

**A NOVEL SCHEME FOR THE
FRAGMENTATION, ALLOCATION AND
DEDUPLICATION OF DATA TO ACHIEVE
PERFORMANCE IN THE DISTRIBUTED SYSTEMS**

Thesis Submitted for the Award of the Degree of

DOCTOR OF PHILOSOPHY

in

Computer Science Engineering

By

Sashi Tarun

Registration Number: 41800190

Supervised By

Prof. (Dr.) Mithilesh Kumar Dubey

School of Computer Applications

Lovely Professional University



Transforming Education Transforming India

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2023

DECLARATION

I declare that this thesis entitled “A Novel Scheme for the Fragmentation, Allocation and Deduplication of Data to Achieve Performance in the Distributed Systems” has been prepared by me under the guidance of Prof. (Dr.) Mithilesh Kumar Dubey, Professor, School of Computer Applications, Lovely Professional University. No part of this thesis has been formed the basis for the award of any degree or fellowship previously.

Sashi Tarun

School of Computer Science and Engineering,
Lovely Professional University,
Jalandhar Delhi G.T. Road (NH-1),
Phagwara, Punjab, 144411
India.

Date: 20.04.2023

CERTIFICATE

We certify that Mr. Sashi Tarun has prepared his thesis entitled “A Novel Scheme for the Fragmentation, Allocation and Deduplication of Data to Achieve Performance in the Distributed Systems” for the award of Ph.D. Degree of Lovely Professional University under our guidance. He has carried out work at the Department of Computer Science and Engineering, Lovely Professional University, Phagwara (India).

Supervisor:

Dr. Mithilesh Kumar Dubey

Professor

School of Computer Applications

Lovely Professional University

Phagwara, Punjab-144411, India

Date: 20.02.2023

ABSTRACT

Data is always a priority to all users engaged in their routine operations and in data analysis tasks. Earlier, users switched from a flat file to a centralized system for a better system but they have faced some organizational and administrative difficulties in arranging and making data useful for them and become apparent. In order to maintain existing methodologies relevant and in line with user expectations in the modern era, researchers have attempted to improve them throughout time. However, as data expands, it poses new problems and becomes increasingly challenging to handle. Since the advent of social networking sites and web applications, users have been considerably more involved in the data generation process. Throughout their conversations or exchanges, they frequently exchange data in the audio, video, and textual domains. The data in this case is solely in the control of the users or parties involved, which attracts different hurdles or challenges.

The purpose of this study is to strengthen the data distribution process by concentrating on three key areas. Firstly, it stresses more for storage management by introducing data fragmentation, data allocation based on an optimized cost-based scheme, and data deduplication using advanced machine learning architecture. This work is implemented on textual data by offering a revolutionary text-based data fragmentation technique that follows the threshold segmentation approach. It controls data storage by dividing the data into manageable chunks known as data fragments. Second, data distribution is based on the reducing cost paradigm, which guarantees that customers receive all of their data quickly and at a low cost. On the other side, a deduplication approach can be used to get around the problems that arise with having plenty of data instances. It emphasizes the need to spot duplicate data right away so that it may be added to the storage unit. The new data entry into the storage unit is rejected when duplicate data occurs.

The results obtained in the comparison to the state of art techniques show that the proposed algorithms demonstrated a high degree of accuracy, performance, and achieve data quality.

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. (Dr.) Mithilesh Kumar Dubey for always providing me, with his valuable time and support. Through his immense knowledge and patience, I was able to accomplish all my tasks on time.

I wish to thank Dr. Ranbir Singh Batth for providing his precious time and his enormous knowledge. He has been actively interested in my work and has always been available to advise me. I am very grateful for his patience, motivation, and enthusiasm.

Sincere thanks to Mr. Nitin Dev Sharma for his continuous motivation, making this journey easy.

A warm thanks to all my friends for their endless sacrifices and selfless support that made everything possible on this path.

I also want to mention here my gratitude towards my parents, wife, son, and daughter who went through many ups and downs but still continued to shower their endless love and care which always boosted my morale to carry forward this journey and make my efforts visible.

(Sashi Tarun)

CONTENTS

Declaration.....	ii
Certificate	iii
Abstract.....	iv
Acknowledgment.....	vi
List of Tables	xii
List of Figures.....	xiv
1. Introduction.....	1-18
1.1. Background and Problem Statement	1
1.2. Distributed System Overview.....	2
1.3. Fragmentation of Data.....	5
1.4. Fragmentation: Role in Distributed Systems.....	7
1.5. Allocation of Data	8
1.6. Allocation: Role in Distributed Systems	9
1.7. Deduplication of Data	10
1.8. Deduplication: Role in Distributed Systems	11
1.9. Issues in the Design of Distributed Systems	12
1.10. Challenges in Distributed Systems.....	13
1.11. Major Weaknesses of Distributed Systems	16
1.12. Challenges and Motivations	17
1.13. Organization of the Thesis.....	18
2. Review of Literature	20-44
2.1. Introduction	20
2.2. Existing Work Done in Distributed Design.....	21
2.3. Existing Work Done in Data Fragmentation	23

