EFFICIENT CONTENT DEFINED CHUNKING FOR DATA DEDUPLICATION IN IOT ASSISTED CLOUD COMPUTING

By

ABDUL BASIT ASIF

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

June, 2024

Efficient Content Defined Chunking For Data Deduplication In Iot Assisted Cloud Computing

By

ABDUL BASIT ASIF

BSCS, National University Of Modern Languages , Islamabad, 2017

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In **Computer Science**

To

FACULTY OF ENGINEERING & COMPUTING

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

© Abdul Basit Asif, 2024

NATIONAL UNIVERSITY OF MODERN LANGUAGES FACUTY OF ENGINEERING & COMPUTING

THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defence, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering and Computer Sciences for acceptance.

Thesis Title: Efficient Content Defined Chunking for Data De-duplication in IoT assisted Cloud Computing

Submitted By: Abdul Basit Asif **Registration #:** 53 MS/CS/S21

Master of Science in Computer Science (MSCS)
Degree Name in Full

Computer Science
Name of Discipline

Dr. Ata Ullah Research Supervisor	Signature of Research Supervisor
Research Co-Supervisor	Signature of Research Co-Supervisor
Dr. Sajjad Haider Head of Department (CS)	Signature of HoD (CS)
Dr, M. Noman Malik Name of Dean (FEC)	Signature of Dean (FEC)

June 5th, 2024

AUTHOR’S DECLARATION

I Abdul Basit Asif

Son of Asif Mehmood

Registration # 53 MS/CS/S21

Discipline Computer Science

Candidate of **Master of Science in Computer Science (MSCS)** at the National University of Modern Languages do hereby declare that the thesis **Efficient Content Defined Chunking for Data De-duplication in IoT assisted Cloud Computing** submitted by me in partial fulfilment of MSCS degree, is my original work, and has not been submitted or published earlier. I also solemnly declare that it shall not, in future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be cancelled and the degree revoked.

Signature of Candidate

Abdul Basit Asif
Name of Candidate

June 5th, 2024
Date

ABSTRACT

Title: Efficient Content-Defined Chunking for Data De-duplication in IoT-assisted Cloud Computing

Consistently, a large number of data is being created because of the utilization of rising innovations including SaaS products and IoT, etc. The produced data has a lot of redundant items. It is challenging to store and deal with duplicate data. Data De-duplication is a solution to resolve this problem. It is referred to as a method in which only one copy of data is stored on the server to avoid storing excessive copies of data. It's a highly effective approach used to reduce data storage costs as well as make it easier to manage those data. To overcome this issue, one of the approaches called the block-level approach mainly involves the process of chunking, hashing, and indexing those hashes. A chunk of data called a block is converted to a fingerprint and saved in a lookup table. The concerning issue is that as the number of blocks increases, the size of the lookup table also grows which results in additional cost in terms of searching and memory occupation. This work will focus on evaluating an appropriate amount of chunks or blocks to create a balance between redundant data on storage devices and the size of the lookup table. The data used for this purpose belongs to the healthcare sector. The patient's body is connected to various sensors that take readings from the body and transmit them to the fog server. The fog server will further transfer this data to the cloud server. Before transmitting data to the fog server, duplicated data readings are sent in the form of Boolean digit '1' while critical data is sent in its original form to overcome duplication issues. An MQTT-based pub-sub architecture is used to transmit healthcare IoT data in JSON format every second via an MQTT broker to connected receivers. The experimental results indicate that the proposed HDDS scheme outperforms its counterparts.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	THESIS AND DEFENSE APPROVAL FORM	ii
	AUTHOR'S DECLARATION	iii
	ABSTRACT	iv
	TABLE OF CONTENTS	v
	LIST OF TABLES	ix
	LIST OF FIGURES	x
	LIST OF ABBREVIATIONS	xi
	ACKNOWLEDGMENT	xii
	DEDICATION	xiii
1	**INTRODUCTION**	**1**
	1.1 Overview	1
	1.2 Data De-duplication	1
	1.3 Motivation	2
	1.3.1 Architecture of Data De-duplication	2
	1.3.2 Advantages of Data De-duplication	5
	1.3.3 Constraint in Data De-duplication	8
	1.4 Cloud Computing Architecture	10
	1.4.1 Front end Side	10
	1.4.2 Back-end Side	11
	1.4.3 Infrastructure	11
	1.5 Problem background	11
	1.5.1 Negative Impact of Problem	12
	1.6 Problem Statement	13

	1.7	Research Questions	13
	1.8	Aim of Research	13
	1.9	Research Objectives	14
	1.10	Scope of Research	14
	1.11	Thesis organization	14
2	**LITERATURE REVIEW**		**15**
	2.1	Overview	15
	2.2	Data De-duplication	15
		2.2.1 Techniques of Data De-duplication	16
	2.3	Cloud Computing	16
	2.4	Deployment Models	18
	2.5	Internet of Things	19
	2.6	IoT assisted Cloud Computing	20
	2.7	Data De-duplication in IoT assisted Cloud Computing	21
	2.8	Existing Studies in Data de-duplication in IoT assisted Cloud Computing	22
		2.8.1 File Level Data De-duplication	23
		2.8.2 Block-Level Data De-duplication	29
	2.9	Comparison of Data de-duplication in IoT assisted Cloud Computing-based strategies	31
	2.10	Research Gap	37
	2.11	Summary	38
3		**METHODOLOGY**	**39**
	3.1	Overview	39
	3.2	Operational Framework	39
	3.3	Research Design and Development	41
	3.4	Detailed Study of Literature	43
		3.4.1 Problem Identification	43
		3.4.2 Evaluation Metrics	43

		3.4.3 Experimental Dataset and Selection of Simulation Tool	44
	3.5	Summary	44
4	**HEALTHCARE DATA DEDUPLICATION SCHEME**		**45**
	4.1	Overview	45
	4.2	Healthcare Data De-duplication Scheme (HDDS)	45
	4.3	System Model	48
	4.4	Flow Chart of HDDS	50
	4.5	Algorithm for Healthcare Data De-duplication Scheme	51
	4.6	Algorithm for Removing Duplicate Data	52
	4.7	Summary	52
5	**RESULT AND ANALYSIS**		**54**
	5.1	Overview	54
	5.2	Simulation tools and Environment	54
	5.3	Compressed Patients Data File Size	58
	5.4	Average Execution Time	59
	5.5	Size of Data after Post Execution	60
	5.6	Patient Data with Minor Duplicates	61
	5.7	Patient Data with Major Duplicates	62
	5.8	Execution Time for Processing Patients with Minor Duplicates	63
	5.9	Execution Time for Processing Patients with Major Duplicates	63
	5.10	Cumulative Execution Time for Processing Number of Blocks	64
	5.11	Execution Time for Processing of Blocks per Minute	65
	5.12	Discussion	66
	5.13	Summary	67
6	**CONCLUSION AND FUTURE WORK**		**68**
	6.1	Overview	68

6.2 Summary of Research Work 68
6.3 Future Work 69

GLOSSARY **70**

REFERENCES **73**

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Summary of Deduplication-based Strategies	31
5.1	Simulation Parameters	56

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	Duplicated vs De-duplicated Data	2
1.2	Replication Factor	3
1.3	On-Premises Data De-duplication	6
1.4	Cloud Computing Architecture	10
2.1	Data De-duplication	15
2.2	IoT Cloud or Cloud of Things	21
3.1	Working Framework of the Research	40
3.2	Research design and development Plan for HDDS	42
4.1	Working Flow of Proposed Scheme	47
4.2	System Model	49
4.3	Flowchart of HDDS	50
4.4	Algorithm for Aggregating Vital Data	51
4.5	Algorithm for Removing Duplicate Data	52
5.1	Compressed patients data file size	58
5.2	Average Execution Time	59
5.3	Size of Data after Post Execution	60
5.4	Patient Data with Minor Duplicates	61
5.5	Patient Data with Major Duplicates	62
5.6	Execution Time for Processing Patients with Minor Duplicates	63
5.7	Execution Time for Processing Patients with Major Duplicates	64
5.8	Cumulative Execution Time for Processing Number of Blocks	65
5.9	Execution Time for Processing of Blocks per Minute	66

LIST OF ABBREVIATIONS

IoT	-	Internet of Things
CDC	-	Content-Defined Chunking
RF	-	Replication Factor
IaaS	-	Infrastructure as a Service
PaaS	-	Platform as a Service
FaaS	-	Function as a Service
SaaS	-	Software as a Service
VMs	-	Virtual Machines
PFP	-	Per-File Parity
DER	-	Data Elimination Ratio
JC	-	Jump-Based Chunking
RAM	-	Rapid Asymmetric Maximum
HDDS	-	Healthcare Data De-duplication Scheme
EFDS	-	Efficient File-Level De-Duplication Scheme
PEP	-	Per-File Parity
MLW	-	Message Locked Encryption
BP	-	Blood Pressure
ECG	-	Electrocardiogram
SHA	-	Secure Hash Algorithm
MD	-	Message Digest
CSP	-	Cloud Service Provider
EHR	-	Electronic Health Record
MLE	-	Multi-Level Encryption
HDS	-	Healthcare Data Repository
ECP	-	Elliptic-Curve Cryptography

ACKNOWLEDGMENT

First of all, I wish to express my gratitude and deep appreciation to Almighty Allah, who made this study possible and successful. This study would not be accomplished unless the honest espousal that was extended from several sources for which I would like to express my sincere thankfulness and gratitude. Yet, there were significant contributors for my attained success, and I cannot forget their input, especially my research supervisor, Associate Prof. Dr. Ata Ullah who did not leave any stone unturned to guide me during my research journey.

I shall also acknowledge the extended assistance from the administrations of Department of Computer Science who supported me all through my research experience and simplified the challenges I faced. For all whom I could not mention I thank all for their significant contribution.

DEDICATION

I would like to dedicate this thesis to my beloved family and teachers. They have been a source of encouragement and inspiration to me throughout my life and actively supported me in my determination to find and realize my potential, and to make this effort possible.

CHAPTER 1

INTRODUCTION

1.1 Overview

In this chapter, firstly the concept of data duplication and de-duplication is explained. Afterwards, the motivation behind this research is explained. Subsequently, the architecture and applications of data duplication are explained. After this, issues during the de-duplication process are discussed. Then, the background and main problem are explained. Furthermore, the research questions and objectives are stated. At the end, the thesis organization is stated.

1.2 Data De-duplication

The amount of digital data has increased immensely during the past few years. According to the statistics generated, the amount of data produced globally was 64.2 zettabytes in 2020 [1], and over the next 5 years, it is expected to reach more than 180 zettabytes. The actual data produced in 2020 was significantly higher than originally expected. On account of the Pandemic condition, more people worked and learned from home which as a result produced dramatically higher data volumes on the servers. Hence, how to oversee capacity cost-actually has become perhaps the most difficult and significant undertaking in this big data era.

De-duplication is an ideal method to oversee data duplication. It packs the information by removing copy duplicates of data. De-duplication decreases the extra room up to 90 to 95 percent [2] data transfer capacity rate and provides better storage management [3]. The procedure of de-duplication is tracking down various duplicates of similar data, keeping only one, and eliminating other copy duplicates [4] as shown in Figure 1.1.

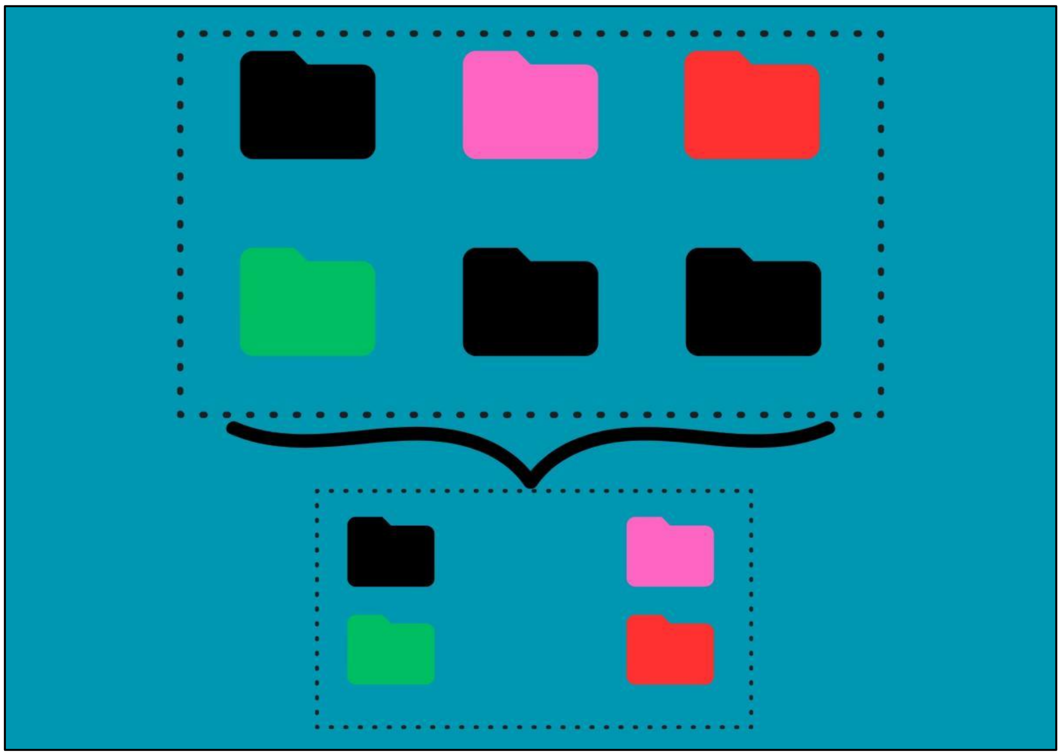

Figure 1.1: Duplicated and De-duplicated Data

1.3 Motivation

As a result of exponential data growth produced every year, a need to optimize data storage management techniques along with less de-duplication overhead has become the most prominent issue of the modern era. Getting along with technologies like the Internet of things (IoT) and SAAS products, it has become a challenge for cloud service providers to maintain, secure, and provide retrieval of data with less cost and transfer delays. Hence, in this work, the aim is to identify and optimize the Content Defined Chunking (CDC) approach to achieve variable-sized chunking with better DER to assist cloud service providers with better data management and to help out researchers and data scientists in their future work.

1.3.1 Architecture of Data De-duplication

Data de-duplication strategies are carried out based on the kind of data. The normally utilized data types are text, image, video, etc. Every data type has an individual and unique storage format and elements. The de-duplication procedure utilizes an alternate procedure for each kind of data to identify and dispense with copy duplicates [5]. Identifying and contrasting the data is troublesome assuming that the data is in various forms. Bit-level portrayal is utilized to play out the de-duplication interaction [6].

The Replication Factor (RF) is the base number of duplicates of similar data [7]. The Cloud storage system keeps a replication factor for all data. On the off chance that any data is more prominent than the replication factor, the de-duplication procedures and algorithms take out that data to decrease the capacity necessity, cost, and data transfer rate. An RF of one means there is only one copy of a row in a cluster, and there is no way to recover the data if the node is compromised or goes down. RF=2 means that there are two copies of a row in a cluster. An RF of at least three is used in most systems. However, most cloud systems do not support the complete removal of redundancy to have a backup or availability. Figure 1 shows that a client sends a request for a write operation to node A. As we have a Replication Factor (RF) of 3. It is replicated to the maximum of 3 of the nodes. However, node A is acting as a coordinator node, not a replicator node as shown in Figure 1.2.

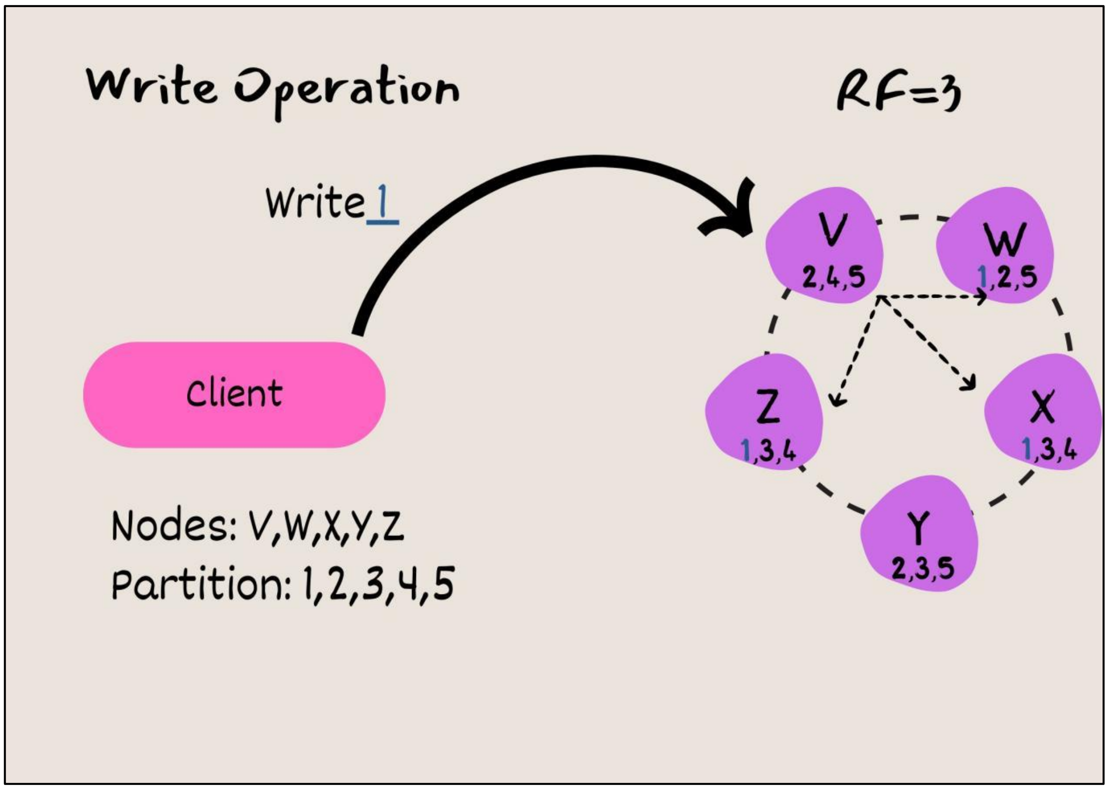

Figure 1.2: Replication Factor

Client-side and server-side de-duplication are the two approaches that can be used to remove duplicate data on storage devices. Server-side data de-duplication is a two-venture process in which copy information is recognized and afterwards, extra room is recovered to eliminate the copy information. Client-side information de-duplication stores the information straightforwardly in a de-duplicated format [8]. De-duplication can be implemented at the File level and Block level. The file level mainly focuses on the storage of a single copy of the file on the storage. As compared to Block level, generally, the four major steps involved in most block-level data de-duplication approaches are chunking, fingerprinting, indexing of fingerprints, and storage management [9].

1.3.1.1 Chunking:

Data de-duplication through chunking involves dividing the data into fixed-size or variable-size chunks, and identical chunks are stored only once, reducing redundancy [9].

1.3.1.2 Fingerprinting:

Fingerprinting generates unique identifiers (hashes or fingerprints) for data chunks, enabling quick comparison and identification of duplicate content without storing the actual data [9].

1.3.1.3 Indexing of Fingerprints and Storage Management:

Indexing involves maintaining a database of fingerprints and their corresponding locations in the storage. Efficient storage management relies on referencing this index to identify and eliminate duplicate chunks, optimizing storage space [9].

Most of the current de-duplication strategies are still lacking productivity in view because of the bad marks of data comparing, hashing, and matching algorithms as well as security vulnerability issues.

1.3.2 Advantages of Data De-duplication

Removal of redundancy of data is essential as it significantly reduces the cost of storage space and reduces how much bandwidth is wasted on data transfer to/from data access and storage locations. Therefore, it saves money. The application areas are referred to below:

i) Cloud-based storage providers

Cloud Storage providers like Amazon S3, Google Cloud Storage, Microsoft Azure, Alibaba Cloud, and Bitcasa are some specialist organizations embracing data de-duplication algorithms [10]. The aim is to assist in the improvement of such algorithms to achieve better storage management and cost-effective services.

ii) On-premises storage

The goal is to provide solutions to organizations who instead of using cloud services mostly rely on their on-premises storage mediums as shown in Figure 1.3. On the other hand, the aim is to also assist organizations who believe in performing de-duplication on data before storing it on the cloud to avoid bearing high costs in advance and storing it without duplicate copies [11].

iii) Save Storage Space

The one of the advantage of data de-duplication is keeping storage space safe. By finding and removing duplicate patterns of similar data, additional data files can be stored in a similar storage place, overcoming the requisite for extra storage capacity [12].

