJIRAE), ISSN:2349-2163, Volume 1, Issue 6(july 2014).

[4]. Jaspreet Singh, 2009; Understanding Data Deduplication.

[5]. K. H. Walse. A Novel Method to Improve Data Deduplication System, Volume 6, Issue-10, Oct.2018.

[6]. Nimala Bhadrappa, Mamatha G. S. 2017, Implementation of De-Duplication Algorithm, International Research Journal of Engineering and Technology (IRJET), Volume 04, Issue 09.

[7]. OSUOLALE A.FESTUS., (2019). Data Finding, Sharing and Duplication Removal in the Cloud Using File Checksum Algorithm. International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), Volume 6, Issue 1, 2019, PP26-47, ISSN:2349-4859.

[8]. Rivest R., 1992 The MD5 Message Digest Algorithm. RFC 1321, http://www.ietf.org/rfc /rfc321.txt

[9]. R. Shobana et al., De-Duplication of Data in Cloud, 2016, Volume 4, ISSN 0972-768X.

[10]. Stephen J. Bigelow,2007 Data Deduplication explained:http://searchgate.org; Accessed February, 2018