2.4.	Existing Work Done in Data Allocation	29
2.5.	Existing Work Done in Data Deduplication.....	34
2.6.	Problem Statement	38
2.7.	Research Questions	39
2.8.	Research Gaps	40
2.9.	Objectives of Research Work.....	40
2.10.	Scope of the Proposed Work.....	40
2.11.	Proposed Methodology.....	41
2.11.1.	Pseudo-code for the Data Fragmentation	43
2.11.2.	Pseudo-code for the Data Allocation	43
2.11.3.	Pseudo-code for Data Deduplication.....	43
3.	Design a New Data Fragmentation Architecture Based on a Similarity-Based Threshold Segmentation Method	45-68
3.1.	Introduction.....	45
3.1.1.	Segmentation Technique.....	47
3.1.2.	Similarity Measures.....	47
3.1.2.1.	Cosine Similarity.....	48
3.1.2.2.	Soft Cosine Similarity.....	49
3.1.2.3.	Hybrid Similarity.....	49
3.1.3.	Machine Learning.....	49
3.2.	Methodology and Implementation	50
3.2.1.	Uploading Twitter Data.....	52
3.2.2.	Removal of English Stop Words.....	52
3.2.3.	Applying Word to Vector on Filtered Data.....	54
3.2.4.	Applying Similarity Calculations.....	55

3.2.5. Finding Initial Centroid (IC) of Data	58
3.2.6. Calculation of Euclid Distance.....	59
3.2.7. Fragment Generations	60
3.3. Results and Discussions.....	61
3.4. Comparative Analysis.....	65
3.5. Conclusion	68
4. Introduce a Novel Data Allocation Scheme to Optimize the Allocation Process Based on Swarm Intelligence.....	69-88
4.1. Introduction.....	69
4.2. Proposed Methodology and Implementation	72
4.2.1. Loading of Query File.....	73
4.2.2. Total Cost Calculation.....	74
4.2.2.1. Level-Wise Estimation of Execution Patterns.....	74
4.2.2.2. Execution Cost Patterns Task-Wise	78
4.3. Proposed Artificial Bee Colony (ABC) Algorithm	80
4.3.1. Artificial Bee Colony Pseudocode.....	80
4.3.2. Re-analyse of Optimal Result using Upgraded Threshold.....	81
4.4. Simulation Results and Discussion.....	83
4.5. Conclusion	88
5. Improve the Existing Data Deduplication Scheme with Machine Learning Architecture.....	89-106
5.1. Introduction.....	89
5.2. Proposed Methodology	91
5.2.1. Pre-Processing of Data	92
5.2.1.1. Pseudocode for Pre-Processing Task	93

5.2.2.	Word to Vector (W2V) Conversion	93
5.2.2.1.	Pseudocode for W2V Translation	94
5.2.3.	Clustering of Data.....	94
5.2.4.	Training and Testing Data Generation	95
5.2.4.1.	Pseudocode for Generating Training and Testing Indexes...	95
5.2.5.	Similarity Calculations Cluster-Wise.....	96
5.2.5.1.	Pseudocode for Similarity Calculation Cluster-Wise	96
5.2.5.2.	Pseudocode to Calculate Cosine Similarity	98
5.2.6.	Applying Ground Truth Test (gt-test)	98
5.2.6.1.	Pseudocode to Apply Ground Truth Test	99
5.2.7.	Similarity Calculation of Testing Data.....	99
5.2.7.1.	Pseudocode for testing Data Similarity Calculations	99
5.2.8.	Detecting Duplication by Comparing Similarity-Test Results with Cluster Similarity	100
5.2.8.1.	Pseudocode for Comparison of Similarity test and Cluster Similarity	100
5.3.	Performance Parameter Estimation.....	101
5.3.1.	Pseudocode to Compute Performance Parameters	101
5.4.	Results and Analysis.....	103
5.4.1.	Evaluation of Performance	103
5.5.	Comparative Study	104
5.6.	Conclusion.....	106
6.	Conclusion, Contributions and Applications, and Future Work.....	107-110
6.1.	Conclusion.....	107
6.2.	Contributions and Applications	109