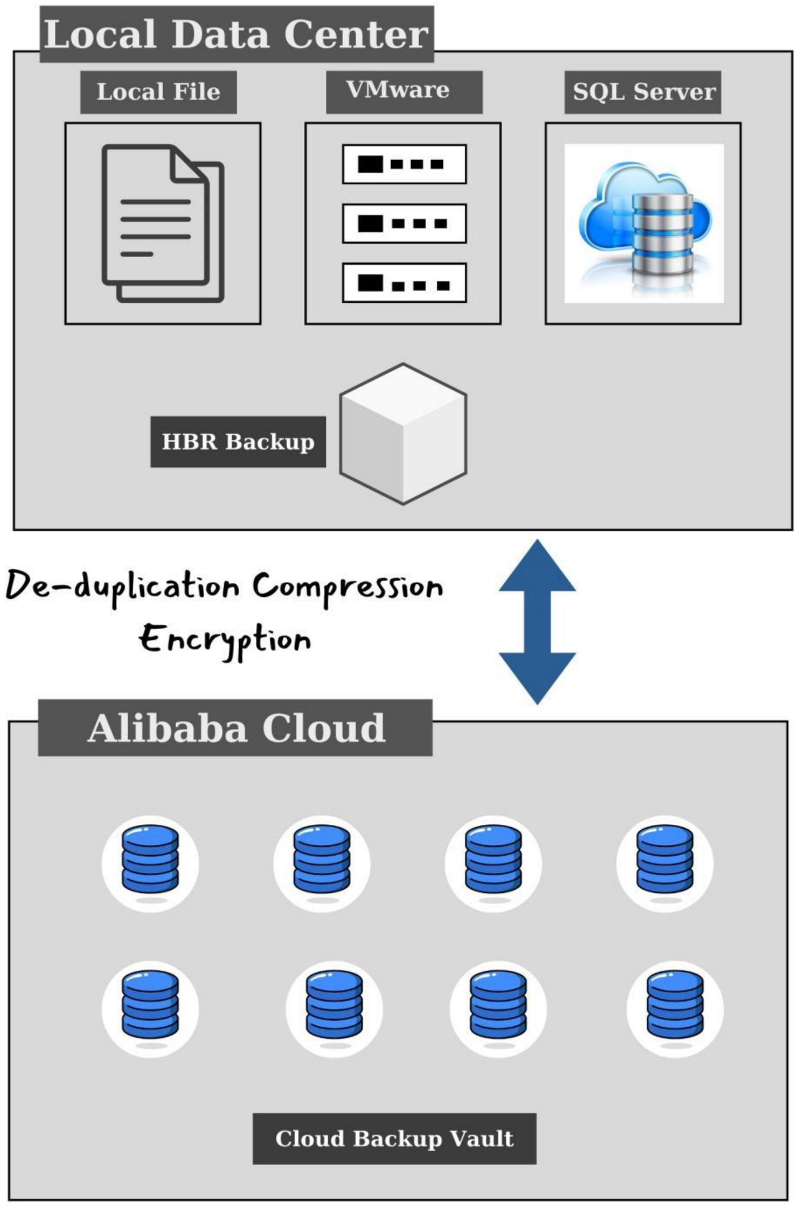

Figure 1.3: On-Premises Data De-duplication

iv) Reduction in storage Cost

Low storage needs assistance in saving cost, and needs low budget for maintenance purposes. As data remains high in volume it takes up a lot of space for storage. Besides this, data remains consistent and can be easily retrieved [13].

v) Better Backup and Recovery

In backup and data retrieval operations, it remains easy to process de-duplicated data as compared to duplicated one. So when duplicated files are reduced, maintaining backup remains easy and economical [14].

vi) Efficient use of Bandwidth

In case, the data is transferred to the cloud, de-duplication reduces the data volume which is required to be sent through the network. This results in decreased bandwidth utilization and lower network-associated costs [14].

vii) Faster Data Transfer

When identical files are removed, the process of transferring data becomes easier since only a unique data packet is sent or stored. Small data packets take less time to back up and enhance system performance [15].

viii) Increase Scalability

With de-duplication, high data is handled in storage space effectively as it keeps data growth efficiently without relative enhancement in hardware and operational costs [16].

ix) Lower Communication Cost

When sensor devices in a system generate replicated data, before transmitting in its original form, it is reduced which reduces communication cost and it makes de-duplication highly effective in IoT based environment [17].

x) Data Transfer Rate

De-duplication allows the transmission of data between various storage systems or clouds because one copy of data needs to be sent. Unique data blocks need to be transferred. It also assists in utilizing lower energy because fog servers transmit unique data to cloud repositories [17].

1.3.3 Constraints in the Data De-duplication Process

The data de-duplication process, while instrumental in reducing redundancy and optimizing storage efficiency, is not without its challenges and constraints. As organizations grapple with ever-expanding volumes of data, understanding and navigating the limitations of de-duplication mechanisms becomes crucial. This paper delves into the intricacies of the constraints inherent in the data de-duplication process, shedding light on factors that impact its effectiveness. From issues related to scalability and processing overhead to considerations of data security and the trade-offs between de-duplication techniques, a comprehensive exploration of these constraints is paramount for devising robust data management strategies. By identifying and addressing these challenges, organizations can enhance the efficacy of their de-duplication efforts and optimize resource utilization in the face of burgeoning data volumes. Data de-duplication has various advantages but some constraints still exist during this mechanism which is addressed below:

i) Performance issue

De-duplication can lead to higher computational overhead, particularly in real-time environments. The mechanism of checking data chunks to recognize repetitive data packets can slow down data processing and recovery tasks [18].

ii) Processing Capacity

The deduplication-based algorithms are exhaustive consume more resources, and need substantial processing energy to check, compare, and perform duplication removal from the dataset. So the need for powerful hardware cannot be ignored [19].

iii) Scalability issues

As data size increases, the procedure of de-duplication becomes further challenging. The Scalability issues exist and it needed to make use of some dispersed de-duplication clarifications or some sort of dedicated storage systems [20].

iv) Maintenance of De-duplication Key

Maintaining and managing keys for duplication-free data is vital for data access and from a security perspective. So it needs to be checked from time to time for its efficient management [21].

v) Files Fragmentation

The de-duplication procedure classifies data into a small number of chunks or blocks, which might increase the complexity of the data retrieval process, predominantly if any slice of the data is missing or corrupted [22].

1.4) Cloud Computing Architecture

Cloud computing is an innovative technology framework that permits users to use computing resources over the internet from anywhere [23]. The cloud computing architecture comprises of following entities as shown in Figure 1.4.

1.4.1) Front end Side

The front end is also known as the client side end which usually interacts with users or clients. The front end contains all applications that a user needs to access the cloud [14].

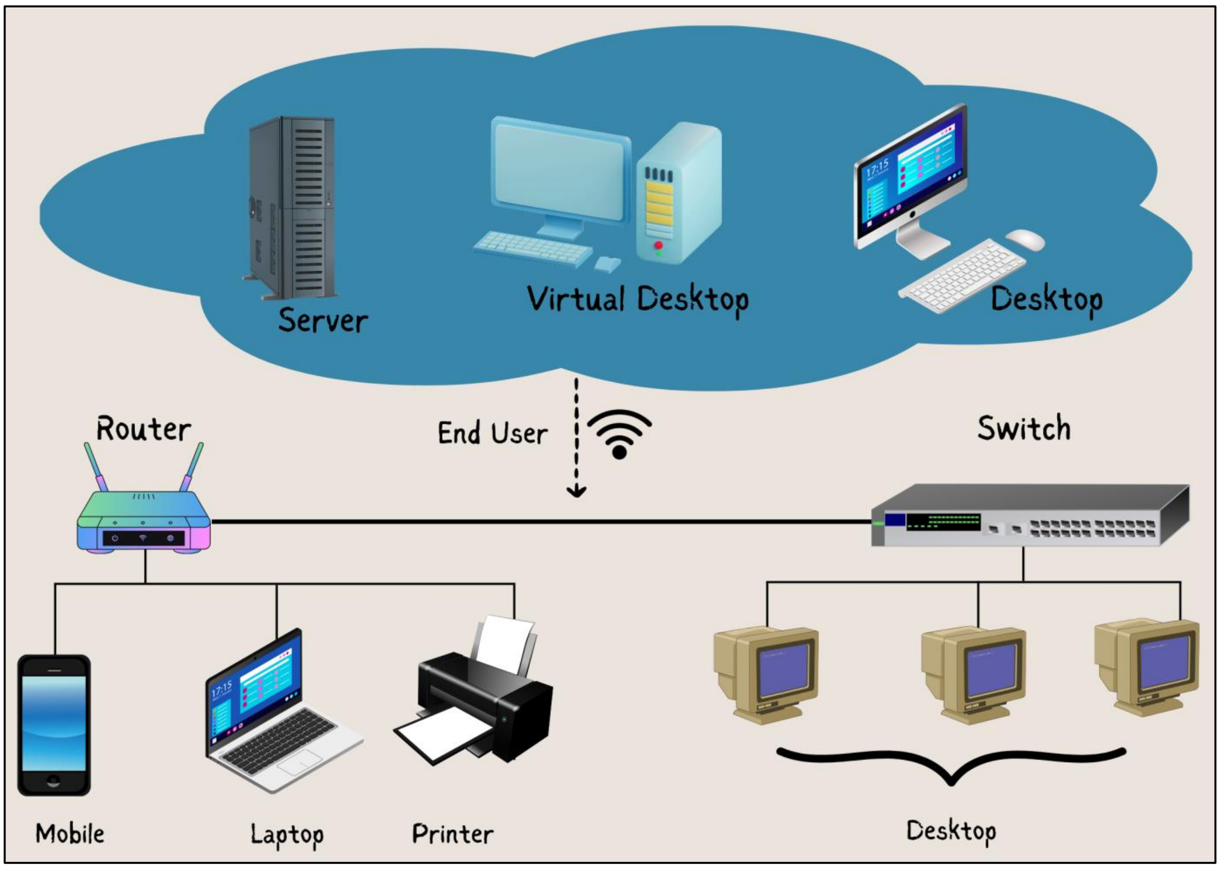

Figure 1.4: Cloud Computing Architecture

1.4.2) Back-end Side

The back-end side is well known as the server side. This Component of architecture can perform various functions including data storage retrieval, authentication, and verification operations.

1.4.3) Infrastructure

This component represents servers, storage devices, VMs, networks, and hardware components that are essential for providing cloud services.

1.5 Problem background

A hash chunk (fingerprint) is essential for the hashing, lookup, and matching stages in data de-duplication. The number of chunks in a de-duplication framework is typically very large to keep a hash table-based index structure in memory. Hence, either the size of the index table needs to be optimized or it needs to be stored on secondary storage, which affects the system efficiency, as the load factor of a reliable hashing index table normally needs to be very less to keep the lookup time limited. The problem arises when the chunk sizes go so large that they continue to have redundant data. On the other hand, if too small chunks are produced which eventually are converted to fingerprints (hashes) each chunk will have its hash value to be put into the lookup table. A challenge to have an optimized size of chunks along with minimum redundancy requires a spot on. The problem with smaller hashes is that they have a higher probability of collision. In other words, as the hash size decreases, the likelihood of two different pieces of data producing the same hash value increases. This can lead to data integrity issues, where different pieces of information are mistakenly identified as identical due to their hash collisions.

1.5.1 Negative Impact of the Problem

The main issues highlighted throughout the research study are described below. The problem arises can increase the load, have high storage cost, and lagged de-duplication process. These issues have bad impact and decrease the efficiency of the implemented approach.

i) High Storage Cost

When identical data chunks takes place in the server resultantly redundant data stores and take extra space. The numerous copies of the same data, each have its hash value, leads to higher storage requirements. The higher computational load and resource consumption related to byte pair comparison can also increase computational costs.

ii) Increased load

The process of using byte pair comparison in the whole byte stream proved to be computationally exhaustive. This shows that the system needs to do extra operations to check suitable cut points in the data. This mechanism has a higher computational load and overall slows down the de-duplication procedure. It significantly affects the overall efficiency of the system. The extra operations needed for byte pair comparison can lead to over-resource consumption, comprising CPU and memory consumption. Resultantly it has higher resource usage and could impact the system’s ability to tackle other tasks. The higher load produced by byte pair comparability leads to a slower de-duplication process. In cases where data de-duplication needs to be done fast, including in real-time or in resource-constrained environments, a slower de-duplication mechanism causes significant drawbacks. The extra operations needed for byte pair comparison can lead to over-resource consumption, comprising CPU and memory consumption. Resultantly it has higher resource usage and could impact the system’s ability to tackle other tasks [24].

v) Slow De-duplication Process

The higher overhead produced by byte pair comparability leads to a slower de-duplication process. In cases where data de-duplication needs to be done fast, a slower de-duplication mechanism causes significant drawbacks [25].

1.6 Problem Statement

In terms of IoT Data scenario, the data collected through sensor nodes continuously send large amount of data to the cloud server resulting in saving too much of redundant data. This redundancy leads to significant waste of storage space and increase communication costs. Existing approach [26] handles data redundancy using link list however the processing cost increases as soon as the link list start to grow.

1.7 Research Questions

This study focuses on the following research questions.

RQ1. What is the effect of replacing values into bits using larger group of health care data upon its size?

RQ2. How to reduce data transfer cost by creating data chunks of multiple nodes?

1.8 Aim of Research

Data de-duplication has been the most widely area to be worked on. Recently, due to an enormous amount of data, the performance of the data de-duplication process has been affected.

This research aims to highlight and overcome memory consumption while maintaining maximum throughput.

1.9 Research objectives

The given below objectives are stated to design and develop the process of data de-duplication.

- To minimize data size and data transfer cost.
- To manage and transfer data of multiple nodes collectively, in order to minimize time consumption.

1.10 Scope of Research

The idea behind this is to consider the identification of cut points in data. Also considering the size of the data stream that needs to be reduced while maintaining maximum throughput. The duplication in data is reduced to optimize storage, reducing communication cost and focusing on storing only crucial data of patients which subsequently improve healthcare services.

1.11 Thesis organization

The remaining thesis is organized as follows: Chapter 2 outlines current techniques based on Data De-duplication, their comparison, and major issues in this area. Chapter 3 illustrates the proposed techniques methodology. Chapter 4 illustrates details about the proposed scheme along with an explanation, Chapter 5 is about results and discussion of the proposed scheme, and in Chapter 6 conclusion and future work are added. In the end, references are given.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In this section, at the start, data duplication and de-duplication mechanism is described. Then, the concepts of cloud computing and the Internet of Things are described. Furthermore, existing literature on Data de-duplication in IoT-assisted Cloud Computing is discussed. A table is also presented that provides a comparison between schemes to enhance understanding. Finally, the whole chapter is summarized.

2.2 Data De-duplication

Data de-duplication is a technique used in data management and storage to eliminate redundant copies of data. This process reduces the overall volume of data so that it can be stored more efficiently, saving storage space and reducing costs. De-duplication works by scanning a set of data and identifying duplicate chunks of data. After the duplicates are found, they are removed, and a single copy is kept, often with a pointer to that data for any other instances where that data is used [27].

The best way to manage data duplication is using de-duplication. By eliminating copies of data, the de-duplicated information is stored [28]. De-duplication involves finding several copies of the same data, maintaining only one, and getting rid of the others as shown in Figure 2.1. De-duplication improves storage management, reduces unnecessary space by 90 to 95 percent, and increases the data transfer capacity rate [29].

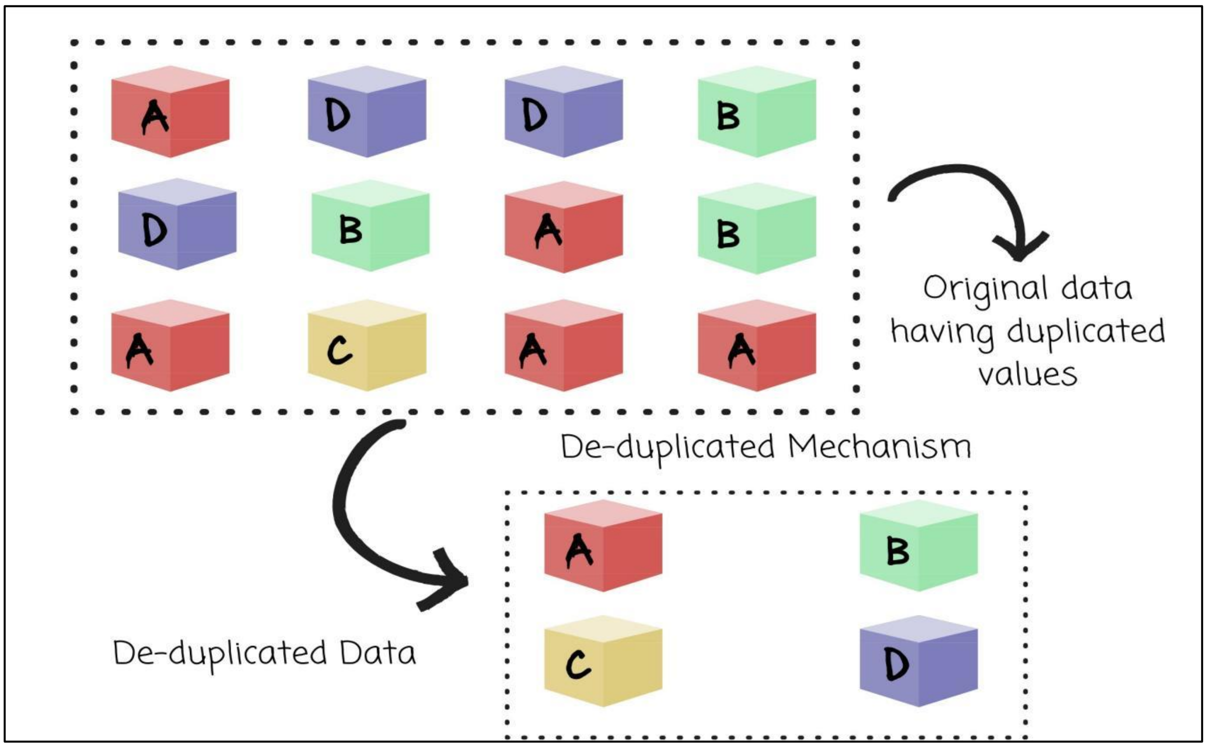

Figure 2.1: Data De-duplication

The exact mechanism for data de-duplication can vary. Some systems use file-level de-duplication, which compares entire files and removes duplicates. Others use block-level de-duplication, where files are broken down into blocks, and identical blocks are removed. Some systems even use a sub-block level or byte-level de-duplication, breaking data into even smaller pieces.

Data de-duplication is particularly useful in backup and archiving processes, where the same files or blocks may be stored multiple times. By keeping only one instance of each file or block, storage requirements can be greatly reduced. The data de-duplication requires processing power to analyze data and identify duplicates, and it can be computationally intensive depending on the amount and type of data. Therefore, the benefits in terms of storage space must be balanced to overcome the computational overhead of the de-duplication process.

2.2.1 Techniques of Data De-duplication

One of the techniques includes Content-defined chunking (CDC) which is an algorithm used for data de-duplication. The process divides data into chunks in a way that is dependent on the data content itself, rather than on arbitrary, fixed-size blocks.

This method helps to ensure that changes to data do not significantly affect the division of chunks. Consider, for instance, if small piece of data was added at the beginning of a large file. If fixed-size block de-duplication were used, this would shift all the blocks, and all of the subsequent blocks would be considered new, even though most of the data is identical. In contrast, with content-defined chunking, the boundaries between chunks are determined by the data itself, so a small change would only affect a few chunks near the change [30].

The chunks in CDC are usually determined by identifying 'breakpoints' in the data. A common method involves applying a Rabin fingerprint (a type of rolling hash function) to a sliding window of bytes and choosing breakpoints where the fingerprint meets certain criteria. The advantage of this method is that it is fast and doesn't require knowledge of the data structure. The result is a set of variable-length chunks that are well-suited to data de-duplication, since small changes to the data will only change a few chunks, rather than the entire file or block of data [31]. The CDC can be more efficient for de-duplication purposes, it can also be more computationally intensive to determine the chunk boundaries, compared to fixed-size block de-duplication

2.3 Cloud Computing

Cloud computing is the on-demand delivery of IT resources over the Internet. Instead of buying, owning, and maintaining physical data center and servers, businesses can access technology services, such as computing power, storage, and databases, from a cloud provider [32]. There are several types of cloud computing which are described below.

2.3.1 Infrastructure as a Service (IaaS)

This is the most basic category of cloud computing services, which allows to rent IT infrastructure servers, virtual machines (VMs), storage, networks, and operating systems from a cloud provider on a pay-as-you-go basis [33].

2.3.2. Platform as a Service (PaaS)

Platform as a service refers to cloud computing services that supply an on-demand environment for developing, testing, delivering, and managing software applications. PaaS is designed to make it easier for developers to quickly create web or mobile apps, without worrying about setting up or managing the underlying infrastructure of servers, storage, network, and databases needed for development [34].

2.3.3. Software as a Service (SaaS)

In SaaS, the service provider delivers software and applications through the internet. These are usually provided on a subscription basis and are centrally hosted. Examples include email and collaboration software, customer relationship management (CRM) software, and virtual meeting software [35].

2.3.4. Function as a Service (FaaS)

FaaS, or server-less computing, allows developers to execute portions of application code (functions) in response to events. It's called server less as the business that owns the system does not have to purchase, rent, or provision servers or virtual machines for the back-end code to run on [36].

2.4. Deployment Models

The deployment model refers to the potential ways that the services of Cloud computing are executed and utilized. It also informs about the access level and management mechanism of data. Based on its features and provided functionality, three different deployment models are described below in detail.

2.4.1 Public Cloud

The public cloud deployment method is described as the services are owned and operated by third-party cloud service providers. The third-party provides resources such as computing resources and storing data online on a cloud repository. These services are shared among different organizations and customers. This model is cost-efficient and economical. However, this deployment model is not considered highly strong and may have some security breaches. So intruders might get data after some malicious attacks [37].

2.4.2. Private Cloud

A private cloud refers to cloud computing resources used exclusively by a single business or organization. A private cloud can be physically located at the company's on-site data center, or it can be hosted by a third-party service provider. It is considered highly secured and data is maintained in it easily [38].

2.4.3 Hybrid Cloud

Hybrid clouds are a combination of public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, a hybrid cloud gives businesses greater flexibility, and more deployment options, and helps optimize existing infrastructure, security, and compliance [39].

Benefits of cloud computing include cost-efficiency (no need to invest heavily in owning and maintaining servers), speed (most cloud computing services are provided self-service and on-demand), scalability (services can be scaled up or down to fit needs), productivity (removes the need for many IT management chores), performance (benefit from massive economies of scale), and reliability (data can be mirrored at multiple redundant sites on the cloud provider’s network) [40].