6.3. Future Work.....	110
References	111-127
Research Publications	128

List of Tables

Table	Title	Page
3.1	Fragments Classification Row-Wise	61
3.2	Classification Accuracy of Each Row	62
3.3	Average Percentage of Iterations	63
3.4	Evaluation of Tp, Fp, Tn and Fn	63
3.5	Evaluation of Precision and Recall	64
3.6	Comparative analysis of parameters followed by different fragmentation techniques	65
3.7.	Comparison of calculated Precision and Recall	66
4.1	Execution Cost Matrix (ECM)	71
4.2	Network Cost Matrix (NCM)	71
4.3	Query Model Flow	74
4.4	Estimation of Total System Cost	78
4.5	Execution Cost Pattern for Each Processing Tasks	79
4.6	Fitness Value and Re-Analyse Process for Optimal Results	82
4.7	Comparative analysis of techniques with their cost Usability	83
4.8	Comparative Studies of Proposed and Existing Approaches	84
4.9	Optimal Task Allocation	85
4.10	Evaluation of System Cost with Existing Methods	85
4.11	Before and After Total Cost Results in mJ	86
4.12	Comparison of Methods Based on Reduced Total Cost	87
5.1	Cluster-Wise Distribution of Datasets	94
5.2	Cluster-Wise Similarity Calculation	97
5.3	Performance Parameters	103
5.4	Comparative Analysis of Performance Parameters in Data Deduplication	105

List of Figures

Figure	Title	Page
1.1	Distributed System Architecture	4
1.2	Different Fragmentation Types	6
1.3	Data Insight	10
1.4	Deduplication Mechanism	11
1.5	Research Methodology Flow	42
3.1	Cosine Similarity	48
3.2	Machine Learning Architecture	50
3.3	Data Flow Structure of Fragmentation	51
3.4	Metadata of Twitter Data	52
3.5	Stop Word Removal Process	53
3.6	Stop Word Removal and W2V Conversion Code Snippet	53
3.7	Vector Representation of Filtered Data	54
3.8	(a) Outcome after similarity measure (b) Similarity index graphical representation	58
3.9	Representation of Clusters and Centroids	59
3.10	Measures Distance from the Centroids	60
3.11	Classification Accuracy	62
3.12	True Positive-False Positive-True Negative-False Negative	64
3.13	Precision and Recall	64

3.14	Precision and Recall Comparison	67
4.1	Directed Acyclic Graph	70
4.2	Proposed Model Flow	73
4.3	Results Before and After Optimization of Cost Using ABC	86
4.4	Reduced Total Cost in %age	87
5.1	Data Deduplication Process	90
5.2	Work Flow of Proposed Methodology	92
5.3	Graphical Representation of Clusters and Centroid with Performance Parameters	104
5.4	Comparison of Proposed and Existing Techniques	106

Chapter 1

Introduction

Data Fragmentation, Allocation, and Deduplication are the thrust areas of distributed systems. This work deals with large data storage and stresses methods to control storage having voluminous data, work for data allocation at a reduced cost, and maintain to achieve a single instance of data at distributed geographical sites. Therefore, it is important to have a general understanding of these topics as well as enough basic information on the distribution of data areas. This chapter provides a brief summary of it.

This chapter begins with an overview of distributed systems, a discussion of the background, and a problem statement. It also discusses the issues, challenges, and weaknesses of distributed adoption. Furthermore, the challenges and weaknesses of this research study are discussed, and at last, concluded with thesis organization.

It is further followed by a detailed study of different data distribution paradigms in the next chapter including fragmentation, allocation, and deduplication of data.

1.1. Background and Problem Statement

The continuous evolvement of data and its further utilization on the distributed ground are the chief concern areas. In the modern world, every organization deals with data repository systems like relational databases, data warehouses, spatial databases, etc. However, many of these organizations are unable to fully utilize their vast repositories. However, current trends favor distributed data management to provide performance for a modern workplace.

This research covers three problems of distributed systems dealing with the distribution of data:

- Firstly, it highlights the researcher's aptitude for using empirical data rather than using current. These consequences to radical results and are not helpful for any solution to the problems and impact the performance of the distributed environment. Here, predictions are based on the available data or highlight dependency on data based on human behavior/intentions and old statistical data.
- **Second**, Research done so far disregards network expenses, and earlier methods of allocating data were not financially viable. Data allocation systems must create a way to help them attain low costs when considering network, communication, and computation expenses all at once. This study looks at the three costs involved in implementing cost-based data distribution for distributed systems.
- **Third**, stress on controlling the storage, query efficiency, and data inconsistency issues. These all are impacted by the availability of redundant data at sites. An advanced machine learning architecture is required that is able to trace duplicate entries at storage units to handle data deduplication.

Distributed System Overview

Several definitions are contributed by eminent scientists, authors, designers, and researchers time-to-time for the distributed system. The most prevailing meaning is depicted in this definition:

“A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.”

- By Maarten van Steen & Andrew S. Tanenbaum

Data is a building block and is viewed as an asset to any organization. Data has its own importance with its value and is employed at multiple levels of

granularity during routine data access operations. Existing data storage is rapidly growing in size as a result of the growing use of chat apps, corporate online web applications, and real-time transaction processing, posing a substantial danger to data quality and potentially detracting from current dispersed performance. It allows individuals to tamper with data without thinking about the consequences and has resulted in a slew of bottlenecks. It is vital to organize and manage existing data to attain performance. To do so, first, regulate data storage, then work on the most cost-effective distribution strategy, and finally address data inconsistency, and storage wastage concerns created by redundant data.

Previously, users had to stay in their activity center to get the data or information they needed. Now, anyone in their network can access their resources. Changing technology trends have made it possible for users to spread their data to remote locations and have created a linked configuration to allow equitable access to data on an everywhere, anytime basis. This aids in distributed design efficiency by ensuring parallel data access with high data availability, reliability, consistency, and fault tolerance. All nodes have rapid access to their data, independent of scheduling or access priority.

However, in the technical design of a distributed structure, some minor but substantial bottlenecks have formed, reducing worker performance. We need to work on and debate some of the concerns relating to distributed design before describing such vulnerabilities.

- How may relational schemas be divided into pieces to govern rising data?
- How to allocate fragments into dispersed environments to achieve distribution at minimal cost?
- How to handle duplicate data patterns exists in distributed data architecture?

The goal of this research is to reinforce distributed systems by proposing new strategies for data fragmentation, cost-based optimum data allocation, and sustaining single data instances across geographical locations. It enables distributed systems to be as efficient as possible for a wider range of applications and managed with greater data management, availability, reliability, scalability, and efficiency (MARSE).

With time, scientists, writers, and researchers have made several contributions to the resilient distributed system model, including methods and outcomes. These characteristics can be used to create a dispersed structural design for a distributed system as shown in Figure 1.1.

- It is a combination of nodes or sites working autonomously and having their local memory [1-2].
- All nodes are communicating with each other by different message-passing schemes [3-4].
- All nodes are geographically dispersed in nature [5].

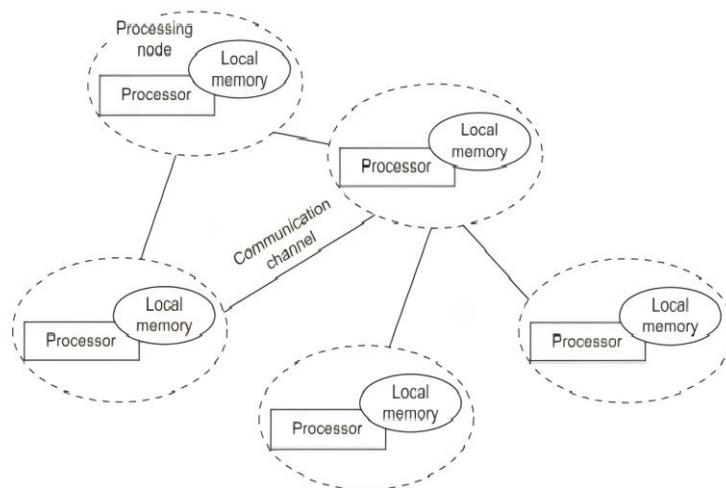


Figure 1.1: Distributed system architecture

Reduced overall performance, interrupted data connections, prolonged wait times, high latency, huge storage to preserve data, and hefty maintenance expenditures are all obstacles that must be overcome [6]. The availability of powerful processors, system memory, increased storage capacity in offline/online modes, sophisticated software, efficient networking capabilities, and flexibility in designing platform-independent distributed application interfaces make the transition from monolithic to distributed computing make possible. A distributed system has a collection of resources that may be accessed from any activity center over the internet with the intent to satisfy clients' needs and meet their business perspectives.

With time, several architectural design models have been offered to strengthen the existing distributed design that determines how data is preserved and accessed conveniently in a geographically scattered network configuration. It is aimed towards enhancing scalability, data processing flexibility, communication efficacy, control efficiency, resilience, and resource sharing.

A smart design takes into account the volume of client interest and suggests how data should be distributed across all sites. Due to a bad design, query processing is delayed, and system performance decreases as a result. When systems respond to their varied questions, provide distributed analytic services, and aid in the construction of effective application interfaces for a range of contexts, end-users are satisfied.

Researchers have made consistent efforts to construct or enhance architectural design in geographically dispersed settings time-to-time. To achieve this, important issues in any serving network include managing data in storage, focusing more on network cost before making any decision/judgments on data placement, and preventing data duplication in distributed storage for the attainment of data quality. The current focus in the design process is to work

for efficient, advanced data fragmentation, allocation, and deduplication methods comparable to existing strategies.

1.2. Fragmentation of Data

Data storage and retrieval depend on the quantity of how the data organized in the server. With the fast development of technology, the requirements of users have also changed. A user who was stationary earlier has become mobile now and requires access to data from anywhere in the world. An unorganized data structure will result in output delay in the network and may further result in user migration from one service provider to another service provider. Data fragmentation is one of the most essential parts when it comes to data storage.