2.5 Internet of Things

The Internet of Things (IoT) is a system of interrelated physical devices, vehicles, buildings, and other items embedded with sensors, software, network connectivity, and necessary electronics that enable these objects to collect, exchange, and act on data [41]. The "things" in IoT often refer to devices that wouldn't ordinarily have internet connectivity capability, thereby allowing them to generate and exchange data with a network or other devices. These connected devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices. Current market examples include home automation also known as smart home devices such as the control and automation of lighting, heating, ventilation, and air conditioning systems, and appliances such as washers/dryers, robotic vacuums, air purifiers, ovens, or refrigerators/freezers that use Wi-Fi for remote monitoring [42]. The internet of things is also a key component of home security systems, wearable technology devices, connected cars, industrial applications, and more. It's increasingly being used in environments like urban planning and management, healthcare, and agriculture.

The concept of IoT promises to revolutionize our lives and the way we interact with our environment by automating routine tasks, optimizing resource usage, and providing new insights and services. However, it also presents challenges in terms of privacy, security, interoperability, power efficiency, and data management [43].

2.6. IoT-assisted Cloud Computing

Internet of Things (IoT) assisted cloud computing refers to the integration of IoT devices with cloud computing infrastructure. This convergence enables data from IoT devices to be easily stored, processed, and accessed in the cloud, and it allows for powerful computation, analytics, and machine learning tasks to be performed on this data. Here's a more detailed explanation of each component [44].

2.6.1. Internet of Things (IoT)

This term refers to the network of physical objects ("things") that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. These devices can range from everyday household items like smart fridges and thermostats to industrial IoT devices like sensors monitoring temperature, pressure, or humidity in an industrial site [45].

2.6.2. Cloud Computing

This is the delivery of different services through the Internet, including data storage, servers, databases, networking, and software. Rather than keeping files on a proprietary hard drive or local storage device, cloud-based storage makes it possible to save them to a remote database [46]. When these two concepts are combined, it's often referred to as "IoT-assisted cloud computing" or "Cloud-based IoT". In this model, data generated by IoT devices is transmitted to the cloud, where it can be processed and analyzed. This allows for real-time processing and analytics, which can lead to more effective decision-making [47]. For example, an IoT-assisted cloud computing application could involve a network of sensors in a manufacturing plant that sends data about machine operation to the cloud. Therefore, the data can be analyzed to predict when a machine is likely to fail, allowing for proactive maintenance [48].

The combination of IoT and cloud computing also allows for easier scalability, as new IoT devices can be added to the network without needing significant upgrades to storage or processing infrastructure since the cloud can easily scale to accommodate the extra data [49]. The IoT-Cloud or Cloud of Things is presented in Figure 2.2.

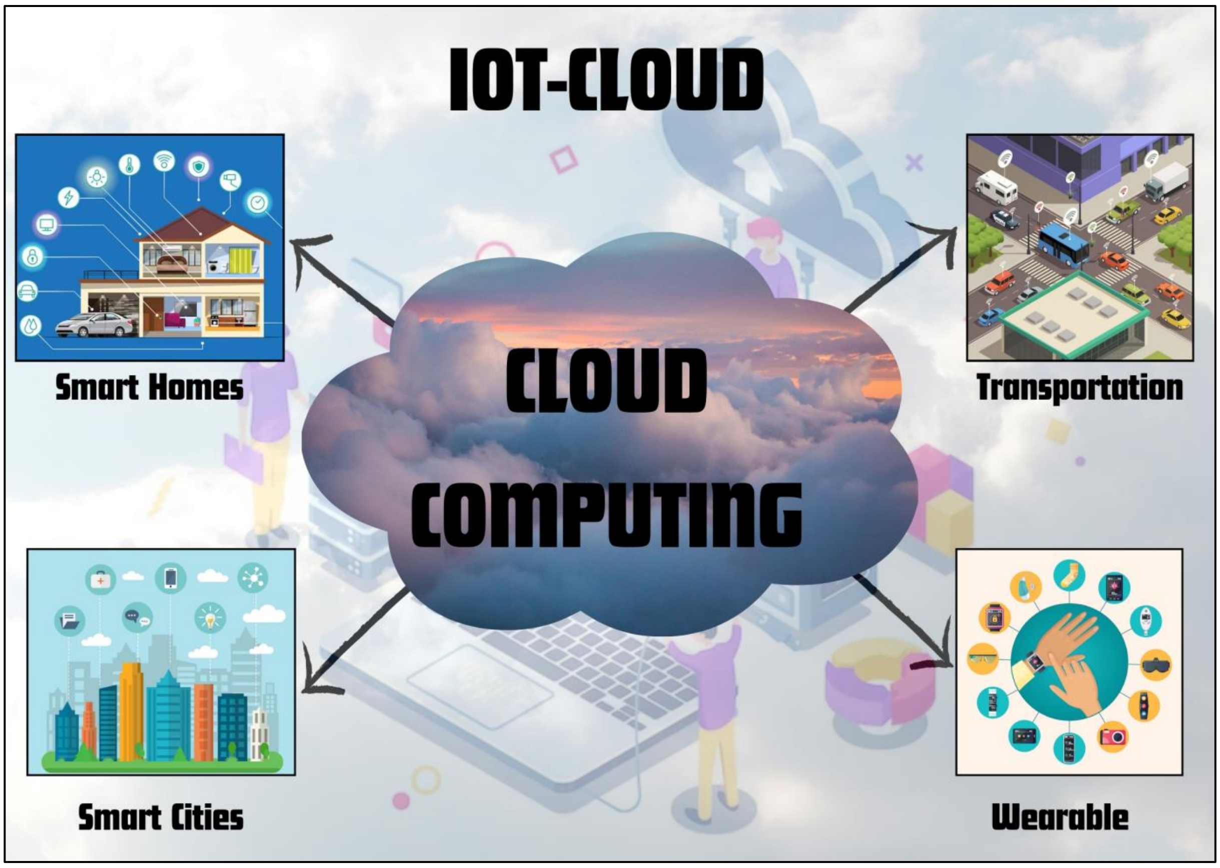

Figure 2.2: IoT Cloud or Cloud of Things

2.7 Data De-duplication in IoT-assisted Cloud Computing

Data de-duplication is a process that eliminates redundant copies of data and reduces storage overhead. In the context of IoT-assisted cloud computing, data de-duplication plays a crucial role due to the enormous amount of data generated by IoT devices [50]. IoT devices, such as sensors, smart appliances, wearables, etc., continuously generate data that gets sent to the cloud for processing and storage. Given the sheer volume of IoT devices and the frequency at which they generate data, there can often be a lot of redundancy. This redundancy could be due to multiple devices recording similar information, or a single device recording the same data point over time when no change has occurred [51]. Data de-duplication in this context would involve analyzing the incoming data from these devices and removing any duplicates before storing it in the cloud. This process can significantly reduce the amount of data that needs to be stored, leading to cost savings in storage expenses, improved efficiency in data transmission and processing, and quicker data retrieval times. It can also enhance the performance of data analytics by reducing the volume of data that needs to be processed [52].

There are different methods of data de-duplication, such as file-level de-duplication (where entire duplicate files are removed), block-level de-duplication (where duplicate blocks of data within a file are removed), and byte-level de-duplication (where duplicate bytes of data are removed). The choice of method can depend on various factors such as the nature of the data, the specific requirements of the system, and the resources available [53]. However, data de-duplication has many benefits, but it also has certain challenges and considerations, such as ensuring data integrity, managing the de-duplication process efficiently, and maintaining the security and privacy of the data [54].

2.8 Existing Studies in Data De-duplication in IoT-assisted Cloud Computing

In recent years, the proliferation of digital data has been immense. Global data production surged to 64.2 zeta-bytes in 2020 and is forecasted to surpass 180 zeta-bytes in the next half-decade. The COVID-19 pandemic, which spurred widespread remote work and learning, greatly contributed to this data explosion, outpacing previous estimates. As a result, effective management of storage costs has become a significant challenge in the era of big data [55]. In response to this data boom, there's been a surge in the exploration of data de-duplication frameworks. Given the vast data volumes, perfecting processes like chunking, fingerprint generation, and fingerprint lookup has become critical. These steps often pose a bottleneck to the adaptability and throughput of de-duplication frameworks. Previous research has provided numerous insights and improvements into these processes [56].

Chunking is a particularly critical step that impacts the hash table size directly. Chunks can be either fixed or variable in size. Research suggests that variable-sized chunks can uncover more duplicates and address the boundary shift issue, thereby enhancing de-duplication efficiency. However, data De-Duplication concerning Cloud of Things is still infancy. So, this chapter specifically defines the existing literature relevant to data de-duplication in the context of IoT-Cloud [57]. On the basis of data de-duplication, the de-duplication mechanism is categorized into two levels: i) File Level Data De-duplication ii) Block Level Data De-duplication

The author in [58] introduced a new parallel chunking method using the PCI algorithm. This algorithm does not include hash functions but works well with hardware implementation. The proposed system not only achieves high computational speed but great scalability as well.

2.8.1 File Level Data De-duplication

In this category, data de-duplication is performed at file level. A file is matched with other files and identifies similar data between them. Then, redundant copies of data are deleted. The schemes of this category are presented comprehensively as following.

This paper [59] presents a new method for content-defined chunking (CDC) called Rapid Asymmetric Maximum (RAM), designed to increase the efficiency of data de-duplication systems. The proposed RAM algorithm eschews the use of hashes, which are often computationally expensive and instead utilizes the byte values to determine the cut points of chunks. The algorithm employs a fixed-size window and a variable-sized window to locate a maximum-valued byte, which is then designated as the cut point.

In SD-IoV, the sensor nodes continuously transmit redundant data to the cloud server which increases duplication communication, and storage overhead. To overcome this issue, the proposed scheme EFDS uses a hash table for the identification of similar data. The data files are transmitted to the cloud through the SDN controller. The SDN controller converts files into small blocks. For each sub-block, the Rabin fingerprint is calculated and checks a hash table that already comprises the same fingerprint or not to determine identical data. If no fingerprint exists, it shows that the data is not identical and is saved in the available index. In the second case, the data file is included in the linked list created for duplicate blocks at the corresponding index of the hash table. This index determines the fingerprint of duplicated data. The EFDS ensures efficient resource utilization but the drawback of this is that the problem is that with time when the number of duplicates increases, it also increases the size of the duplicate reference list. ultimately time needed for maintaining a link list and lookup also increases [26].

This maximum-valued byte is included in the chunk and situated at the boundary of the chunk. This unique configuration reduces the number of comparisons necessary, thus improving computational efficiency while maintaining the desirable properties of CDC. When compared with existing hash-based and hashes de-duplication systems, RAM demonstrates higher throughput and saves more bytes per second, marking it as a promising method for improving the efficiency of data de-duplication. This article proposes FastCDC, a new and efficient approach to Content-Defined Chunking (CDC), aimed at improving performance in data de-duplication systems. Traditional CDC methods tend to cause high CPU overhead as they compute and judge rolling hashes of the data stream byte-byte. FastCDC addresses this inefficiency through the combined application of five key techniques: gear-based fast rolling hash, simplifying and enhancing Gear hash judgement, skipping sub-minimum chunk cut-points, normalizing the chunk-size distribution within a specified region, and rolling two bytes at a time to increase CDC speed. These techniques result in FastCDC being between 3 to 12 times faster than existing CDC methods while achieving a similar, if not higher, de-duplication ratio as the classic Rabin-based CDC. Furthermore, when implemented in Destor, an open-source de-duplication project, FastCDC increases the de-duplication throughput by 1.2 to 3 times compared to Destor based on state-of-the-art clunkers. Overall, the proposed FastCDC method represents a significant advancement in the efficiency and speed of data de-duplication systems [60].

To overcome storage cost by employing a data de-duplication mechanism in IoT-assisted cloud and to increase security, Rasina et al. has presented a scheme that employs a Residue Number System (RNS) for a key generation mechanism. The secret key is divided into reduced, independent portions that can work simultaneously. It splits a secret key into smaller, independent parts that can be processed concurrently, so it becomes difficult for adversaries to detect the entire key. The produced keys use Elliptic Curve Cryptography for encryption and decryption mechanism, an extremely effective public-key encryption mechanism that provides high security with reduced keys, overall lowering computational overhead. For de-duplication, the technique includes a de-duplication process by combining the idea of RNS with Cosine based Similarity checking. This mechanism proficiently detects identical data, decreasing storage space and improving resource utilization in the cloud. For data integrity, a hash key verification mechanism is introduced. In case data is uploaded to the cloud, a hash value is generated. While downloading, a hash is checked to authenticate data. If the hash remained the same it depicts that no tamper attack is done on it. Upon downloading data from the cloud, the hash value is checked to verify the data's integrity. If the hash values remain the same, the data is considered authentic and has not been tampered [61].

In the same context of resolving duplication issues in a cloud environment while maintaining security and data privacy. Darong et al. make use of convergent encryption that assists in generating encrypted keys based on data. This technique helps in the detection of identical data and reduces storage costs by eliminating it. This approach facilitates the identification of duplicate data by utilizing a common label, which simplifies the detection process and significantly reduces storage overhead. In essence, it ensures that redundant copies of the same data are eliminated, optimizing storage resources in cloud environments. Besides this, the bloom filter is used for verification of users trying to access data, and only authorized can access it. The advantage of the scheme is that de-duplicated data is only presentable to legal users. The limitation of the scheme is that while performing cryptographic operations, delays can occur particularly when dealing with large volumes of data [62].

Ahmed Sardar M. Saeed et al. put forward a novel proposition aimed at enhancing certain parameters in Content Defined Chunking (CDC). CDC is a method in which dynamic or variable-sized segmentation is performed, pivoting on the internal characteristics of the data content. Their innovation was directed towards elevating the Data Elimination Ratio (DER) in de-duplication processes. In this method, chunk boundaries are discerned based on the recurrence rate of each byte pair. This ensures that alterations in one chunk do not ripple out to impact other chunks, confining the effect within the initial chunk. During the procedure of determining byte pair frequency, an analysis is carried out at the character or byte level. Along with this, the discussed research also provided a comparative analysis of their results with other CDC-based algorithms, such as the BSW and TTTD algorithms, illustrating the effectiveness of their approach. The impact of hash algorithms on storage size was assessed by calculating the Hashing index table size, given by the formula: Number of Bytes per Fingerprint × Total

Number of Chunks after de-duplication. The limitation of this approach lies in the fixed production of five hashes per chunk, which could be excessive for particularly small chunks, potentially leading to inefficiencies [63].

Subsequently, Ahmed Sardar M. Saeed et al. proposed another solution aimed at reducing the size of the index table. This time, rather than concentrating on chunking, they introduced a mathematically bounded linear hash function to create hashes of chunks. The proposed approach resulted in a significant improvement, accelerating hashing times by more than double compared to MD5 and SHA-1, and reducing the hash table size by 50%. This strategy employs a hierarchical hash-matching procedure. Each chunk generates five hash values with fewer bits (i.e., 16 bits) compared to the larger SHA-1 (160 bits) and MD5 (128 bits). This involves a multilevel lookup process where an incoming chunk is first compared based on size. If the chunks are of identical size, then the first hash is compared, followed by the second, and so on. If all five hashes align, the match is considered successful, obviating the need for byte-to-byte comparison [64].

This paper [65] introduces a novel jump-based chunking (JC) approach for data de-duplication systems aimed at improving throughput and reducing CPU overhead. Traditional content-defined chunking (CDC) methods, which calculate rolling hashes of the input data stream byte, can be computationally expensive and significantly degrade system throughput. The proposed JC approach challenges the necessity of this byte-by-byte calculation. The innovative feature of JC is the introduction of a jump condition, which allows the sliding window to skip over specific lengths of the input data stream if the rolling hashes satisfy this condition. This paper also examines the impact of the cut condition and the jump condition on chunk size. Theoretical studies and experimental results demonstrate that JC not only enhances efficiency and reduces CPU overhead, but also maintains the high de-duplication ratio crucial for effective data de-duplication. The results show JC improves throughput by approximately two times on average compared with traditional CDC methods.

The paper presents SE-PoW, a novel system for data de-duplication in IoT networks. This system enhances security through a dual-level Proof-of-Ownership algorithm and dual encryption techniques. It improves system efficiency, reducing upload time overhead by up to 61.9% compared to existing methods [66].

The author in [67] introduces a new multi-user up-datable encryption scheme for secure de-duplication that allows efficient user revocation. By allowing the data owner to update the remote cipher-text with an update token, this method reduces communication and computation costs, while maintaining high security and efficiency.

The researcher [68] discussed InDe, an in-line de-duplication system, which uses the F-greedy and F-greedy+ rewrite algorithms to reduce data fragmentation and improve restore speeds. For determining average rewriting data chunks, the Equation 2.1 is used. N_{rw} represents previous the total number of data chunk in previous history of backup. By dynamically adjusting the number of old container references, InDe improves restore speeds by 1.3x - 2.4x while maintaining similar backup performance.

$$H_{rw} = \frac{N_{rw}}{total\ segemnts} \qquad (2.1)$$

The paper [69] introduces the Per-File Parity (PFP) scheme to improve the reliability of deduplication-based storage systems by providing parity redundancy protection for each file. PFP outperforms existing solutions in system reliability, with a minor 5.7% performance degradation. The primary objective of their study was to devise methods for reducing the number of hash chunks stored in memory, while also enhancing the efficiency of the hash table lookup process. They also focused on accommodating variable-sized chunks, which can impact the size of the hash table.

In [70], the proposed SDD-RT-BF scheme aims to provide data security and an effective de-duplication mechanism in a distributed environment. The proposed scheme model comprises three important phases including authorized de-duplication, ownership proof, and updating key. When the developer of the file uploads the data file to CSP, a unique token is created for the file which is used for verification purposes. The file is divided into smaller blocks and encrypted by using a protection key and transmitting an authentication request to a Master Controller (MC). MC, which manages a Radix Trie (RT), creates a role key and sends it to U1 for re-encryption of data. The second phase involves proof of ownership as it deals with the security of data. A user provides its token and identifier to CSP, which confirms its validity. If the user is legal, CSP picks block indexes and directs them to the user. The third and last is the role key update that deals with the dynamic updating of user rights. The proposed scheme has a lower computational time and higher de-duplication ratio.

The proposed scheme QuickCDC skip similar data packet that considerably increases chunking speed. By checking and avoiding identified duplicate packets, QuickCDC eliminates the time needed for exhaustive handling, making it for datasets with low duplication. It integrates regulating mask bits dynamically to increase chunking precision and ratio of data de-duplication. These mask bits assist in improving chunk borders that are correctly associated with the data, avoiding issues that overcome de-duplication precision. By dynamically adjusting mask bits, the error jumps are minimized. Error jumps refer to the incorrect chunking boundaries of data packets. The advantage of the scheme is that the de-duplication mechanism is fast and error jumps are low. The dynamic adjustment of mask bits requires extra memory [71].

The scheme ensures data security in a distributed environment by using cryptographic hashes and detecting duplicate data in the data stream. Convergent encryption (CE) is utilized to detect identical data whenever data is uploaded. A tag T is produced based on data. Whenever the same data is again observed in the data stream, the tag for that data will be the same showing duplication. This duplicated data is not again transmitted to ensure a de-duplication. To maintain privacy, CEK mechanism is introduced which ensures that only legal users can decrypt and have access to the data. The encrypted key is updated so no intruder can access this data [72].

The scheme is a three-tier data infrastructure to store de-duplicated data in a cloud repository. The proposed technique uses four main entities which are KGC, CS, DO, and DU. KGC provides key pairs to authenticated users for security purposes. Cloud server is responsible for storage and retrieval of data. Data owners can upload data anytime and authorized users can access it. The data stream is split into chunks by utilizing the cuckoo algorithm. When an owner intends to store a data file, firstly it is checked for duplication in the server. For this purpose, Merkle Hash tree is utilized in which hash values are computed for data blocks. if the same data exists there, the system indicates it by placing a pointer, else it is saved in encrypted format. The scheme considers dividing data chunks in an optimum as per the context of data [73].

To overcome data duplication in medical records at a large scale, this approach uses a rewriting scheme based on similarity. When data is stored, the upcoming data is matched with the previous one, and chunks that are different from the previously stored data are stored while similar data chunks are removed. The fragments created during de-duplication are reduced. To provide security from unauthorized access, blockchain technology is employed. The content of the file is determined by using a combination of fingerprint and offset. These contents are stored in the blockchain FISCO BCOS system. By storing data readings on the blockchain platform, data integrity is assured. Any malicious or tampering attack is identified by auditing. To increase the speed of the de-duplication mechanism, the concept of parallelization is introduced. The concepts of multi-core and multi-processor are utilized to process data for de-duplication. Overall, ESDedup has a better and faster mechanism for de-duplication but complexity increases as the environment diversity increases and it is costly to deploy [74].

In the healthcare sector, to overcome duplicate ratios in stored data, Xiao et al. [75] presented a scheme in which semantic awareness analysis is performed to eliminate redundancy. The data features are analyzed and checked for similar data within the same file contents and in different files. To find the similarity between two chunks, Equation 2.2 is used.

$$H_1(C_{i,}^{k}, C_{j,}^{q}) = \begin{cases} h_1 C_i^k \cap, C_{j,}^{q} > h \\ 1, C_i^k = C_j^q \end{cases} \qquad (2.2)$$

C_i^k Shows the first data chunk and C_i^k represents the second data chunk. $C_i^k \cap, C_{j,}^{q}$ represents an intersection between two blocks for similar data content. The data having a high volume of redundant data is compressed by using the LZ compression approach. For low-ratio duplicate values, the Merkle tree mechanism is used. For security and maintaining patient data privacy, an auditing framework is employed. The scheme has low storage costs but takes a long time to process huge volumes of data.