Data fragmentation is an approach used during data distribution in a distributed setup. It is responsible to partition voluminous data into logical data units in geographically dispersed sites in subtables or sub-relations. This partitioned data structure in distributed architecture is termed fragments. This is an attempt to apply cut down on the size of unwanted data accessed and lower the number of disk accesses. Data fragmentation is applied by using one of the approaches including horizontal, vertical, and mixed [7],[8] as shown in figure 1.2.

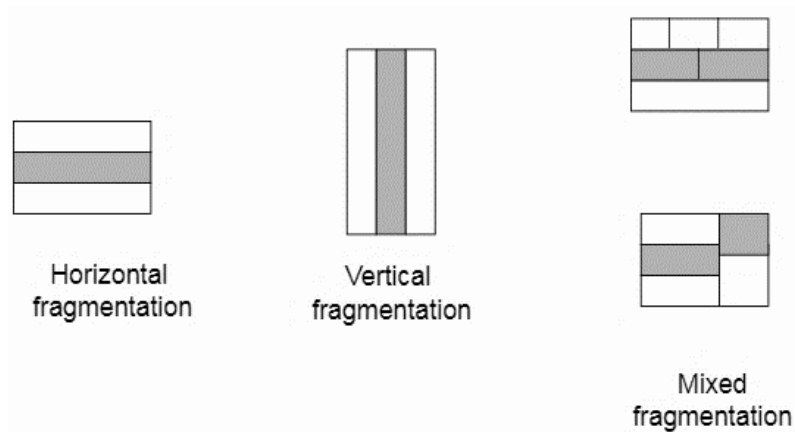


Figure 1.2: Different fragmentation types

- **Horizontal Fragmentation:** It refers to the refinement of global schema records into disjoint tuples. The idea behind horizontal fragmentation is that each site should store all of the data that is used to query the site and that the data should be fragmented so that the site's queries execute more quickly.
- **Vertical Fragmentation:** It allows the global schema to be partitioned into disjoint columns or attributes. In vertical fragmentation, primary key attributes are repeated on all partitioned relations.
- **Hybrid Fragmentation:** The design features of horizontal and vertical fragmentation combine to create a hybrid fragmentation data structure. Such architecture is used to satisfy applications designed for special requirements. To build this type of fragmentation design sub relations are divided into arbitrary blocks.

1.3. Fragmentation: Role in Distributed System

Fragmentation has revolutionized storage management methods for distributed systems. It effectively handles data handling difficulties that have an impact on workforce performance, such as excessive complexity, data inconsistency, and latency concerns. It is an attempt to manage and regulate internal storage on a distributed basis, as well as to introduce it in a normalized manner. The fragmentation approach helps in many ways:

- **Reduce Cost:** Data fragments are reduced-size grouped/clustered data structure that holds relevant data records and satisfies access patterns in the interest of users. It helps to maintain total cost by reducing network cost, computational, and execution costs at the fragment level. Fragments contain fine-grained attributes solely responsible to hold required information used to query the site.

- **Data Placement:** Earlier, data access patterns in a monolithic computing environment are based on block-level and led to performance issues. But fragmentation manages by putting data at geographical sites in sub-relations form to reduce data contention and loading issues. An effective data management strategy will always direct to set better possibilities for the placement of data at network sites.
- **Query Response:** Data response on queries indicates the total time taken during the finding of relevant data information in data fragments at geographical sites. The easy availability of data is depending on defined structural arrangements for fragments in a distributed environment. Clustering enables a group of similar data patterns so that data is acquired at one location or from adjoining sites.
- **Distributed Application Design:** Vertical fragmentation enables designers to use fragment columns to build distributed application interfaces.
- **Storage Management:** Earlier data was structured and managed in single large schemas. Fragmentation gives the flexibility to maintain large relational data into sub-relations and enables them to reduce storage capacity at individual sites. It helps to increase query performance and reduce storage, and data complexity.

1.4. Allocation of Data

The process of fragment allocation to different geographical areas to make data available to intended users is referred to as data allocation. To gain confidence, the allocation approach validated those queries from desired sites must be satisfied at the fragment level. Before assigning data fragments to remote sites, an evaluation of the total cost of data access is required.

To allocate data (fragments) to distinct network nodes, data patterns must be placed at different nodes in such a manner that users can quickly obtain the data or information they want without having to visit numerous network sites. Without the participation of remote nodes, data can be obtained from the requested node or its surrounding linked nodes [9].

The data allocation does not appear to be dependent on load. The distribution of data fragments is determined by two factors: execution and transfer costs. In most situations, execution costs are fixed, but transfer costs vary from one engine to the next. In addition, the overall cost includes certain runtime entity charges. So, data allocation would have to give attention to optimization of total cost during execution, and transfer of data based on data queries rather than allocation of data fragments on the distributed sites.