2.8.2 Block-Level Data De-duplication

Block-level schemes use the mechanism to compare data within the blocks, sub-blocks, and other stored blocks for the identification of duplicated values. This category is considered more effective than file-level data duplication.

To overcome duplication, data is transmitted to the cloud repository. In this scheme, data owners create and upload data to the cloud repository. The cloud service provider (CSP) creates hash codes along with an index table by using a scalable index table technique. It lowers false positive rates by recognizing identical data chunks. When data rises in volume, the index table increases its size to handle more data, keeping a minimum false-positive ratio because of its huge size. Furthermore, the TL-CH technique adopts a procedure of dividing the byte stream to create exclusive signatures, reducing the time to match and overcoming data collision issues commonly raised in traditional hash functions including MD5 and SHA-1. Additionally, to enhance data security, an improved map chaotic data encryption algorithm is used for encrypting healthcare data. This encryption mechanism comprises chaotic systems and dynamic sequences to considerably enhance the key space and avoid adversary attacks [76].

In the technique HealthDep, de-duplication is assured and health records are securely maintained. To optimize data management, the data is divided into two categories which are sensitive medical data and less sensitive medical data. The highly sensitive data is encrypted by employing conventional encryption methods. In the case of less sensitive health data, Multi-Level Encryption is used. This data is then transmitted to the database for storage, where data is verified and checked for duplicated instances. After detecting duplicating chunks, unique data records are stored which reduces data volume and enhances storage efficiency. But to implement this approach, extra resources are required which makes the technique costly to deploy [77].

The scheme intends to maintain data integrity and privacy while overcoming the de-duplication ratio in data. For de-duplication purposes, the client lightweight component C is used which communicates with a cloud service provider and index service. Three cases are considered in this scheme which includes popular, and unpopular data upload and popularity-based transition. For all cases, data popularity is checked, and based on outcomes, action is performed. For popular data blocks, associated IDs are stored in a hash table. From a security perspective, data with low privacy is protected using CE encryption and for high privacy data, SE is used. The scheme has achieved a high de-duplication ratio but if the entity like index service (IS) is compromised, then user's data can be damaged [78].

2.9 Comparison of Data de-duplication in IoT-assisted Cloud Computing-based Strategies

In this section, the comparison of existing techniques is presented with their contribution, advantages, and limitations. A summary of current research on data de-duplication with file and block level de-duplication is presented in Table 2.1.

Table 2.1: Summary of Deduplication-based Strategies

Scheme	**Basic idea**	**Mechanism**	**Advantages**	**Limitations**
File Level Data De-duplication				
RAM [59]	Uses Content-Defined Chunking Technique	This paper proposes a high-throughput hash less chunking method called Rapid Asymmetric Maximum (RAM)	The proposed algorithm has higher throughput and bytes saved per second compared to other chunking algorithms.	Although the proposed algorithm shows promising results, it may not perform equally well on all types of datasets or environments.
FastCDC [60]	Use of five key techniques, namely, gear-based fast rolling hash, simplifying and enhancing the Gear hash judgement, skipping sub-minimum chunk cut-points, normalizing the chunk-size distribution	This study proposes, a Fast and efficient Content-Defined Chunking approach, for data de-duplication-based storage systems.	FastCDC is 3-12X faster than the state-of-the-art CDC approaches, while achieving nearly the same and even higher de-duplication ratio as the classic Rabin-based CDC.	These results may not necessarily hold in all settings.

Improvised-ECC-IRNS [61]	Improve Data Storage In a Cloud-Based IoT Environment by Resolving Duplication.	The RNS method is utilized for key generation, cosine method is used for similarity in the data.	Increases security, overcome data storage requirement, and reduces data duplication ratio.	The Cosine mechanism for checking duplication is not reliable in case of a complex environment.
Bloom Filter based de-duplication scheme [62]	Reduce identical data while maintaining privacy.	To resolve identical data, the Bloom filter is used. By employing a cryptographic signature and validation mechanism, access to data is granted only to legal users.	Fast data retrieval. Provides higher security to data. Reduced storage cost.	Not appropriate to use in a resource-constrained environment. Complexity increases as data grows.
BFBC [63]	Chunking based on the frequency of a pair of bytes	This research presents a novel bytes frequency-based chunking (BFBC) algorithm that optimizes data de-duplication performance.	The proposed triple hash function algorithm is five times faster than SHA1 and MD5 and achieves a better de-duplication elimination ratio (DER) than other CDC algorithms.	Comparison at character level has expensive computation cost.
Multi-Hash Function [55]	Multiple Hashes of a single chunk	This study introduces a novel and effective mathematical bounded linear hashing algorithm specifically designed for use in data de-duplication systems.	Our suggested system reduces the high latency imposed by de-duplication procedures, primarily the hashing and matching phases.	Due to the enormous number of chunk hash values, looking up and comparing hash values takes longer for large datasets.
JC Scheme [65]	Content Define Chunking using input data stream byte by byte	This study proposed a jump-based chunking (JC) approach.	.JC improves the throughput by about 2× on average compared with the state-of-the-art CDC approaches while still guaranteeing a high de-duplication ratio	Complexity of scheme makes it difficult to implement.

SE-PoW [66]	Message-locked encryption (MLE) to encrypt data.	The paper introduces SE-PoW, a novel efficient, and location-aware hybrid encrypted de-duplication scheme with dual-level security-enhanced Proof-of-Ownership (PoW) in edge computing.	SE-PoW reduces up to 61.9% upload time overheads compared with existing methods, which represents a significant performance enhancement.	The proposed system is complexed in terms of implementation
Secure de-duplication with efficient user revocation in cloud storage [58]	A multi-user up-datable encryption scheme.	The paper proposes a multi-user updatable encryption scheme that allows the data owner to update the remote cipher text under a new group key by sending an update token to the cloud.	Secure de-duplication requires the data owner to download, decrypt, re-encrypt, and upload data to the cloud when updating data authority.	The proposed scheme significantly reduces communication and computation costs as the data owner only needs to send a token to the cloud to update the data authority.
InDe [68]	A greedy algorithm to detect valid container utilization	The paper contributes an innovative system, InDe, which addresses data fragmentation in in-line de-duplication.	The proposed InDe system improves the restore speed by 1.3x - 2.4x compared to two state-of-the-art algorithms (Capping and SMR) while maintaining similar backup performance.	Complex in terms of implementing the new algorithms, F-greedy and F-greedy+.
PFP [69]	A data de-duplication chunking technique	The paper proposes the Per-File Parity (PFP) scheme, a new approach to improve the reliability of deduplication-based storage systems.	The Per-File Parity scheme improves the reliability of deduplication-based storage systems by providing parity redundancy protection for all files.	One limitation could be the average 5.7 percent performance degradation to the deduplication-based storage system.

EFDS [26]	By employing hash table approach, duplicate data generated by sensors is identified and eliminated.	Rabin fingerprints are calculated and a hash table in maintained. For duplicate sub files, linked list is used, The de-duplicate data is transmitted to cloud repository.	Resource consumption is low. Efficient identification and handling of duplicate data.	For large data volume, performance starts degrading.
SDD-RT-BF [70]	Utilizing a Secure framework for data de-duplication in a distributed environment	For data integrity and minimizing duplication Bloom filter, Radix Trie, and cryptographic strategies are employed.	Overcome duplication ratio. Maintains data integrity.	Complexity increases for large volumes of data. To maintain Radix Trie, additional memory is required.
QuickCDC [71]	Dynamically setting mask bits to correct identification of chunk boundaries.	Uses feature vector and mask bits for data chunking, which results in appropriate packet size, identifies similar data chunks from the dataset, and skips them	Higher chunking speed. Can accurately identify and overcome duplication ratio.	Resource consumption is high.
ESDedup [74]	Block chain based scheme to provide security to data and de-duplication is assured in healthcare sector.	Checked data for duplication by matching similarity. Similarity based rewriting algorithm is utilized to remove identical chunks and store only one copy.	Reduced data redundancy and storage overhead. Ensures data security and integrity.	Data reliability is minimized.
SLRE [75]	Ensures duplication free data healthcare storage on cloud.	For duplicates identification, content defined entropy and similarity is checked.	Effective resource consumption. Overcome storage overhead.	Not appropriate to use for fault tolerant systems.

Secure de-duplication key management mechanism [72]	Ensures de-duplicated data and privacy is assured by the CEK mechanism	The data is transmitted by using tags based on content, when the same data is again received it is detected due to again same tag, and the duplicated copy of data is not transmitted.	De-duplication is ensured by the use of tags, Effectively managing Proofs of Ownership without showing data to users.	Management and updating of keys cause performance overhead. Limited scalability due to its complexity. High network traffic can lead to latency.
SEEDDUP [73]	Uses a combination of cryptographic approaches to save duplication-free data in the cloud.	Data is divided into chunks. By employing the Merkle hash tree, de-duplication is ensured. Key centre issues keys which are upgraded from time to time and data is saved in the cloud in encrypted format.	Efficient resource consumption. Protects against security attacks.	Higher complexity. Performance slows down in case of high data volume.
Block Level Data De-duplication				
A Design of Parallel Content-Defined Chunking System Using Non-Hashing Algorithms on FPGA	Use of PCI algorithm	This research presents a novel parallel chunking method designed specifically for hardware implementation, filling a significant gap in the current understanding and usage of content-defined chunking (CDC) algorithms.	The proposed design achieves not only high computational speed but also great scalability.	This research primarily focuses on the application of the PCI algorithm. Other non-hashing methods for the CDC could behave differently when implemented similarly.
TL-CH [76]	Efficient storage of duplication-free data by using encryption methods to make it secure.	Uses encryption mechanism to protect healthcare data and scalable data index table to manage de-duplicated data.	Ensures data integrity of healthcare data.	Handle data when index table size rises, it increases the cost to maintain it.

HealthDep [77]	Manage data by relying on a smartphone and using ARM Trust zone for security purposes of devices.	Categorization of healthcare data based on data readings severity, checked for duplication, and encryption mechanism is employed to encrypt data to maintain data privacy.	Storage- efficient scheme. Provides security to healthcare data.	Integration of the Healthcare system and HER is quite complex. The assumption considered in this approach cannot be useful in real-time scenarios.
Perfectdedup [78]	Ensures de-duplication and offer privacy to data with low and high privileged data differently.	Employ Perfect Hash Function for identification of popular data blocks and comprises of an Index Service for secure popularity transitions. The clients have flexibility to adjust dynamic threshold.	Reduced data duplication. Low chances of data leakage.	Computation overhead increases as the number of user's increases.

The existing schemes are stated above along with the pros and cons. The scheme in [58] does not make use of a hash key its primary focus remains on hardware configuration. In [59] using variable and fixed windows is employed for determining cut points. FastCDC [60] increases de-duplication accuracy and speeds up this process, in [61] involves using the mechanism of RNS & ECP for encryption, the hash key is used for data integrity and to perform verification while [62] scheme ensures security while resolving duplication issue, the cryptographic procedure is used for this process which may result in potential delays. In [63], to make the CDC better, the DER approach is used but fixed hashes result in low efficiency. To reduce hash table size, the bounded linear hash function is employed [79]. It includes a jump condition to avoid input data stream length while maintaining a rolling hash [65]. In [66] a dual encryption mechanism is introduced for security enhancement for deduplication-based data. To enhance de-duplicated data security, cipher text is upgraded with the help of a token [67]. To overcome data fragmentation issues, F-greedy and F-greedy+ algorithms are employed [68]. In [69], the author focuses on increasing the efficiency of the hash table lookup process. The false positive rate is increased by dividing data and creating signatures for them [76]. In [70], different roles are given to users, and tokens are generated for them so authorized users can get access as per their requirements. The healthcare data is categorized into high-sensitive and lower-sensitive data. Both categories are checked for similar data and dealt with by using different encryption mechanisms [77]. QuickCDC focuses on skipping the same data and the ratio of error jumps is reduced [71].

The scheme [62] shows higher efficiency in the de-duplication mechanism and reduces storage cost, while [63] has a low hash lookup time and balanced table size but is inappropriate to use for highly diverse data. In [66] [70] provides higher security to data to avoid any attack. The communication overhead is quite low in [67]. The [69] has high performance and lower performance degradation in highly diverse environments. The [70] requires more computational time. The [77] has good performance but resource consumption is high and not suitable for a resource-restricted environment. QuickCDC has a high de-duplication speed because setting mask bit [71], [65] [71] has better throughput. Each scheme has its benefits and limitations. On the whole, if scalability is the main concern in this case, [62] [79] [69] are better choices. If data privacy is the main concern [66] [70] should be considered. The [68] [71] is more economical and efficient resource consumption. The performance overhead is high in [64], while [65] performs de-duplication with efficient resource utilization but it has high latency and is not appropriate in case of high data volume. The cost of deployment is high but provides fast speed in the duplication mechanism [66]. The data is provided with different levels of privacy based on popularity but the computational overhead is high and if any entity becomes malicious, the user data can be compromised [78].

2.10 Research Gaps

Data duplication in the healthcare sector can cause serious problems. The crucial data can be overlooked or high energy consumption leads to early depletion of energy. To reduce duplication while keeping efficiency high, there is a high need for of implementing an optimized approach that ensures de-duplication. Instead of transmitting fixed data chunks, variable-sized chunks should be used. The byte pair comparison causes high energy consumption and memory overhead. The cut points in the data stream are required to be tackled intelligently. In case, data cut points are not appropriately done then some of the information in a packet is missing, which reduces the performance of a scheme. Besides this long hash table takes extra time for lookup and degrades the performance of the scheme. Longer hash values can reduce the likelihood of collisions and improve data integrity, they also come with drawbacks such as increased computational overhead and storage requirements. Therefore, finding the optimal balance between hash size and efficiency is crucial in designing robust and efficient systems. It is recommended to address this issue by exploring methods to mitigate collision risks with smaller hashes while also optimizing the performance of longer hashes for practical implementation. These factors need to be handled smartly so that the duplication is managed in a better way without taking a longer time to process data, lowers communication costs, and consumes little energy.

2.11 Summary

In this chapter, a detailed review of existing literature is presented. The main terms used throughout the thesis are explained well. The previous schemes are described in the context of data duplication. Afterwards, the research gap is explained which exists in current techniques. A table of comparison is shown with their basic idea, mechanism advantages, and drawbacks. Afterwards, the whole chapter is concluded.

CHAPTER 3

METHODOLOGY

3.1 Overview

In this chapter, the detailed research methodology of the proposed solution is presented which is employed throughout the research. The chapter comprises information about the literature review study, problem identification, and evaluation metrics, along with dataset and simulation tool to evaluate the proposed scheme. In the end, the entire chapter is concisely concluded.

3.2 Operational Framework

In this era of advancing technology, the healthcare sector has also shifted to the online paradigm for monitoring patients remotely and deciding on better treatment for them. For this purpose, various sensors are attached to the bodies of patients to observe and evaluate patients' health purpose. Typically, the data gathered by these sensors takes sixteen or thirty two bits. In many cases, the obtained data lies within the usual range, and storing this data recurrently can be ineffective and resource-intensive. For this purpose, the concept of data de-duplication is employed in data storage and management to remove repetitive data copies. By eliminating these redundancies, the total data quantity is diminished, leading to more efficient storage utilization and cost reductions. The de-duplication procedure involves observing the data set to check repeating data values. Once identified, these duplicated chunks of data are reduced in size, sustaining just one instance, in the proposed scheme to reduce resource consumption and increase efficiency. The study of data de-duplication in healthcare within the context of IoT-assisted cloud computing is essential because of the above-mentioned issues. The Working Framework of the Research is presented in Figure 3.1.

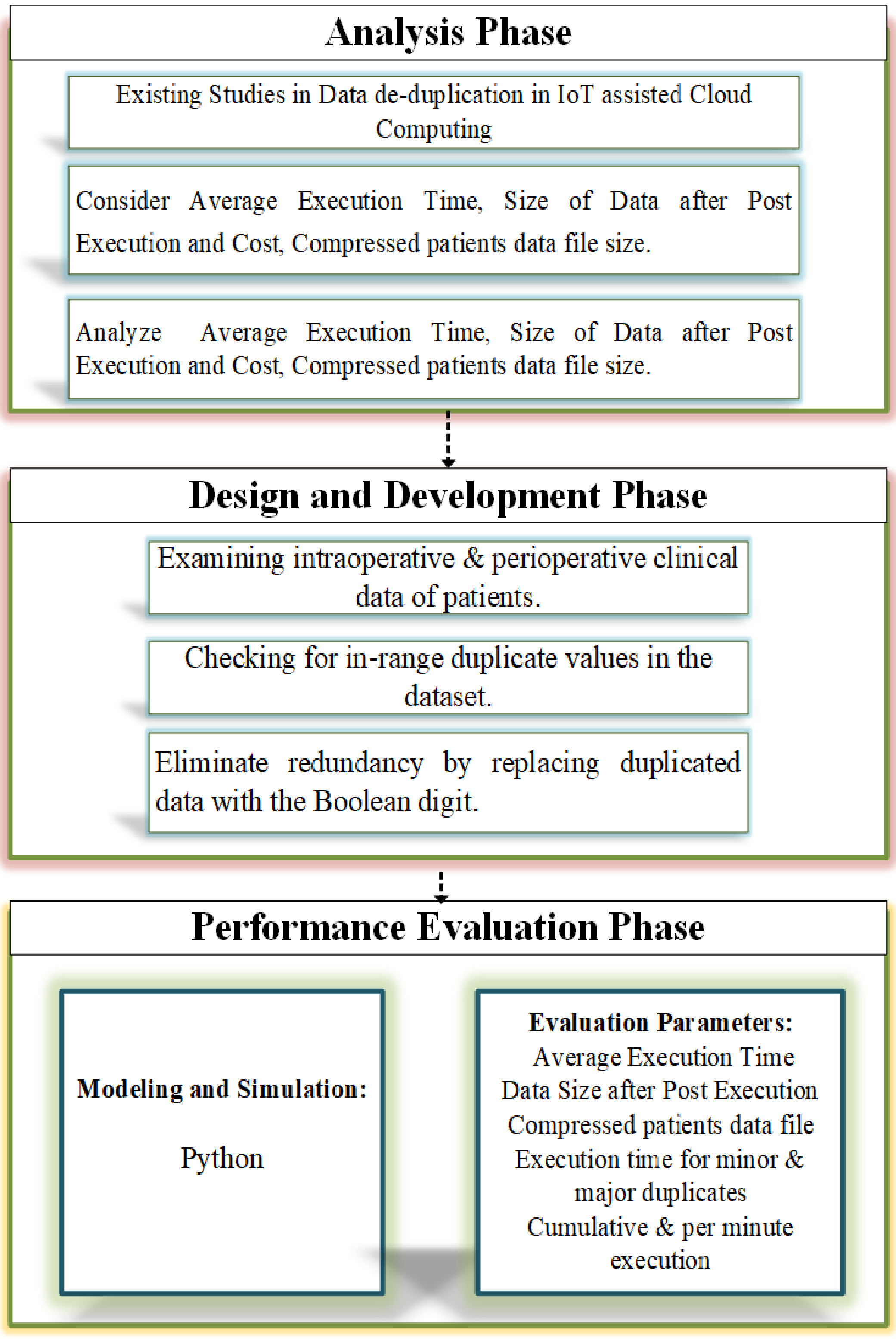
Analysis Phase
Existing Studies in Data de-duplication in IoT assisted Cloud Computing
Consider Average Execution Time, Size of Data after Post Execution and Cost, Compressed patients data file size.
Analyze Average Execution Time, Size of Data after Post Execution and Cost, Compressed patients data file size.
Design and Development Phase
Examining intraoperative & perioperative clinical data of patients.
Checking for in-range duplicate values in the dataset.
Eliminate redundancy by replacing duplicated data with the Boolean digit.
Performance Evaluation Phase
Modeling and Simulation:
Python
Evaluation Parameters:
Average Execution Time
Data Size after Post Execution
Compressed patients data file
Execution time for minor & major duplicates
Cumulative & per minute execution

Figure 3.1: Working Framework of the Research

3.3 Research Design and Development

To resolve the above-mentioned problem, the proposed solution passes through three important levels which are elaborated here. The wearable devices are injected into the skin of patients or attached to their bodies to obtain various required health-related readings. Our primary objective is to manage data de-duplication, regardless of whether the devices are wearable or non-wearable. However, using them has hazards, including tissue damage and infection, so medical practitioners must exercise caution. This data is transmitted repetitively to the fog server. Not all transmitted data is crucial, most of the time variation exists in a few device values while other data remain in the normal form. So transmission of normal data again and again causes redundancy and duplication. To resolve this issue, the proposed scheme Healthcare Data De-duplication Scheme (HDDS) is introduced. In the current research, intra-operative and preoperative information of patients are considered who are undergoing a surgical process.

The use of invasive and non-invasive sensing devices to capture patients' vital signs for the purpose of collecting healthcare data. Despite its intention to address data redundancy and duplication, the Healthcare Data De-duplication Scheme (HDDS) still needs to get initial data from these devices. These tools are necessary to immediately collect and record precise physiological data from patients' bodies, which serves as the foundation for additional processing and analysis. Sensing instruments are therefore still essential for efficient healthcare monitoring during surgical procedures, even with HDDS optimization. The Research Design and Development phase of the proposed scheme is shown in Figure 3.2.

In the first phase, the aggregated healthcare data sets from vital devices are loaded from the corresponding dataset. This dataset is related to blood pressure, ECG, glucose level, White blood cells, Haemoglobin, Platelet, and Hematocrit etc. These vital data vary from each other as some have two parts, totalling producing 32-bit long data packets.

In the second phase, this data is sent to the fog server for storage and then checked for usual and unusual health-related values. In the third phase, repeated data that is in the usual range is replaced by a smaller chunk consisting of a single Boolean digit 1. Transmitting data chunks of a single bit in this case, instead of the original full data packet, reduces the load on the server and efficiently stores the data in a cloud repository while serious healthcare data is sent in its original form to the cloud server. Moreover, this approach not only cuts down energy consumption, communication costs, and storage space but also gives suitable and critical patient reports to medical staff. That helps them in the identification of patients' health situations more effectively and follows more efficacious treatment options.

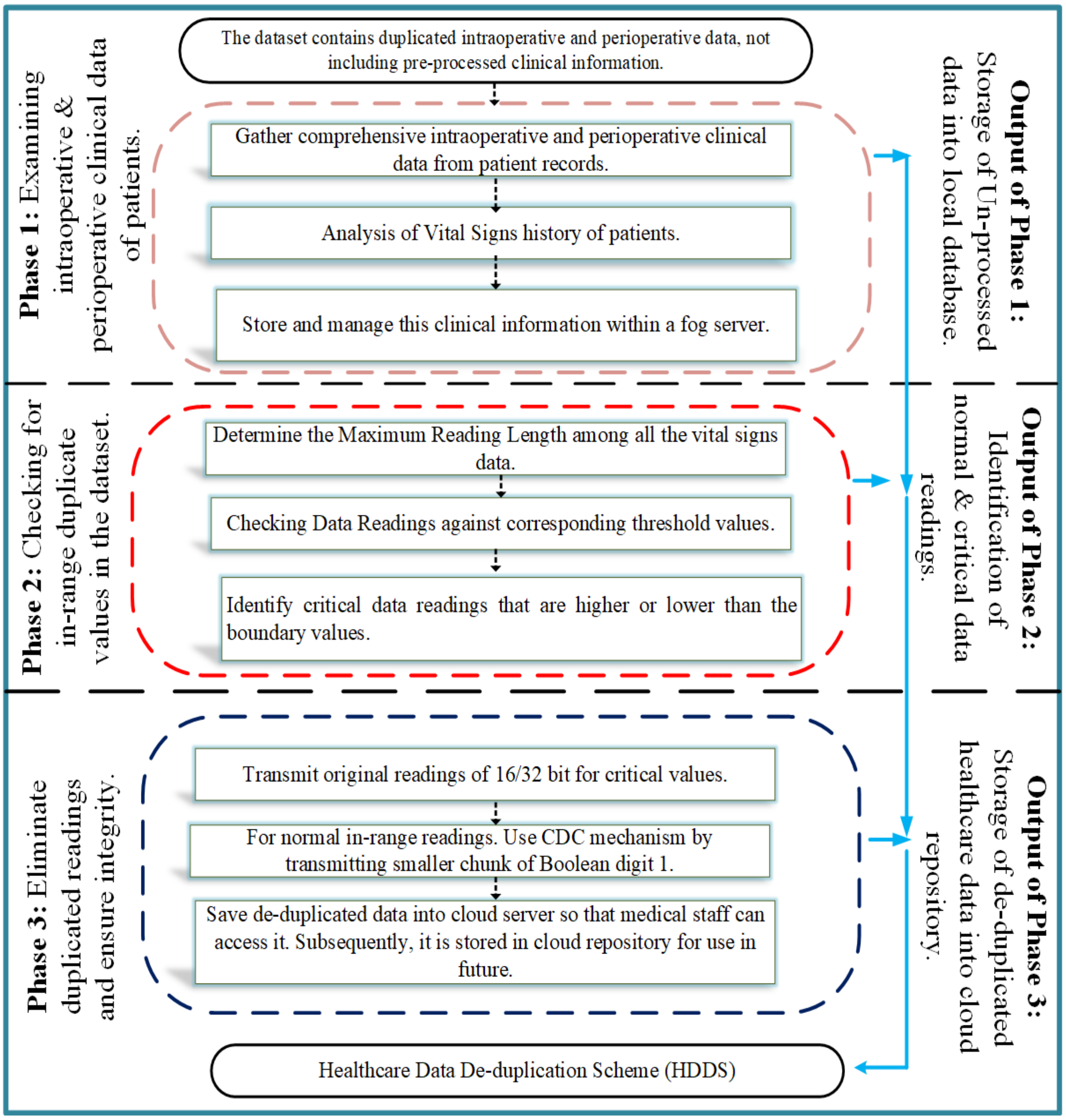
Phase 1: Examining intraoperative & perioperative clinical data of patients.
The dataset contains duplicated intraoperative and perioperative data, not including pre-processed clinical information.
Gather comprehensive intraoperative and perioperative clinical data from patient records.
Analysis of Vital Signs history of patients.
Store and manage this clinical information within a fog server.
Output of Phase 1: Storage of Un-processed data into local database.
Phase 2: Checking for in-range duplicate values in the dataset.
Determine the Maximum Reading Length among all the vital signs data.
Checking Data Readings against corresponding threshold values.
Identify critical data readings that are higher or lower than the boundary values.
Output of Phase 2: Identification of normal & critical data readings.
Phase 3: Eliminate duplicated readings and ensure integrity.
Transmit original readings of 16/32 bit for critical values.
For normal in-range readings. Use CDC mechanism by transmitting smaller chunk of Boolean digit 1.
Save de-duplicated data into cloud server so that medical staff can access it. Subsequently, it is stored in cloud repository for use in future.
Output of Phase 3: Storage of de-duplicated healthcare data into cloud repository.
Healthcare Data De-duplication Scheme (HDDS)

Figure 3.2: Research design and development Plan for HDDS

3.4 Detailed study of Literature

The up-to-date research papers on de-duplication within the context of IoT -assisted cloud computing are considered. The paper's title, abstracts, proposed solution, and conclusion are thoroughly studied to build an idea of the relevant domain. Around 50 research papers are studied and 23 papers are the supreme appropriate. Based on the conclusion drawn through these papers, the research problem is formulated. These research papers are found based on keywords and the most recent papers are also checked in the reference list of relevant research papers. Subsequently, based on the literature review, the problem background is developed and the main problem is identified. Afterwards, a table with having analysis of suitable schemes comprised of the main research idea, procedure, advantages, and issues of these papers is presented.

3.4.1 Problem identification

Based on the studied literature, the problem statement is formulated. The negative impact of the problem includes high storage, higher energy consumption, and high communication costs. These outcomes of the problem provide the basis to address this critical issue and take efficient steps to mitigate it.

3.4.2 Evaluation Metrics

To check the proficiency of the proposed scheme, it is compared with the existing robust schemes. The metrics are selected which are most crucial from a de-duplication perspective. These parameters consist of Average Execution Time, Size of Data after Post Execution, and compressed patient data file size. Through graphs, the performance of the proposed scheme is shown along with other schemes.

3.4.3 Experimental Dataset and Selection of Simulation Tool

To evaluate the effectiveness of the proposed scheme, the simulation method is employed. The Python programming language is utilized for simulation. The dataset used for the research is available at https://physionet.org/content/vitaldb/1.0.0/. The vital signs data consists of ECG, BP, body temperature oxygen saturation level, and many others. For each patient, a 2000 dataset of each patient is maintained. It includes 12 waveform tracks, as well as 184 numeric data tracks, gathered using various anaesthesia instruments attached to the patient’s body during the surgery mechanism. These vital signs data contains 1-second interval. The present data is not pre-processed keeping in view the real-world scenario.

3.5 Summary

In this section, a comprehensive pathway is elaborated that is followed in the research work. From the operational framework to the simulation tool, every step is described in detail. The evaluation metrics and characteristics of the dataset implemented are described thoroughly to enhance understanding.

CHAPTER 4

HEALTHCARE DATA DE-DUPLICATION SCHEME

4.1 Overview

In this chapter, the proposed scheme is described which is used to resolve the issue of data duplication. Afterwards, the system model of the scheme with all relevant entities is described. Then, an algorithm is stated as well and its steps are defined well to increase readability. Finally, the whole chapter is concluded.

4.2 Healthcare Data De-duplication Scheme (HDDS)

In this section, the proposed scheme Healthcare Data De-duplication Scheme (HDDS) is described which aims to overcome the issue of data duplication. In the context of the healthcare sector, the vital devices are attached to the body of patients to be monitored and treatment is provided to them as per their health condition. In the current research dataset of patients are used that are undergoing surgery. The intra-operative and preoperative clinical information is gathered through vital devices. These attached devices vary in nature. Some sensor devices generate data packets of 16 bits because they produce only a single portion of the data. Some device such as BP tracking, sugar tracking, etc. produces data in 2 halves. Each one of them occupies 16 bits so jointly it takes 32-bits. The attached vital devices continuously send body readings to the fog server which sends it further.

Repeatedly transmitting these readings takes up high storage space and drains energy quickly. Some data readings display instabilities among the patients, while other data values remain constant. These data readings that continuously experience variation are termed unusual data. To overcome above mentioned issues, the concept of content-defined chunking is employed. The data readings are split based on actual data content. The variable-sized chunks are created depending on the nature of the data. After that duplicated data chunks in the data are checked. The content of each chunk is examined to determine duplicated values. For usual duplicated data readings, instead of sending the whole data packet of all bits, transmit a smaller chunk of Boolean digit 1, to indicate that the data reading remains within the normal range. So transmitting a data chunk of 1 bit and then keeping meta-data about this duplicated content reduces the load and saved into the cloud repository. On the other hand. it not only reduces data file size, communication cost, and storage space also this relevant and vital information is useful for medical staff to focus on serious data that assist them in diagnosis and taking better treatment approaches as per the condition of the patients.

This approach of content-defined chunking and transmitting smaller Boolean digits to represent duplicated data within normal ranges offers several advantages. Firstly, it significantly reduces the storage space required, as only relevant and varying data is stored rather than repetitive information. This optimization also translates into reduced communication costs, as smaller chunks of data are transmitted, minimizing bandwidth usage.

Moreover, by focusing on unusual data readings and transmitting only the necessary information, the energy consumption of devices is minimized, prolonging battery life and reducing the need for frequent recharging or battery replacements.

Additionally, the use of metadata to keep track of duplicated content enables efficient retrieval and analysis of data. Medical staff can quickly identify and prioritize unusual or significant data readings, allowing them to focus their attention on patients who require immediate attention or intervention. This targeted approach to data analysis can lead to more timely and accurate diagnoses, as well as more tailored treatment approaches based on individual patient needs.

Overall, the implementation of content-defined chunking and selective transmission of data chunks offers a practical and efficient solution to the challenges of storing, transmitting, and analyzing medical data in a resource-constrained environment. It not only optimizes resource usage but also enhances the effectiveness of healthcare delivery by providing medical staff with timely and relevant information for decision-making and patient care. The working flow of the proposed scheme is shown in Figure 4.1

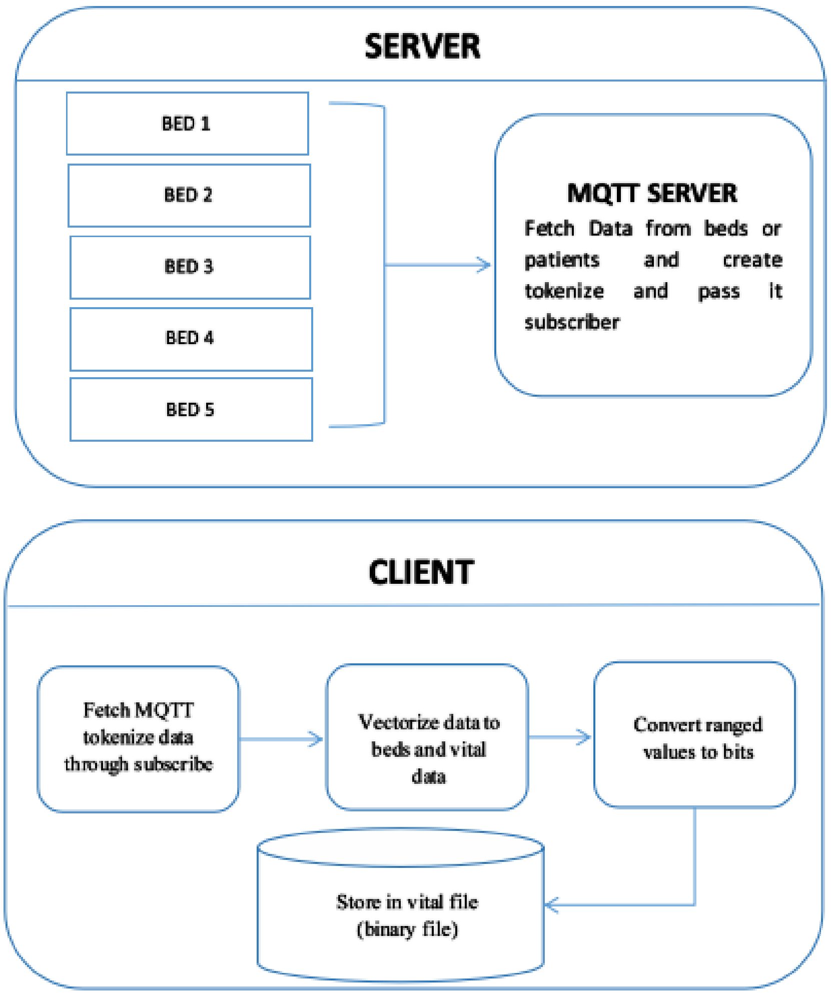
SERVER
BED 1
BED 2
BED 3
BED 4
BED 5
MQTT SERVER
Fetch Data from beds or patients and create tokenize and pass it subscriber
CLIENT
Fetch MQTT tokenize data through subscribe
Vectorize data to beds and vital data
Convert ranged values to bits
Store in vital file (binary file)

Figure 4.1: Working Flow of Proposed Scheme

The conventional methods of de-duplication, such as byte pair comparison throughout the byte stream introduce extra computations that the system needs to perform to identify cut points in the data. These extra computations increase overhead, which is unwanted, particularly in resource-restricted healthcare environments. The content-defined chunking approach used in this scheme reduces the extra computation by focusing on the actual repetitive data values and replacing them with Boolean 1 rather than checking it byte by byte. It not only lessens the computational overhead but is also suitable for functioning within healthcare to ensure an effective de-duplication mechanism.

4.3 System Model

The System model for the proposed scheme comprises vital devices, a Fog Server, a Cloud server, and a cloud repository. The vital devices attached to the body of patients. This data has identical readings related to patients’ health history. The purpose of these sensor devices is to take data commonly from patients’ bodies and transfer it to the fog server. The fog server in the system model is responsible for storing data of these vital devices and after removing duplicates transmits it to the cloud server. For removing redundancy in this data, Boolean digit 1 is used. This de-duplicated data is further stored in the cloud server. From the cloud server, authorized persons such as doctors no nurses can access it. Along with it, this de-duplicated medical history of patients is also stored in the cloud repository. In case data is required later on, it can be retrieved from the cloud repository. The System model of the proposed scheme is presented in Figure 4.2.

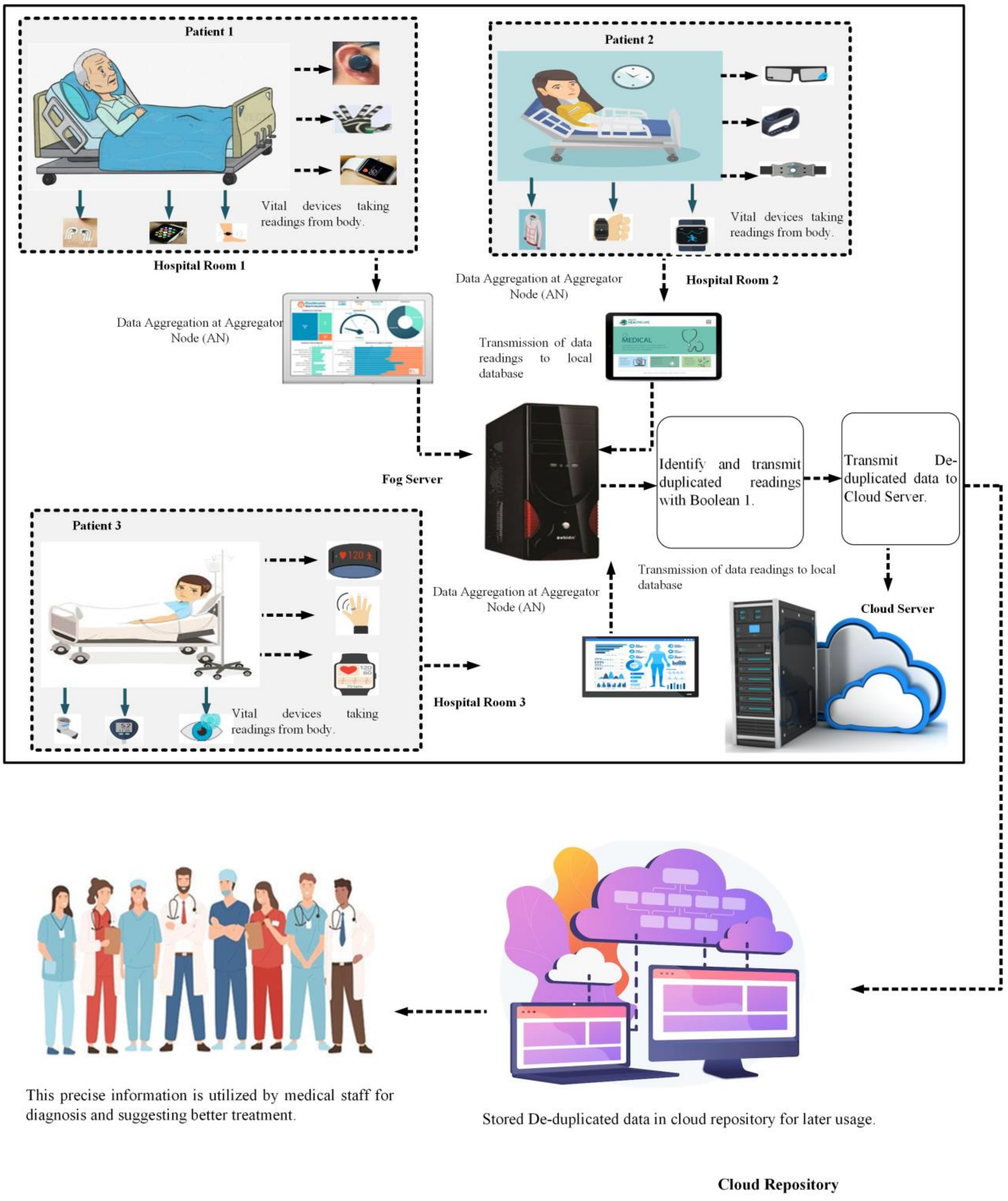
Patient 1
Vital devices taking readings from body.
Hospital Room 1
Data Aggregation at Aggregator Node (AN)
Patient 2
Vital devices taking readings from body.
Data Aggregation at Aggregator Node (AN)
Hospital Room 2
Transmission of data readings to local database
Fog Server
Identify and transmit duplicated readings with Boolean 1.
Transmit De-duplicated data to Cloud Server.
Patient 3
Data Aggregation at Aggregator Node (AN)
Transmission of data readings to local database
Cloud Server
Vital devices taking readings from body.
Hospital Room 3
This precise information is utilized by medical staff for diagnosis and suggesting better treatment.
Stored De-duplicated data in cloud repository for later usage.
Cloud Repository

Figure 4.2: System Model

4.4 Flowchart of HDDS

The healthcare dataset of patients is loaded that are undergoing surgical procedures. After loading the dataset, the vital sign values including Blood pressure, ECG, Sugar level, pulse rate, and some other data files associated with vital signs are checked and stored in the fog server. The normal range varies for each of the vital signs. So they are checked against corresponding standard values. If they fall in the normal range then it is termed non-critical or normal readings. The Flowchart of the proposed technique HDDS is shown in Figure 4.3.