1.5. Allocation: Role in Distributed System

In a distributed environment, data-fragments allocation is a major concern because it relied on data from multiple sources. Due to dependency on partitioned and replicated allocations large data volumes may now be managed effortlessly across dispersed sites. It is because of the advancement in database and communication technology. On one hand, it decreases data access burden and data complexity, but on the other hand, it causes problems with data retrieval, relevance, and consistency, and has an impact on distributed system performance. To overcome these obstacles, mapping an effective data allocation method with a distributed design that effectively improves the existing system, increases data availability, extends expansion options, decreases access time, and works to minimize overall cost is required. Data allocation roles in distributed systems include:

- a. **Cost Reduction:** An effective data fragment allocation approach can help to lower the total cost of data access. The entire cost spent is comprised of many costs such as communication, computation, and execution.

- b. **Better Data Insight:** A systematic data allocation not only helps for increasing system throughput but also gives better data insight for data analytics operations, and query results at geographical sites. As a whole, it helps to fetch relevant data from large data volumes as shown in figure 1.3 and can be achieved by optimal fragments allocation strategy.

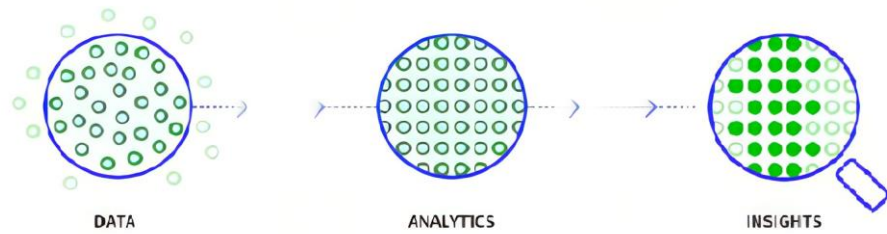


Figure 1.3. Data Insight

- c. **Location Traceability:** Data allocation at geographical sites becomes productive when regular data access operations fetch relevant data at nearby locations. It reduces delays, communication costs, and data loss issues.

1.6. Deduplication of Data

On different grounds, data in distributed systems are equally responsible for the occurrence of duplicate data. It occurs, due to prolong attaining of data in the storage devices, more dependency on social websites for chats, and sharing information or data in the network. The distributed environment shows a wider scope and more chances of finding duplicate data at different locations. The probability of finding errors in one device is less as compared to finding them on a large network space. The post-processing and inline deduplication process is shown below in Figure 1.4. Post-processing detects duplicate data at storage units and maintains the single-instance paradigm of data. On the other side, inline remove the data duplicity before storing the data in storage units in distributed systems.

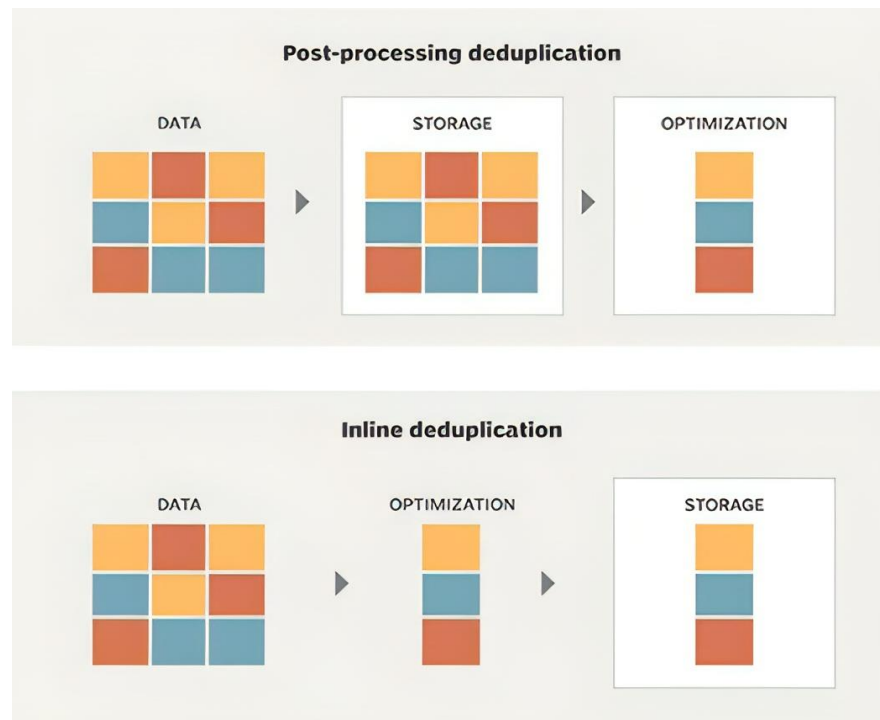


Figure 1.4. Deduplication Mechanism

1.7. Deduplication: Role in Distributed System

Deduplication is an inline as well as a background process that works parallelly to remove multiple copies of data during new data entry or find duplicity of data internally in the existing storage devices. Deduplication has its role in the management of data to make it useful. It has its significance includes:

- Fast Query:** Deduplication helps to execute the tasks or queries for data fastly. It increases the data access rate due to non-redundant data in the storage. Relevancy of data can be achieved and eliminate the use of unwanted data records.
- Save Disk Storage:** It is helpful to save disk storage at large on large data backup files. It maintains a single instance of data and works for data inconsistency issues, controls excessive use of storage capacity, and reduces data access costs to a large extent.

- c. **Data Quality:** It improves data quality in existing storage devices. It works a lot for data analysts, IT managers, and business users involved in the process of drawing future trends based on data for making decisions for company aspects.
- d. **Cost Affective:** It helps to reduce operational and maintenance cost. Earlier data-intensive operations required more hardware equipment, power, cooling, and floor space as well. It is helpful for the retention of hardware for a long period and reduces cost.
- e. **Affective Results:** Interpretation of redundant data always give irrelevant results. Here, deduplication works effectively on the data query model to access required data and its computed results.
- f. **Bandwidth Usage:** It optimizes network communication due to less consumption of bandwidth in regular data transfer. Normally, backup and replication operations required more bandwidth for the accomplishment of their operations.

1.8. Issues in the Design of Distributed System

An issue indicates a lack in the progression of the process involved. So, an Overhaul of existing design strategies will facilitate us to enhance the performance of the distributed environment. Some of the small but major bottlenecks involves in designing distributed systems are:

- a. **Incomplete Network Information:** The lack of knowledge about network infrastructure is a significant barrier to overcome when creating a distributed system. To develop a resilient system, thorough knowledge of architectural design is essential. It also had an impact on the entire network cost calculation.

- b. **Load Balancing:** This is done to ensure that data is distributed evenly across all sites to maximize throughput. In this case, the load balancer satisfies all incoming requests by processing them as quickly as possible. It can be accomplished by intelligently scheduling requests and sending them to the appropriate network nodes. It delivers the load to sites based on the number of users who log in for regular transactions, and new sites are occasionally added to the system, causing the distributed system's status to rapidly change.
- c. **Data Availability and Consistency:** An unplanned architectural design is a hindrance to achieving user satisfaction levels during their regular data access course. The distribution of data over several locations must satisfy a maximum number of responses in the event of high data load, and failure occurs at the node level. Another side, if any data item is changed at any of the activity centers, the same changes must be broadcast to other sites to ensure consistency. Queries can be satisfied in a concurrent environment by using good scheduling or supplying relevant data from low-cost nodes.
- d. **Data Deduplication:** Having duplicate copies of data in geographical sites not only creates confusion or ambiguous situations but also increases existing storage space. It is equally responsible to degrade the performance of the systems because it will take more time to parse each data record in relation. As a result, query results are not showing any relevancy due to data duplicity.

1.9. Challenges in Distributed System

Challenges are the trends and directions that system designers and researchers should follow to continuously improve existing challenges that affect distributed system design, such as:

- a. **Data Incompleteness:** It indicates the non-availability of complete data at nearby sites from the requested site in response to any query. This partial availability of data slows down the query response time and affects system throughput. Such a problem occurs when the data-relevancy factor is not taken into consideration during the designing of distributed architecture.
- b. **Bugs Issues:** Distributed bugs create hurdles in the system performance of the system by decreasing the response time of any request. These bugs spread themselves like an epidemic within the network of computers and slow down data flow in the channel. They make the whole network busy and affect network efficiency.
- c. **Implementation Cost:** Designing and implementing a distributed system is very expensive and not bearable for small organizations thinking to avail or hiring their services.
- d. **Network Untrustworthiness:** It disrupts the communication progress between two or more ends during the fetching of data. This unreliability of the network indicates:
 - i. The request may be queued and has to wait for their turn.
 - ii. Chances of request loss in the network.
 - iii. Response node may have any fault, or failure situation.
 - iv. Network overloading results delayed in response.
 - v. Network partition problems divide the whole network into two fragments/partitions. As a result, the response from the other side is not able to reach the source, where the request generates.
- e. **Latency:** In a widely distributed network, deterioration in data during the transferring of data or information from one place to another indicates a latency issue. As enterprises thought about migration of their valuable services into distributed environment structure but network latency issue

not able to lead this operation because migration activity involves backup and restore operations and found not flexible in disaster recovery in an emergency due to its speed.