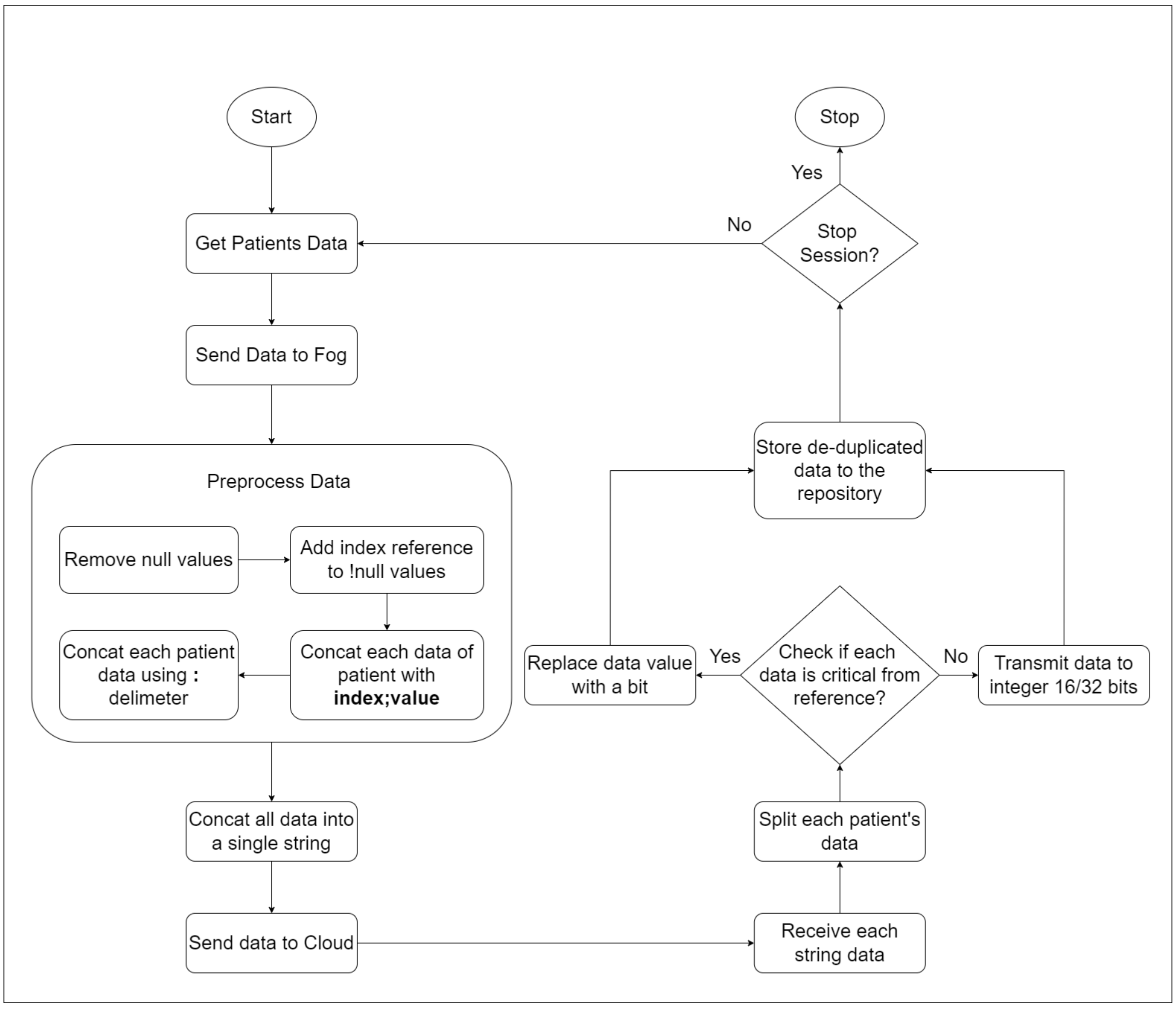
Start
Get Patients Data
Send Data to Fog
Preprocess Data
Remove null values
Add index reference to !null values
Concat each data of patient with **index;value**
Concat each patient data using **:** delimeter
Concat all data into a single string
Send data to Cloud
Receive each string data
Split each patient's data
Check if each data is critical from reference?
Yes
Replace data value with a bit
No
Transmit data to integer 16/32 bits
Store de-duplicated data to the repository
Stop Session?
No
Yes
Stop

Figure 4.3: Flowchart of HDDS

On the contrary, if these readings are greater or lower than the relevant standard level, they are well-known as critical values. The critical values are transmitted in their full bits. If the readings are non-critical, in this scenario, instead of sending data in its full form, it is replaced with a small data chunk of Boolean digit 1. After removing redundancy from the healthcare data it is stored further in a cloud server where medical staff can access it to suggest better treatment and afterwards, it is saved in a cloud repository for future use.

4.5 Algorithm for Healthcare Data De-duplication Scheme (HDDS)

For removing redundancy from healthcare data, an algorithm is presented. The thorough detail of the algorithm is shown in Figure 4.4. Then, we provided an in-depth explanation to improve understanding.

```
tn = track names, ct = concat string
1. Input:  List of Vital data
2. Output: Concatenated string
3. vf1, vf2, vf3, vf4, vf5 vitals data
4. maxLen => pick the maximum data length
5. For v till maxLen
6.       check if all vitals data contains index v
7.             pick vitals data from each vf's
   End For
8.             ignore nan values and pick column names of the values that
are not nan
9.             get index i of column names from tn
10.             ct = create concat string where i is index and value is
concatenated by semicolon. e.g i1;value,i2;value,i3;value
11.            concat next vf's data to the ct
12.            Transmit ct to client
```

Figure 4.4: Algorithm for Aggregating Vital Data

In 1-2 step, the input and output of the algorithm is stated. In step 3- vf1, vf2, vf3, vf4, and vf5 represent vital data such as blood pressure, pulse rate, sugar level, oxygen saturation, etc. level in the body. The maximum data length is determined for all the vital data sources and For loop is run for all vital signs. It is checked that all sets of vitals data are comprised of data at index v. It is done to ensure that data is available at the current iteration index in all vitals data sets. Among the data points at index v in each set, ignore if it has "nan" values (which signifies missing data). The columns' names of the values are obtained that are not ‘nan’. The column names got in the prior step to their corresponding indices in the track names. Join the data obtained from the next set of vitals data to the previous concatenated string. This step is done in every loop iteration, adding data for the next index (v) in each iteration. After processing the entire data for the current index v from all vitals data sets, transmit the concatenated string further. “nan” values are basically empty columns in the dataset. So, we are ignoring those values and not passing to the concentrated strings.

4.6 Algorithm for Removing Duplicate Data

After the data is transmitted to the fog server. It is checked for duplicated values. For this purpose, an algorithm is presented in Figure 4.5 that depicts the functionality in detail.

```
1. Input: Duplicated data
2. Output: De-duplicated data
3. fetch ct from Mqtt
4. split column number and value
3. check threshold value according to column
5. if threshold satisfies then
5.        replace value with bit 1
6. else
7. keep the same value
8. save data to cloud server /json/file/vitalfile without other
missing columns, to reduce the size of the data
```

Figure 4.5: Algorithm for Removing Duplicate Data

In steps 3-8, the stored concatenated data string is fetched from Mqtt. The column and numbers are separated. These data values are checked against standard threshold values as per the nature of the data. In case of redundant readings, transmit only Boolean digit 1 otherwise the original data file is sent to the cloud server.

4.7 Summary

This chapter aims to provide comprehensive information about the proposed scheme. Initially, the complete explanation of the proposed scheme is elaborated there. Then, the system model is presented with the functionality of all entities. Afterwards, a flowchart is illustrated for the proposed scheme with an explanation. Then, algorithms for the proposed scheme are added with a complete explanation of all steps.

CHAPTER 5

RESULT AND ANALYSIS

5.1 Overview

In this chapter, the proposed technique HDDS is compared with some existing robust schemes. The results are formulated and analysis is performed by using the simulation method. For evaluation, crucial metrics are considered from the data de-duplication perspective. Along with it, information is given about the simulation environment and parameters that are considered to obtain results.

5.2 Simulation Tools and Environment

An MQTT-based pub-sub architecture is used to simulate the transmission of healthcare IoT data in JSON format at 1-second intervals by an MQTT broker, which is afterwards broadcast to connected MQTT receivers.

The dataset includes intra-operative vital signs data and preoperative clinical information of patients. This setup comprises five separate patient files, each signifies vital health data for the patient in a hospital bed. These files are related to fog nodes as it is responsible for collecting patient data and combining it into a single string for transmitting it to the cloud. Relevant information to the vital data parameters, which can be considered as a "helper" or meta-data file, is kept distinctly in a file named "track_names.csv”. This file comprises details such as the device name, and description (i.e., the parameter being monitored, such as heart rate or ECG lead wave). The unit of measurement and minimum and maximum permissible ranges for each parameter are presented in the file. The fog node is responsible to retrieve the information from the "track_names.csv" file to compile it into a string.

This string is separated by using various delimiters to differentiate between different parameters and patient data. Subsequently, the compiled data is sent to the cloud. The delimiter for index values is known as a semicolon, for example, "6;66.237030029296," where "6" signifies the index number corresponding to a specific parameter (e.g., "ECG"), and "66.237030029296" represents the parameter's value. The value delimiter is a comma that separates various health parameters within the string. For example, "6;66.237030029296,7:120" shows that "6" relates to ECG with a value, followed by "7" shows Blood Pressure with its corresponding value. Patient data is differentiated by using a colon, representing the start of data for the next patient.

On the aggregate node at the cloud end, received data is parsed and separated using the delimiters. Additionally, a validation step is performed to ensure that the values fall within the predefined minimum and maximum ranges for each vital parameter. If a value falls within this range, it is replaced with "1" to indicate normal. Otherwise, the original value is retained and stored in the cloud.

In the proposed scheme described, it seems that only the upper value for blood pressure (BP) is stored. This could be due to a design decision or requirement specific to the application. Storing only the upper value might be considered sufficient for monitoring purposes or for certain analytical needs. Additionally, storing both upper and lower values might introduce redundancy if they tend to change together within the normal range for a given patient.

However, if there's a need to store both upper and lower values of BP, the proposed format can be adapted accordingly. One way to store both values within the existing format could be by allocating additional index numbers for the lower and upper values separately. For instance:

- Let's say we use index "7" for blood pressure.

- The original format might store the upper value like this: "7;120".

- To include the lower value, you could designate another index, for example, "8" for the lower value.

- So, the lower value could be stored as "8;70", assuming the lower value of BP is 70.

- Thus, the entire string representing BP for a patient might look like: "7;120,8;70".

This way, both upper and lower values can be accommodated within the existing scheme, maintaining clarity and consistency in data storage and retrieval.

The original data of patients is stored in Healthcare Data Repository (HDR). The results of proposed scheme HDDS is compared with Efficient File-Level De-Duplication Scheme (EFDS) [26] scheme. The Simulation parameters utilized to check the efficiency of the proposed technique are presented in Table 5.1.

Table 5.1: Simulation parameters

Parameters	**Unit**	**Reference Threshold Values**
Patients Number	-	5
Simulation time	sec	100
White Blood Cell Count	x1000/mcL	4~10
Haemoglobin	g/dl	13~17
Hematocrit	%	39~52
Platelet count	×1000/mcL	130~400
Erythrocyte Sedimentation Rate	mm/hr	0~9
Glucose	mg/dl	70~110
Albumin	g/dl	3.3~5.2
Total Bilirubin	mg/dl	0.2~1.2
Creatinine clearance	mL/min	75~125
Sodium	mmol/L	135~145
Potassium	mmol/L	3.5~5.5
ionized Calcium	mmol/L	1.05~1.35
Chloride	mmol/L	98~110
Ammonia	mcg/dL	27.2~102

C-reactive protein	mg/dL	0~0.5
Lactate	mmol/L	0.5~2.2
Prothrombin time	INR	0.8~1.2
Prothrombin time	%	80~120
Prothrombin time	sec	10.6~12.9
Activated partial thromboplastin time	sec	26.7~36.6
Fibrinogen	mg/dL	192~411
Arterial pressure wave	mmHg	40-50
partial pressure of CO2	mmHg	35~48
Bicarbonate	mmol/L	18~23.0
Diastolic arterial pressure	mmHg	40-50
Systolic arterial pressure	mmHg	40-50
Percutaneous oxygen saturation	%	40-50
Heart rate	/min	40-50
Respiratory rate (from ventilator)	/min	40-50
Respiratory rate based on capnography	/min	40-50

5.3 Compressed Patient Data File Size

The compressed size refers to the reduction in the size of the data file as compared to its original size. For this purpose, the data of 5 patients are considered comprises of 2000 datasets for each. Each dataset contains collected data from 70 to 75 attached vital devices. The simulation results demonstrated as shown in Figure 5.1 that HDDS has a higher compressed size as compared to the EFDS scheme. The EFDS scheme keeps a reference list of repeated values in the coming data. This reference list keeps increasing significantly as more repetition of data occurs, which eventually has a negative impact on the compression rate. The reason behind better compression size in HDDS adopts a predefined set of reference values, known as meta-data. This predefined reference list of reference values stays constant during the de-duplication procedure, irrespective of the amount of duplicated data. Furthermore, HDDS also uses a mechanism in which when data values lie within a stable range are transmitted in the form of Boolean digit 1 instead of transmitting whole data. This technique additionally plays its role in the proficient compression of data.

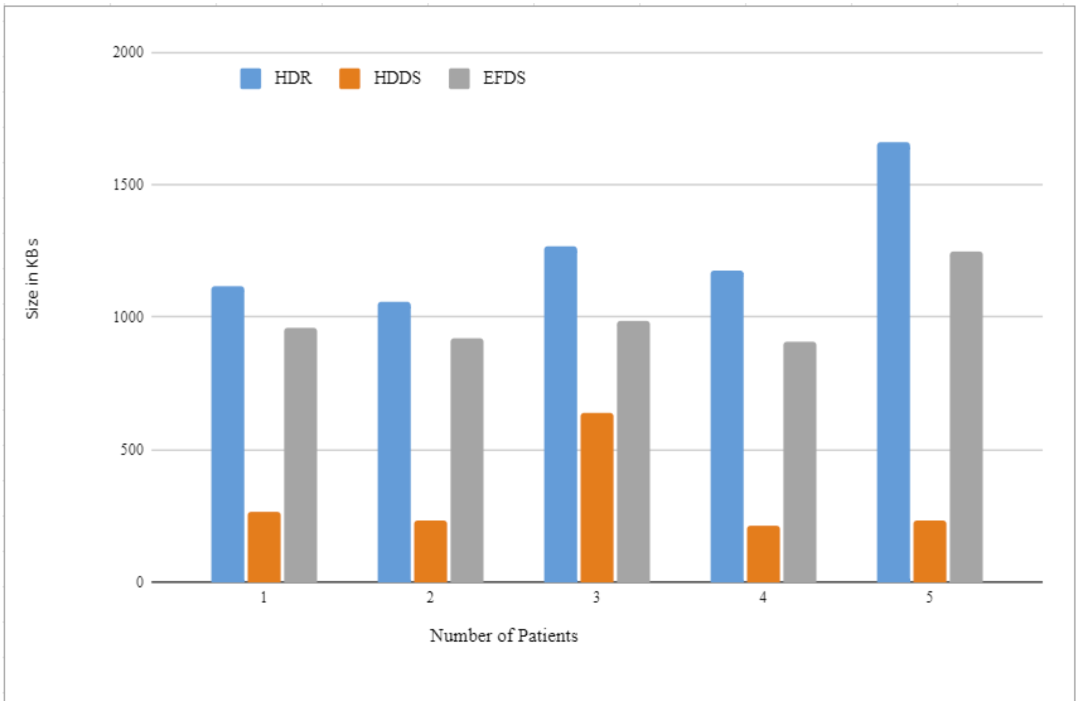

Figure 5.1: Compressed patients' data file size

The X-axis represents the number of patients and the Y-axis shows the size of the file in Kilobytes (KBs). When the original size of the file is 1059 KBs, after employing EFDS the size of the file is compressed to 920 whereas after using the HDDS strategy it is further reduced to 235.

5.4 Average Execution Time

During the initial phase, the performance of EFDS is much better. At the start of the de-duplication process, EFDS is performing more proficiently in terms of execution time as compared to HDDS. Duplication refers to the mechanism of removing normal repetitive data values collected by vital devices. With time the de-duplication process remains for a longer time and more identical data is discovered. To cope with this redundant data, both EFDS and HDDS schemes keep reference lists. When more duplicate data is received, the EFDS scheme needs more lookup and updates its reference. The continual updating of the reference list in EFDS has a bad impact on execution time. As more duplications occur in healthcare data, the time needed to EFDS for lookup and update the reference list surges fast. This increasing reference list increases complexity and takes more time to update.

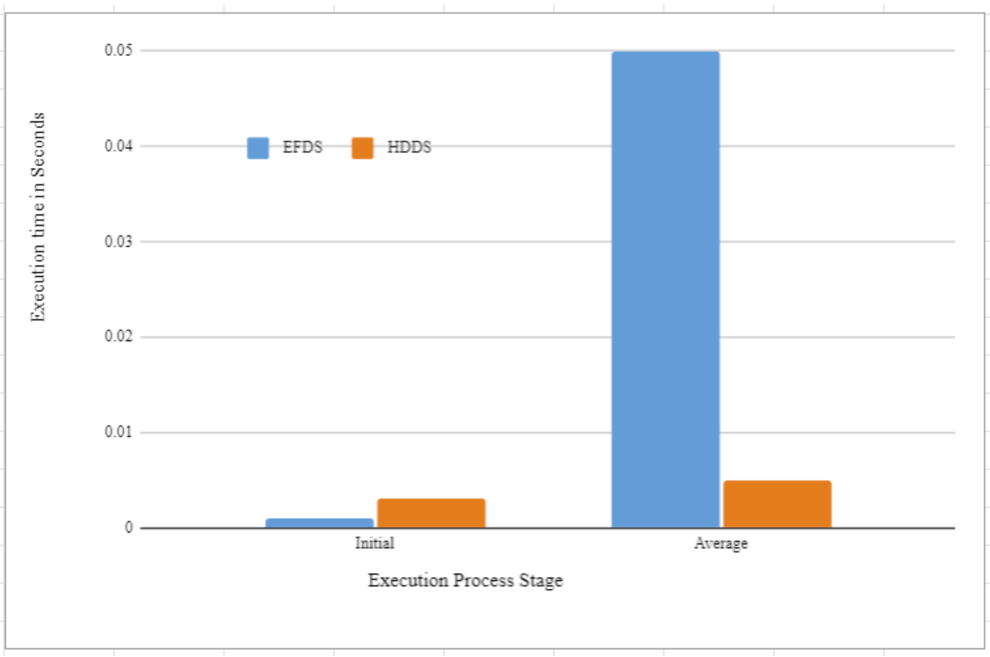

Figure 5.2: Average Execution Time

On the other hand, HDDS needs less execution time because it maintains a fixed, constant reference list during the whole execution procedure. The size of the reference list in HDDS remains consistent, irrespective of duplicated data that needs to be processed. Figure 5.2 illustrates the performance of both schemes when the number of vital devices surges up to 73, the EFDS requires 0.05 seconds while HDDS needs 0.005 seconds.

5.5 Size of Data after Post Execution

The healthcare data is frequently aggregated by 73 vital devices attached to the patients after every second. There is a higher rate of duplicated data in it, which is not essential to transmit again and again to the cloud by fog sever as it reduces energy consumption, increases communication cost, and takes extra storage space. So the size of the data is reduced by removing duplicated values so that the above-mentioned parameters can be controlled. The basic motive behind de-duplication is to data size to optimize healthcare data transmission and storage. In Figure 5.3, the original data is shown along with de-duplicated data presented by using EFDS and HDDS schemes. The results showed that after the de-duplication mechanism, the smaller data size is in HDDS as compared to EFDS.

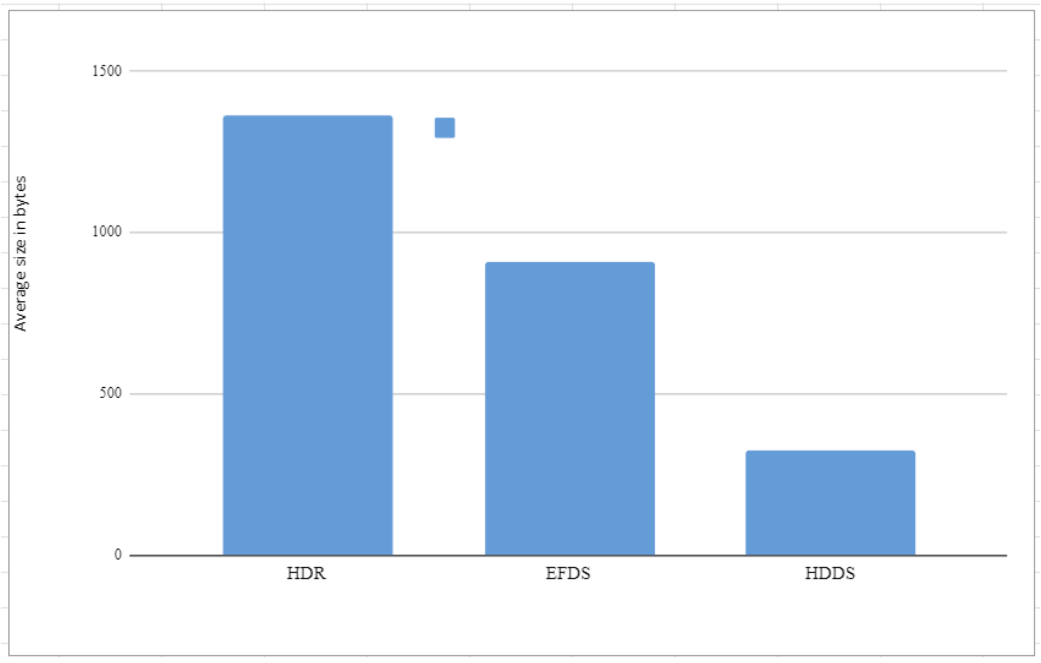

Figure 5.3: Size of Data after Post Execution

After de-duplication, the size of the data is significantly smaller with HDDS compared to EFDS. The original data gathered from the 73 devices after an interval of every 1 second, needs 1364 bytes. Besides this after using the EFDS de-duplication mechanism the data size is reduced to 907 bytes while in HDDS it is further reduced to 326 bytes. The proposed scheme HDDS has a lower data size than EFDS which shows that ultimately HDDS has lower storage cost and lower communication cost.

5.6 Patient Data with Minor Duplicates

When the dataset is randomized it presents that most of the data readings are not the same as it contains values with variation. There are a lower number of duplicate data values and very few null or missing readings. The performance of both schemes is better because of the low duplicate data ratio. When duplicated data is low in volume, EFDS reduces the duplication reference list to grow and the time needed for lookup is also reduced. In HDDS, when minor duplicated data appears in the dataset, it is converted into digit 1 and then transmitted further which reduces data size for duplicated values. While out-of-range data is transmitted in its full form. Figure 5.4 shows the performance of both HDDS and EFDS.

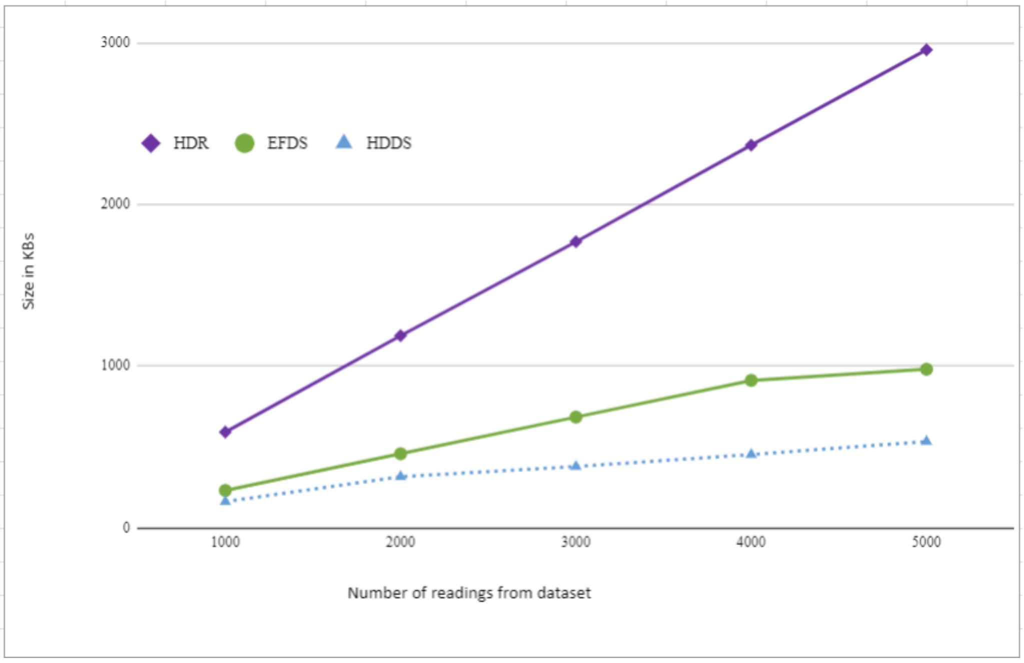

Figure 5.4: Patient Data with Minor Duplicates

The x-axis shows the number of readings from the dataset and the y-axis represents file size in KBs. When readings are 2000 in number the size of the file stored in the healthcare repository is 1188, while the EFDS has maintained a size of 457 and HDDS maintained the lowest size of 315.

5.7 Patient Data with Major Duplicates

As in the previous performance metric, the case observed with data values having a lower duplication ratio is observed. In this scenario, the dataset with major duplicates is considered. When the duplication ratio is high, it shows that the data contains a higher ratio of similar data. The EFDS has to maintain a huge list of references for duplicated records. When the identical data starts rising, the performance of EFDS degrades. In the case of HDDS, instead of high duplicated values, the size of the file remains lower because of better management of duplicated data. The feature of data reduction for similar data by using the Boolean number system increases the efficiency of the proposed scheme. The file sizes in both schemes are shown in Figure 5.5. When readings in the dataset are 9000, the size of the stored file in the original form in a healthcare data repository takes 4745 KBs, by employing the EFDS scheme it takes 1765 KBs, and the proposed scheme HDDS maintained the lowest size of 1021 KBs.

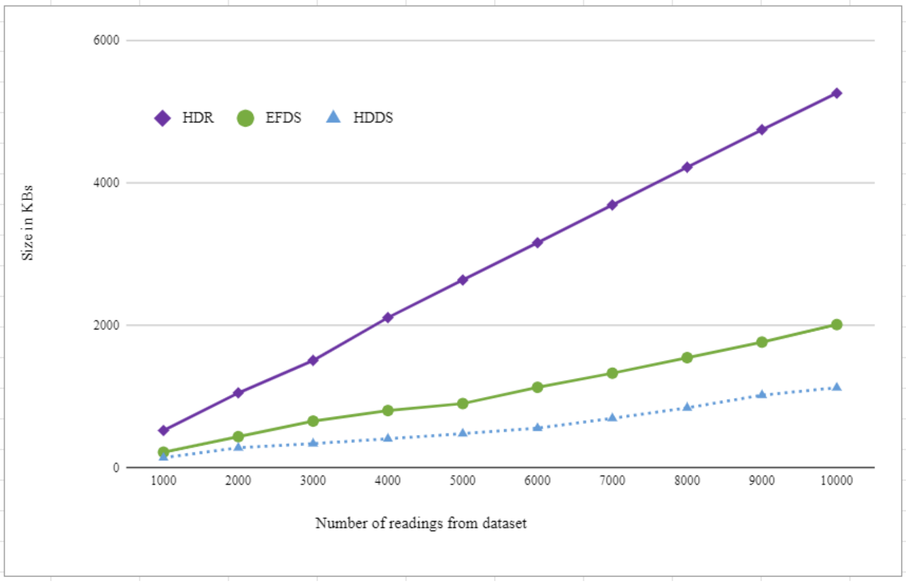

Figure 5.5: Patient Data with Major Duplicates

5.8 Execution Time for Processing Patients with Minor Duplicates

When minor duplicates exist in the healthcare dataset as shown in case 1, there is not much processing needed, and handling a small number of duplicates is relatively easy and efficient. When duplicates are high in number, handling them is resource-intensive, particularly in EFDS. When duplicates are fewer, then the step of handling them is not exhaustive. So execution time remains the same for both schemes in case of minor duplicates.

The performance of both schemes is illustrated in Figure 5.6. The x-axis shows data readings from the dataset and the y-axis represents execution time. When data values are about 9000 in number, it takes 82 minutes for EFDS and 81 minutes for HDDS.

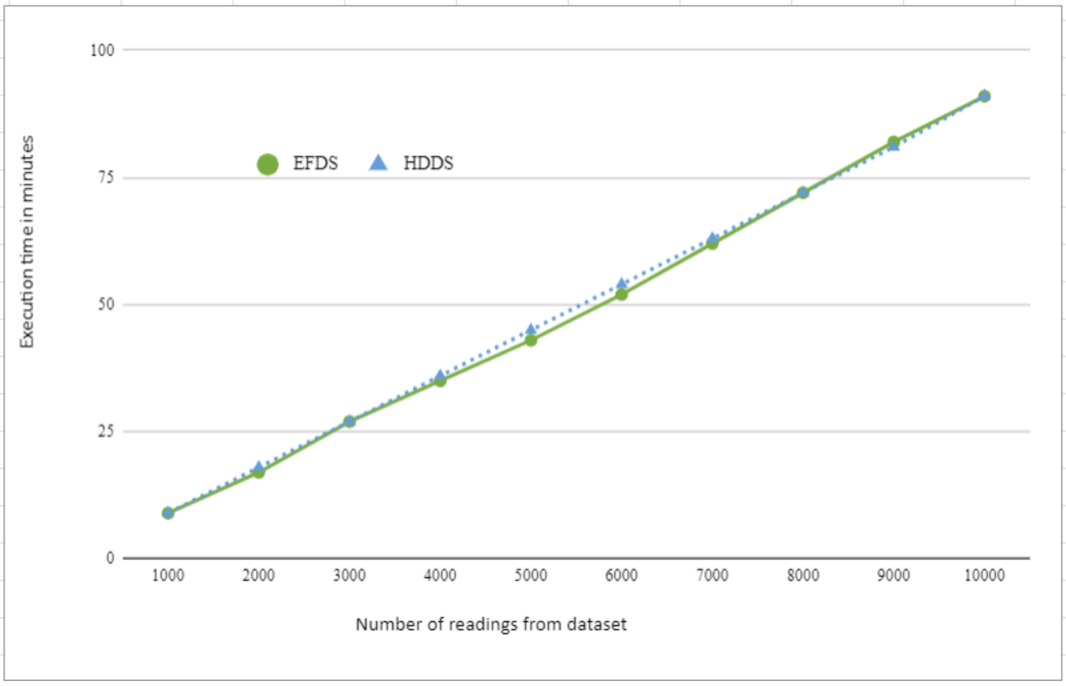

Figure 5.6: Execution Time for Processing Patients with Minor Duplicates

5.9 Execution Time for Processing Patients with Major Duplicates

When high Duplicate values are present in data, maintaining them takes time. In the case of EFDS while processing data, when duplicate values occur, these are added to the duplication reference list. So, when duplication starts to increase, the reference list keeps getting longer as it needs to store references of increasing duplicate values. While HDDS takes only 1 bit for normal duplicate values. So transmitting Boolean digits needs a lower execution time.

The execution time of both schemes is presented in Figure 5.7. When the data readings are 6000 in number, the execution time for EFDS is 62 minutes and 54 for HDDS.

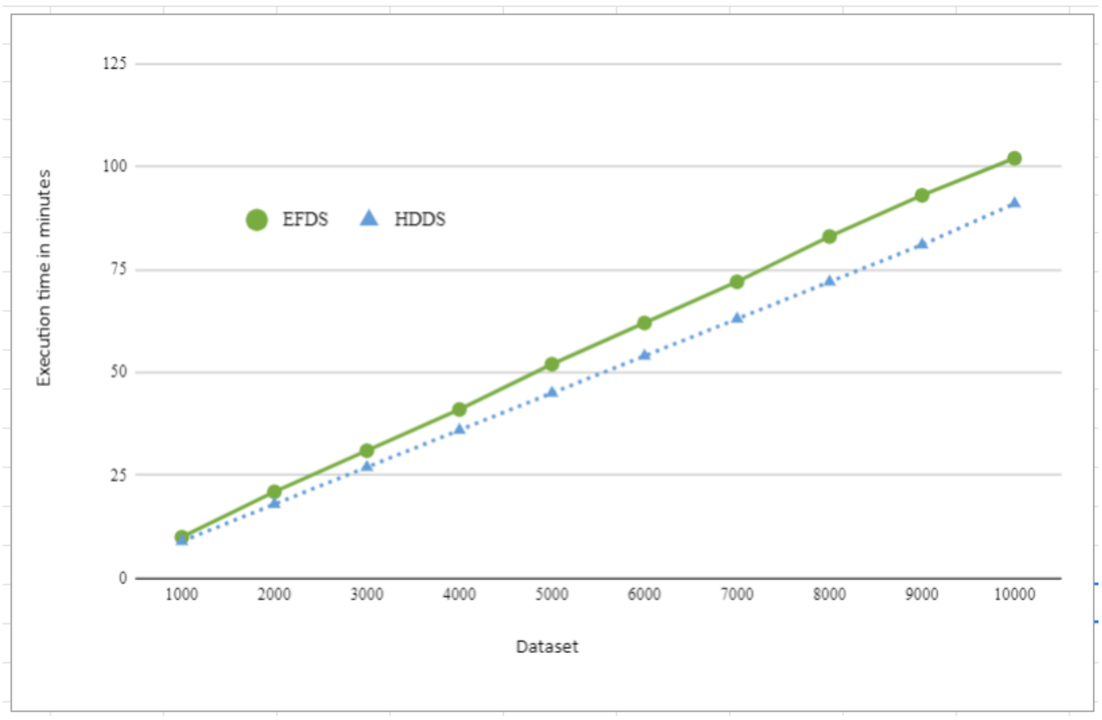

Figure 5.7: Execution Time for Processing Patients with Major Duplicates

5.10 Cumulative Execution Time for Processing Number of Blocks

For this metric, the performance of both schemes is noticed with time. The graph is cumulative and showed progress collectively after sometime. At the start, EFDS maintained robust performance as the duplicate reference list is blank at so there is no need to check for duplicated values while processing. However, with time when more data is handled, the number of similar data starts increasing. It causes to increase in the reference list. The list keeps growing in size, as more data is processed. The EFDS needs to check duplicate values in the list with each iteration. This gradually slower performance as the list increases and ultimately increases execution time for EFDS.

On the other hand, HDDS demonstrated reliable performance over time, which shows that rising duplicate data does not affect processing speed. The reason behind this is that duplicated chunks are managed effectively. As compared to EFDS, which had to be looked up in a large reference list. In Figure 5.8, it is shown clearly that HDDS maintained lower execution time and even increased duplicate references.

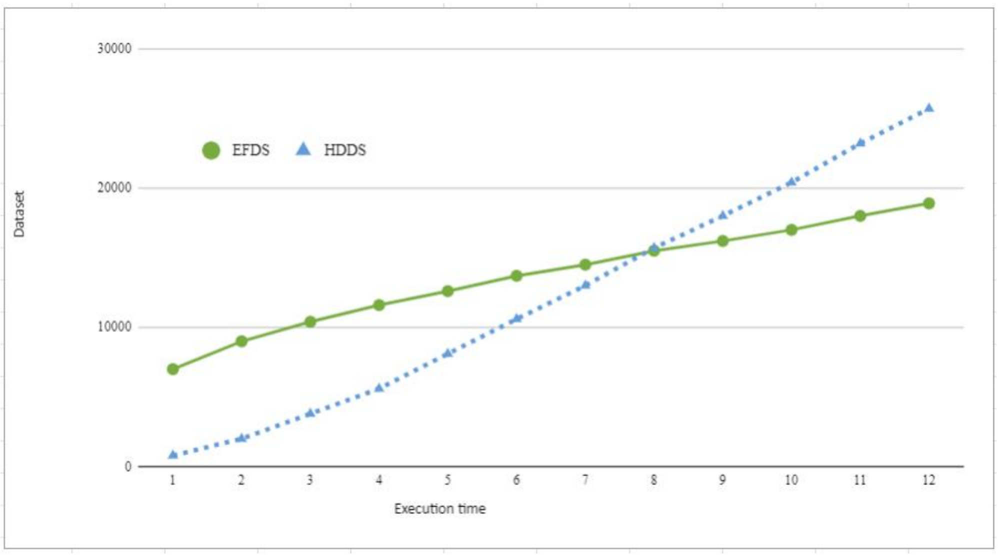

Figure 5.8: Cumulative Execution Time for Processing No. of Blocks per Minute

5.11 Execution Time for Processing of Blocks per Minute

In this graph, processing time of both graphs per minute is shown. Initially, the reference list is empty thus there is nothing to check in the list, therefore EFDS processed data quickly, resulting achieved better performance. When time passes more data is processed, more duplication occurs, and ultimately reference list surges. EFDS needs to search within its large reference list to process duplicate data. This exhaustive processing degrades EFDS performance. While HDDS mechanism to handle duplicates is simple and easy. Due to it, HDDS maintains relatively consistent performance. Figure 5.9 shows showing progress of both schemes per minute. While EFDS performance has fluctuations while processing duplicate data.

In Figure 5.9, performances of both schemes are shown. In 4 minutes, the data size of 1800 is processed in HDDS while EFDS has processed 1400 data readings in 4 minutes.

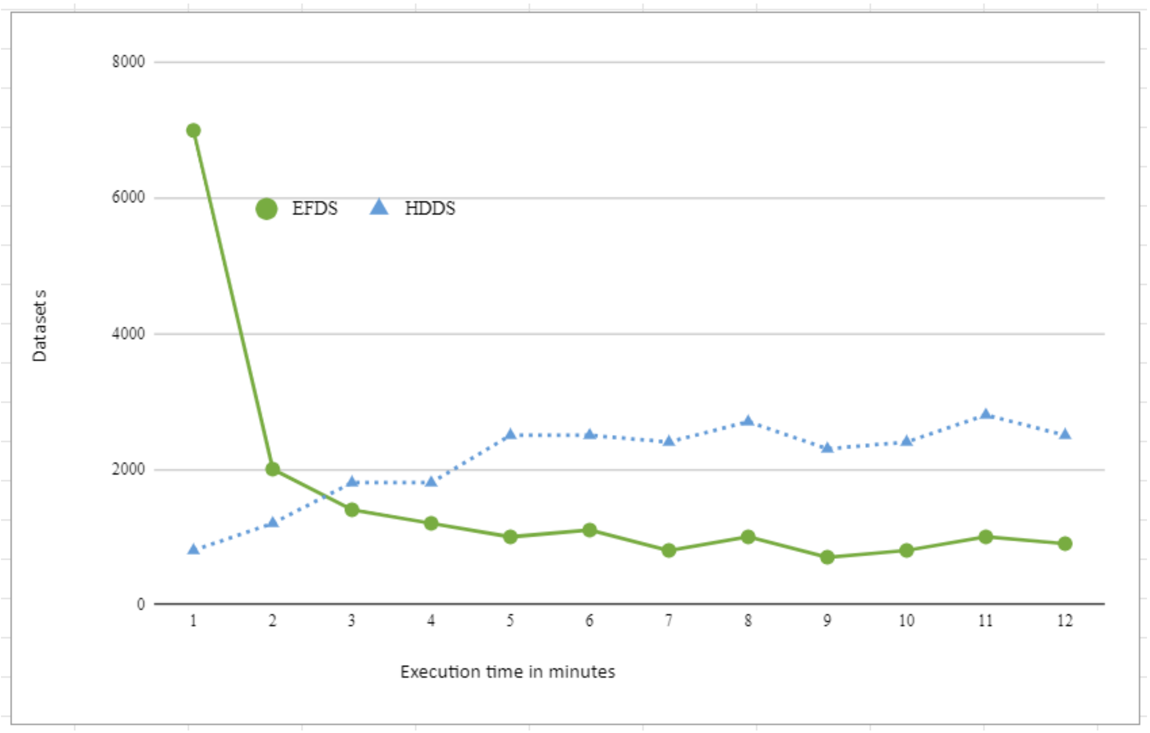

Figure 5.9 Execution Time for Processing No. of Blocks per Minute

5.12 Discussion

The proposed technique HDDS aims to resolve the issue of data duplication in the context of the healthcare sector. To overcome this problem, data values are checked against standard threshold values. If the values have no significant change then it is simply replaced by a smaller chuck of Boolean digit 1, which untimely reduces data packet size and overall increases the efficacy of the proposed scheme. To check its effectiveness, it is compared with another deduplication-based scheme based on some vital evaluation parameters. The proposed scheme maintains a consistent list of meta-data regardless of environment variation and repetitive data. The proposed scheme considers healthcare sector data generated by approximately 73 vital devices that transmit the data after a 1-second interval. This data is effectively compressed and takes little time to transmit as compared to the existing scheme EFDS. The EFDS scheme takes extra time to look at, updating the reference list every time duplication occurs in healthcare data. The updating reference list takes extra time for computation and increases the complexity. Besides this, as the number of vital devices increases the effectiveness of EFDS starts falling due to intensive computation.

5.13 Summary

The HDDS technique is introduced to resolve de-duplication issues without utilizing too much time for Average Execution, reduce the size of data after Post Execution, and compress patient data file size, cumulative execution time, and execution time per minute. The simulation is performed in a Python environment to test the competency of the proposed scheme against another de-duplication solution EFDS. The simulation results proved that HDDS outperformed EFDS.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Overview

This chapter aims to provide a summary of this research and future plans. The primary focus of this work is to overcome duplication issues in data of the healthcare sector and introduce a de-duplication mechanism so that crucial data is maintained well in the cloud repository. To evaluate the efficiency of the proposed technique HDDS, a simulation method is employed. For comparison with the base scheme, the most crucial parameters containing Average Execution Time, Compression Rate, and Size of Data after Post Execution are considered. By comparing the results of HDDS with earlier schemes, it is proved that HDDS has performed well.

6.2 Summary of Research Work

The Internet of Things (IoT) technology has brought revolution in the context of healthcare by collecting data from various wearable devices to a central database system. Though it has various advantages it also raises some challenges that need to be resolved to get full advantage from it. When vital devices take readings from a patient's organs and send them to a fog server, they contain data in different formats. Mostly, normal values are transmitted over 1 second, which unnecessarily takes up storage space. To overcome this issue, the proposed scheme Healthcare Data De-duplication Scheme (HDDS) is presented. The scheme reduces data size by replacing repetitive normal data with digit 1 of the Boolean system. So for such values, the idea of content-defined chunking is utilized. A small chunk of digit 1 is transmitted and its meta-data is maintained for future use. For such values, those are fluctuating and showing abnormal behavior are transmitted in their unique format. So finally the de-duplicated data is stored in the cloud repository. It not only resolves the issue of duplication but also reduces data size ultimately overcoming the issue of higher Execution Time, large size of Data after Post Execution, and compressed patient data file size. Some limitations exist in this thesis. The proposed scheme performs well for de-duplication but the security mechanism to keep patients medical history as confidential is not considered in this research. The communication cost after the de-duplication procedure is not extensively analyzed in this work. The vital data of a few patients are considered in this study, increasing the number of patients might raise new issues. Till now, no strategy has been considered to handle lost or missing data values.

6.3 Future Work

In the future, the number of patients will be increased to check the performance of the proposed scheme in a highly diverse environment. Some security mechanisms will be introduced to protect patients' medical history. Some more advanced evaluation parameters will be used to check the effectiveness of the proposed scheme. Beyond this, the HDDS scheme is compared with more deduplication-based techniques.

GLOSSARY

1. Internet of Things (IoT): Interconnected devices

2. Content-Defined Chunking: Dynamic data partitioning

3. Replication Factor: Redundancy level

4. Infrastructure as a Service (IaaS): Cloud computing foundation

5. Platform as a Service (PaaS): Cloud platform provision

6. Function as a Service (FaaS): Serverless computing

7. Software as a Service (SaaS): Cloud-based software

8. Virtual Machines: Software-based emulation

9. Per-File Parity: File-level redundancy

10. Data Elimination Ratio: Redundancy reduction efficiency

11. Jump-Based Chunking: Data partitioning method

12. Rapid Asymmetric Maximum: Encryption algorithm

13. Healthcare Data De-duplication Scheme: Medical data redundancy removal

14. Efficient File-Level De-Duplication Scheme: Optimal file redundancy elimination

15. Message Locked Encryption: Secure message encoding

16. Secure Hash Algorithm: Cryptographic hash function

17. Message Digest: Compact hash value

18. Cloud Service Provider: Cloud computing vendor

19. Electronic Health Record: Digital health information

20. Multi-Level Encryption: Layered data protection

21. Elliptic-Curve Cryptography: Cryptographic curve method

22. Replicator node: Copying server

23. Co-ordinator node: Central control point

24. Bandwidth: Data transfer capacity

25. Fog servers: Edge computing servers

26. Computational overhead: Processing resource burden

27. Hash chunk: Hashed data segment

28. Hash table: Data structure for hashing

29. Index structure: Organized data index

30. Load factor: System resource utilization

31. Cut points: Data segmentation points

32. Byte pair comparison: Binary unit matching

33. Break points: Discontinuity locations

34. Cosine-based Similarity checking: Similarity measurement method

35. Linear hash function: Straightforward hashing

36. Cipher text: Encrypted data

37. Radix Trie: Tree data structure

38. CEK mechanism: Cryptographic key management

REFERENCES

[1] M. Wada and N. Tanaka, “Big data: survey, technologies, opportunities, and challenges,” *Sci. world J.*, vol. 29, no. 8, p. 1497, 2016, doi: 10.1143/JJAP.29.L1497.

[2] R. Kaur, I. Chana, and J. Bhattacharya, “Data deduplication techniques for efficient cloud storage management: a systematic review,” *J. Supercomput.*, vol. 74, no. 5, pp. 2035–2085, 2018, doi: 10.1007/s11227-017-2210-8.

[3] K. Vijayalakshmi and V. Jayalakshmi, “Analysis on data deduplication techniques of storage of big data in cloud,” *Proc. - 5th Int. Conf. Comput. Methodol. Commun. ICCMC 2021*, Iccmc, pp. 976–983, 2021, doi: 10.1109/ICCMC51019.2021.9418445.

[4] J. Li, Z. Yang, Y. Ren, P. P. C. Lee, and X. Zhang, “Balancing storage efficiency and data confidentiality with tunable encrypted deduplication,” *Proc. 15th Eur. Conf. Comput. Syst. EuroSys 2020*, pp. 1–15, 2020, doi: 10.1145/3342195.3387531.

[5] D. P. Sinha, G. R, Thwel, T. T., Mohdiwale, S., Shrivastava, “Performance Analysis of Secure Cloud Storage by Using Data De-Duplication,” *Data Deduplication Approaches. Acad. Press*, vol. 13, pp. 1–15, 2021.

[6] S. S. Patra, S. Jena, J. R. Mohanty, and M. K. Gourisaria, “Concepts, strategies, and challenges of data deduplication." Data Deduplication Approaches.,” *Data Deduplication Approaches Concepts, Strateg. Challenges*, pp. 37–55, 2020, doi: 10.1016/B978-0-12-823395-5.00009-4.

[7] B. Alami Milani and N. Jafari Navimipour, “A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions,” *J. Netw. Comput. Appl.*, vol. 64, pp. 229–238, 2016, doi: 10.1016/j.jnca.2016.02.005.

[8] K. M. Et. al., “Deduplication Supporting Strong Privacy Protection for Cloud Storage,” *Turkish J. Comput. Math. Educ.*, vol. 12, no. 2, pp. 2334–2340, 2021, doi: 10.17762/turcomat.v12i2.1981.

[9] A. Bhalerao and A. Pawar, “A survey: On data deduplication for efficiently utilizing cloud storage for big data backups,” *Proc. - Int. Conf. Trends Electron. Informatics, ICEI 2017*, vol. 15, pp. 933–938, 2018, doi: 10.1109/ICOEI.2017.8300844.

[10] J. Cable, D. Gregory, L. Izhikevich, and Z. Durumeric, “Stratosphere: Finding vulnerable cloud storage buckets,” *ACM Int. Conf. Proceeding Ser.*, vol. 11, pp. 399–411, 2021, doi: 10.1145/3471621.3473500.

[11] A. Tahir *et al.*, “A systematic review on cloud storage mechanisms concerning e-healthcare systems,” *Sensors (Switzerland)*, vol. 20, no. 18, pp. 1–32, 2020, doi: 10.3390/s20185392.

[12] N. M. Abdulkareem, S. R. M. Zeebaree, M. A. M. Sadeeq, D. M. Ahmed, A. S. Sami, and R. R. Zebari, “IoT and Cloud Computing Issues, Challenges and Opportunities: A Review,” *Qubahan Acad. J.*, vol. 1, no. 2, pp. 1–7, 2021, doi: 10.48161/qaj.v1n2a36.

[13] T. Alam, “Cloud Computing and its role in the Information Technology,” *IAIC Trans. Sustain. Digit. Innov.*, vol. 1, no. 2, pp. 108–115, 2020, doi: 10.34306/itsdi.v1i2.103.

[14] W. Ahmad, A. Rasool, A. R. Javed, T. Baker, and Z. Jalil, “Cyber security in IoT-based cloud computing: A comprehensive survey,” *Electron.*, vol. 11, no. 1, pp. 1–34, 2022, doi: 10.3390/electronics11010016.

[15] P. J. Sun, “Security and privacy protection in cloud computing: Discussions and challenges,” *J. Netw. Comput. Appl.*, vol. 160, p. 102642, 2020, doi: 10.1016/j.jnca.2020.102642.

[16] U. A. Butt *et al.*, “A review of machine learning algorithms for cloud computing security,” *Electron.*, vol. 9, no. 9, pp. 1–25, 2020, doi: 10.3390/electronics9091379.

[17] M. O. Agbaje, O. B. Ohwo, T. G. Ayanwola, and O. Olufunmilola, “A Survey of Game-Theoretic Approach for Resource Management in Cloud Computing,” *J. Comput. Networks Commun.*, vol. 11, no. 8, p. 13, 2022, doi: 10.1155/2022/9323818.

[18] J. Liu, H. Shen, H. S. Narman, W. Chung, and Z. Lin, “A Survey of Mobile Crowdsensing Techniques,” *ACM Trans. Cyber-Physical Syst.*, vol. 2, no. 3, 2018, doi: 10.1145/3185504.

[19] Q. Yang, R. Jin, and M. Zhao, “SmartDedup: Optimizing deduplication for resource-constrained devices,” *Proc. 2019 USENIX Annu. Tech. Conf. USENIX ATC 2019*, vol. 11, no. 3, pp. 633–646, 2019.

[20] M. K. Abiodun, J. B. Awotunde, R. O. Ogundokun, E. A. Adeniyi, and M. O. Arowolo, “Security and Information Assurance for IoT-Based Big Data,” *Stud. Comput. Intell.*,

vol. 972, pp. 189–211, 2021, doi: 10.1007/978-3-030-72236-4_8.

[21] P. Chemate, N. Karande, A. Patil, S. Patil, and P. V. Nale, “Ensuring Data Security in Internet of Things Through Threshold Cryptography and Bloom Filters for Data De-Duplication Purposes,” *Int. Res. J. Mod. Eng. Technol. Sci.*, no. 06, pp. 161–168, 2023, doi: 10.56726/irjmets41402.

[22] S. Li, T. Lan, B. Balasubramanian, H. W. Lee, M. R. Ra, and R. K. Panta, “Pushing Collaborative Data Deduplication to the Network Edge: An Optimization Framework and System Design,” *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2110–2122, 2022, doi: 10.1109/TNSE.2022.3155357.

[23] S. Mishra and S. Panda, “Cloud Computing: Applications, Challenges and Open Issues,” vol. 9, no. 4, pp. 9–10, 2023.

[24] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami, and M. Ayaz, “A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies,” *IEEE Access*, vol. 9, pp. 57792–57807, 2021, doi: 10.1109/ACCESS.2021.3073203.

[25] H. Tabrizchi, “A survey on security challenges in cloud computing: issues, threats, and solutions,” *J. Supercomput.*, vol. 76, no. 12, pp. 9493–9532, 2020.

[26] G. Said *et al.*, “Hash Table Assisted Efficient File Level De-Duplication Scheme in SD-IoV Assisted Sensing Devices,” *Intell. Autom. Soft Comput.*, vol. 0, no. 0, pp. 1–17, 2023, doi: 10.32604/iasc.2023.036079.

[27] K. Pawar, V. Phatale, R. Kumari, A. Kanade, and S. Ujgare, “Data De-Duplication Engine for Efficient Storage Management,” *Int. Res. J. Eng. Technol.*, vol. 9, no. 6, pp. 702–709, 2022, [Online]. Available: www.irjet.net

[28] R. Movaliya and H. Shah, “A Survey of Secure Data Deduplication,” *Int. J. Comput. Appl.*, vol. 138, no. 11, pp. 6–8, 2022, doi: 10.5120/ijca2016908986.

[29] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami, *Data replication schemes in cloud computing: a survey*, vol. 24, no. 3. Springer US, 2021. doi: 10.1007/s10586-021-03283-7.

[30] X. Zou, J. Yuan, P. Shilane, W. Xia, H. Zhang, and X. Wang, “From Hyper-dimensional Structures to Linear Structures: Maintaining Deduplicated Data’s Locality,” *ACM Trans.*

Storage, vol. 18, no. 3, 2022, doi: 10.1145/3507921.

[31] A. M. and D. S. B., "Security Analysis and Preserving Block-Level Data DE-duplication in Cloud Storage Services," *J. Trends Comput. Sci. Smart Technol.,* vol. 2, no. 2, pp. 120–126, 2020, doi: 10.36548/jtcsst.2020.2.006.

[32] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog Computing in Healthcare-A Review and Discussion," *IEEE Access,* vol. 5, pp. 9206–9222, 2017, doi: 10.1109/ACCESS.2017.2704100.

[33] J. Scheuner and P. Leitner, "Performance benchmarking of infrastructure-as-a-service (IaaS) clouds with cloud workbench," *Proc. - 2019 IEEE 4th Int. Work. Found. Appl. Self* Syst. FAS*W 2019,* vol. 11, no. 7, pp. 257–258, 2019, doi: 10.1109/FAS-W.2019.00070.

[34] Z. Li, "Using public and free platform-as-a-service (PaaS) based lightweight projects for software architecture education," *Proc. - Int. Conf. Softw. Eng.,* pp. 1–11, 2020, doi: 10.1145/3377814.3381704.

[35] J. Logeshwaran, "The control and communication management for ultra dense cloud system using fast Fourier algorithm," *Ictact J. Data Sci. Mach. Learn.,* vol. 3, no. 2, pp. 281–284, 2022.

[36] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," *Proc. Annu. Int. Symp. Microarchitecture, MICRO,* vol. 10, no. 7, pp. 1063–1075, 2019, doi: 10.1145/3352460.3358296.

[37] G. Yugansh and G. Bhavana, "A Comprehensive Survey Of Infrastructure As A Service By Top Public Cloud Vendors," *Int. J. Creat. Res. Thoughts - Ijcrt,* vol. 10, no. 11, pp. 888–895, 2022, [Online]. Available: https://www.ijcrt.org/papers/IJCRT2211217.pdf

[38] R. Dhaya, R. Kanthavel, and K. Venusamy, "Dynamic secure and automated infrastructure for private cloud data center," *Ann. Oper. Res.,* vol. 326, no. 1, p. 127, 2021, doi: 10.1007/s10479-021-04442-0.

[39] P. Trakadas *et al.*, "Hybrid clouds for data-intensive, 5G-enabled IoT applications: An overview, key issues and relevant architecture," *Sensors (Switzerland),* vol. 19, no. 16, 2019, doi: 10.3390/s19163591.

[40] G. A., S. S., G. M., T. V., and J. P., "A detailed study of various challenges in cloud

computing," *IIOAB J.*, vol. 10, no. 2, pp. 18–26, 2019.

[41] Y. Perwej, K. Haq, F. Parwej, and M. M., "The Internet of Things (IoT) and its Application Domains," *Int. J. Comput. Appl.*, vol. 182, no. 49, pp. 36–49, 2019, doi: 10.5120/ijca2019918763.

[42] M. Lombardi, F. Pascale, and D. Santaniello, "Internet of things: A general overview between architectures, protocols and applications," *Inf.*, vol. 12, no. 2, pp. 1–21, 2021, doi: 10.3390/info12020087.

[43] R. Asif and S. R. Hassan, "IoT Exploring the Confluence of IoT and Metaverse : Future Opportunities and Challenges," *Sensors*, vol. 4, no. 3, pp. 412–429, 2023.

[44] K. T. Kadhim, A. M. Alsahlany, S. M. Wadi, and H. T. Kadhum, "An Overview of Patient's Health Status Monitoring System Based on Internet of Things (IoT)," *Wirel. Pers. Commun.*, vol. 114, no. 3, pp. 2235–2262, 2020, doi: 10.1007/s11277-020-07474-0.

[45] A. A. Pise *et al.*, "Enabling Artificial Intelligence of Things (AIoT) Healthcare Architectures and Listing Security Issues," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–14, 2022, doi: 10.1155/2022/8421434.

[46] K. Nandhini and V. Vidhya, "An Alleviation of Cloud Congestion Analysis of Fluid Retrial User on Matrix Analytic Method in IoT-based Application," *J. Niger. Soc. Phys. Sci.*, vol. 5, p. 1148, 2023, doi: 10.46481/jnsps.2023.1148.

[47] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 964–975, 2018, doi: 10.1016/j.future.2016.11.031.

[48] S. Hamdan, M. Ayyash, and S. Almajali, "Edge-computing architectures for internet of things applications: A survey," *Sensors (Switzerland)*, vol. 20, no. 22, pp. 1–52, 2020, doi: 10.3390/s20226441.

[49] S. Mishra and A. K. Tyagi, "The Role of Machine Learning Techniques in Internet of Things-Based Cloud Applications," *Internet of Things*, no. 7, pp. 105–135, 2022, doi: 10.1007/978-3-030-87059-1_4.

[50] P. M. Kumar, G. Usha Devi, S. Basheer, and P. Parthasarathy, "A study on data de-duplication schemes in cloud storage," *Int. J. Grid Util. Comput.*, vol. 11, no. 4, pp. 509–

516, 2020, doi: 10.1504/IJGUC.2020.108450.

[51] S. Ahmed, M. F. Hossain, M. S. Kaiser, M. B. T. Noor, M. Mahmud, and C. Chakraborty, “Artificial Intelligence and Machine Learning for Ensuring Security in Smart Cities,” *Adv. Sci. Technol. Secur. Appl.*, pp. 23–47, 2021, doi: 10.1007/978-3-030-72139-8_2.

[52] J. Gnana Jeslin and P. Mohan Kumar, “Decentralized and Privacy Sensitive Data De-Duplication Framework for Convenient Big Data Management in Cloud Backup Systems,” *Symmetry (Basel).*, vol. 14, no. 7, pp. 1–20, 2022, doi: 10.3390/sym14071392.

[53] N. A. Et al., “An Enhanced Approach to Improve the Security and Performance for Deduplication,” *Turkish J. Comput. Math. Educ.*, vol. 12, no. 6, pp. 2866–2882, 2021, doi: 10.17762/turcomat.v12i6.5797.

[54] S. M. A. Mohamed and Y. Wang, “A survey on novel classification of deduplication storage systems,” *Distrib. Parallel Databases*, vol. 39, no. 1, pp. 201–230, 2021, doi: 10.1007/s10619-020-07301-2.

[55] C. Prathima, N. B. Muppalaneni, and K. G. Kharade, “Deduplication of IoT Data in Cloud Storage,” *Mach. Learn. Internet Things Soc. Issues*, vol. 9, pp. 147–157, 2022, doi: 10.1007/978-981-16-5090-1_13.

[56] A. Nauman, Y. A. Qadri, M. Amjad, Y. Bin Zikria, M. K. Afzal, and S. W. Kim, “Multimedia internet of things: A comprehensive survey,” *IEEE Access*, vol. 8, pp. 8202–8250, 2020, doi: 10.1109/ACCESS.2020.2964280.

[57] B. Bhushan, A. Kumar, A. K. Agarwal, A. Kumar, P. Bhattacharya, and A. Kumar, “Towards a Secure and Sustainable Internet of Medical Things (IoMT): Requirements, Design Challenges, Security Techniques, and Future Trends,” *Sustain.*, vol. 15, no. 7, 2023, doi: 10.3390/su15076177.

[58] H. Vuong, H. Nguyen, and L. Tran, “A Design of Parallel Content-Defined Chunking System Using Non-Hashing Algorithms on FPGA,” *IEEE Access*, vol. 10, no. June, pp. 82036–82048, 2022, doi: 10.1109/ACCESS.2022.3196775.

[59] R. N. S. Widodo, H. Lim, and M. Atiquzzaman, “A new content-defined chunking algorithm for data deduplication in cloud storage,” *Futur. Gener. Comput. Syst.*, vol. 71, pp. 145–156, 2017, doi: 10.1016/j.future.2017.02.013.

[60] W. Xia *et al.*, “The design of fast content-defined chunking for data deduplication based

storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2017–2031, 2020, doi: 10.1109/TPDS.2020.2984632.

[61] R. B. B, S. Y. T, and M. U. S. A, "A Secured Cloud Storage Scheme Using ECC-IRNS Based Deduplication Approach in IoT Networks," *J. Xidian Univ.*, vol. 17, no. 8, 2023, doi: 10.37896/jxu17.8/044.

[62] B. Mi, Y. Li, H. Darong, T. Wei, and Q. Zou, "Secure data de-duplication based on threshold blind signature and bloom filter in internet of things," *IEEE Access*, vol. 8, pp. 167113–167122, 2020, doi: 10.1109/ACCESS.2020.3023750.

[63] M. A. S. M. Saeed and L. E. George, "Data Deduplication System Based on Content-Defined Chunking Using Bytes Pair Frequency Occurrence," *symmetry-MDPI*, vol. 12, no. 10, pp. 1–20, 2020.

[64] A. Ahmed, S. Abdullah, M. Bukhsh, I. Ahmad, and Z. Mushtaq, "An Energy-Efficient Data Aggregation Mechanism for IoT Secured by Blockchain," *IEEE Access*, vol. 10, pp. 11404–11419, 2022, doi: 10.1109/ACCESS.2022.3146295.

[65] X. Jin, H. Liu, C. Ye, X. Liao, H. Jin, and Y. Zhang, "Accelerating Content-Defined Chunking for Data Deduplication Based on Speculative Jump," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 9, pp. 1–12, 2023, doi: 10.1109/tpds.2023.3290770.

[66] Y. Zhou, "An efficient encrypted deduplication scheme with security-enhanced proof of ownership in edge computing," *BenchCouncil Trans. Benchmarks, Stand. Eval.*, vol. 2, no. 2, p. 100062, 2022.

[67] Y. Wang, M. Miao, J. Wang, and X. Zhang, "Secure deduplication with efficient user revocation in cloud storage," *Comput. Stand. Interfaces*, vol. 78, p. 103523, 2021, doi: 10.1016/j.csi.2021.103523.

[68] L. Lin, Y. Deng, Y. Zhou, and Y. Zhu, "InDe: An Inline Data Deduplication Approach via Adaptive Detection of Valid Container Utilization," *ACM Trans. Storage*, vol. 19, no. 1, 2023, doi: 10.1145/3568426.

[69] S. Wu *et al.*, "Improving reliability of deduplication-based storage systems with per-file parity," *Proc. IEEE Symp. Reliab. Distrib. Syst.*, vol. 2019-Octob, pp. 171–180, 2019, doi: 10.1109/SRDS.2018.00028.

[70] S. E. Ebinazer, N. Savarimuthu, and S. Mary Saira Bhanu, "An efficient secure data

deduplication method using radix trie with bloom filter (SDD-RT-BF) in cloud environment,” *Peer-to-Peer Netw. Appl.*, vol. 14, no. 4, pp. 2443–2451, 2021, doi: 10.1007/s12083-020-00989-0.

[71] Z. Xu and W. Zhang, “QuickCDC: A Quick Content Defined Chunking Algorithm Based on Jumping and Dynamically Adjusting Mask Bits,” *19th IEEE Int. Symp. Parallel Distrib. Process. with Appl.*, pp. 288–299, 2021, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00049.

[72] H. Kwon, C. Hahn, K. Kang, and J. Hur, “Secure deduplication with reliable and revocable key management in fog computing,” *Peer-to-Peer Netw. Appl.*, vol. 12, no. 4, pp. 850–864, 2019, doi: 10.1007/s12083-018-0682-9.

[73] B. Rasina Begum and P. Chitra, “SEEDDUP: A Three-Tier SEcurE Data DedUPlication Architecture-Based Storage and Retrieval for Cross-Domains Over Cloud,” *IETE J. Res.*, vol. 69, no. 4, pp. 2224–2241, 2023, doi: 10.1080/03772063.2021.1886882.

[74] L. Xiao, B. Zou, C. Zhu, and F. Nie, “Reduced data : An efficient and secure deduplication scheme based on data similarity and blockchain for cloud-assisted medical storage systems,” *J. Supercomput.*, vol. 79, no. 3, pp. 2932–2960, 2023, doi: 10.1007/s11227-022-04746-3.

[75] L. Xiao, B. Zou, and X. Kui, “A Secure Lossless Redundancy Elimination Scheme With Semantic Awareness for Cloud-Assisted Health Systems,” *IEEE Syst. J.*, vol. 17, no. 3, pp. 4615–4626, 2023, doi: 10.1109/JSYST.2023.3253690.

[76] T. Benil and J. Jasper, “Blockchain based secure medical data outsourcing with data deduplication in cloud environment,” *Comput. Commun.*, vol. 209, no. 6, 2022, pp. 1–13, 2023, doi: 10.1016/j.comcom.2023.06.013.

[77] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, “HealthDep: An Efficient and Secure Deduplication Scheme for Cloud-Assisted eHealth Systems,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 9, pp. 4101–4112, 2018, doi: 10.1109/TII.2018.2832251.

[78] P. Puzio, R. Molva, M. Önen, and S. Loureiro, “Perfectdedup: Secure data deduplication,” *Data Priv. Manag. Secur. Assur. 10th Int. Work. DPM 2015, 4th Int. Work. QASA 2015, Vienna, Austria*, vol. 9481, no. 644412, pp. 150–166, 2016, doi: 10.1007/978-3-319-29883-2_10.

[79] A. S. M. Saeed and L. E. George, “Fingerprint-based data deduplication using a mathematical bounded linear hash function,” *Symmetry (Basel).*, vol. 13, no. 11, 2021, doi: 10.3390/sym13111978.