- f. **Data Synchronization:** Changes that took place in geographical sites needed to propagate all network sites to maintain the correctness, availability, and reliability of data. But, due to variation in the hardware configuration clock pulse minor calibration differences are found, as a result, the system might refuse to sync with others. It is impossible to take care of one global notion of the same clock time. This results in the inconsistency of data at each node. On the other side, network delay and network gateway/firewall are equally responsible to affect the data synchronization process.
- g. **Fault Tolerance and overloading:** To establish reliability in a large geographical dispersed system fault tolerance methods play an important role. It is maintained by applying a dynamic replica propagation strategy, resilience in hardware resources so that corrective operations can be performed, and checkpointing on redundant data to recover the system during any fault, failure, or error by restoring the system to the previous checkpoint status. Overloading shows an imbalance in the functioning and degrades the performance of the system. There is a need for a good fault-tolerant system, which carefully examines the reason for failures and responds to the same.
- h. **Distributed tracing:** We are rapidly going towards an environment that is increasingly dynamic and dispersed. This means that distributed tracing is extremely important. And distributed monitoring is something that makes managing dynamic structures convenient for you.

1.10. Major Weaknesses of Distributed System

In distributed environment forecasts about the lifetime of any system is unpredictable. Success depends upon wrapping up every component of the distributed system in the present and future scenarios. But despite imparting the best measures still, there is a need to work out some weaknesses. Some of them are discussed below:

- a. **Lack of vision:** Time-Based Replication scheme introduces many such mistakes that further result in inefficiency, insecurity, and cost to maintain data in a distributed environment. But modern mathematics approaches such as the Paxos algorithm resolve such hurdles up to some extent [10].
- b. **Lack of Effective Software:** As a distributed environment is an existence of different machines working in a sophisticated environment where systems are communicating or interacting simultaneously for their task completion. It is difficult to deliver refined programs that enable us to react in different situations like route-finding during congestion, the network going down, and security issues in a distributed environment. As data access is for all users in the system so safeguarding of data from collusive attacks prevails there.
- c. **Lack of Self-Management Self-Control System:** Such a system is responsible to configure, heal, and optimize machine and network services to make them functional in disrupting situations. Configuring is all about the removal or adding of any existing resource or new components. Healing is for automatically finding, identifying, restricting, and regaining due to the event of any fault, failure, or error. Optimization refers to the tuning of existing processes and proactively reacting in different conditions.
- d. **Lack of Legal Jurisdiction:** Users' freedom to use different resources intended for and performs transactional activities to maintain their data.

Business services accessed from outside required some government bodies who intervene in legal disputes related to data security, privacy, its ownership, rights on intellectual property, and auditing in case of any complicity.

- e. Global status of Knowledge: Every node is having its memory for maintaining personal as well as other data or information. In such a case, it is not easy to keep track of the worldwide status of a completely distributed environment. Every process execution involves in the distributed environment proves a coherent vision of the system but in reality, it shows a limited observation of the system.

1.11. Challenges and Motivations

The challenges in the data distribution proved to be a hurdle that detracts from the performance in a distributed environment and some of them are described as follows:

- Earlier methods rely on user-defined parameters, which leads to bad distributed designs and facing drawbacks and lead to NP-hard problems.
- Dependency or usage of datasets having poor data quality for experimental purposes. Such datasets are having issues like missing values, noisy data, and incomplete data.
- Earlier techniques were unable to handle complex data in several scientific fields. Some hybrid similarity measure calculations are required in order to deal with such data.
- For an efficient load-based distributed architecture, earlier allocation strategies eliminate cost parameters or disregarded their utility in the allocation design.

- Past research seems not capable to maintain a deduplication ratio and data quality issues occur. It is found that there is less use of modern techniques like machine learning.

These prevailing challenges motivate us to propose a hybrid similarity-based threshold segmentation-based approach for fragmentation, a cost-based data allocation scheme to work for a load-based architecture, and propose a data deduplication technique to eliminate redundancy in data.

To deal with datasets, proper measures are taken into care. The data pre-processing technique is applied to the data to reduce the data dimension before data is used for experiments.

1.12. Organization of Thesis

The thesis is organized into chapters. A brief outline of the chapters is given below.

Chapter 1 discusses fragmentation, allocation, deduplication of data, and their roles, highlighting design issues, challenges, weaknesses, and about challenges and motivations.

Chapter 2 discusses prior work in the design of distributed systems for data fragmentation, allocation, and deduplication. It also covers research gaps, research questions, objectives, and methodologies with their scope.

Chapter 3 proposes a novel fragmentation scheme for data in a distributed network environment to deal with issues of storage management.

Chapter 4 proposes an optimized cost-based data allocation model for heterogeneous distributed computing systems.

Chapter 5 proposes a new scheme for handling duplication of data to achieve data quality and performance in distributed systems.

Chapter 6 concludes the proposed work, contributions and applications, and future work directions.

Chapter 2

Review of Literature

Globalization has transformed companies and given a push to keep an enormous amount of information received from different resources. The rate of increase of these resources is so rapid that companies are finding it difficult to focus on relevant information content. Some of the technologies that contribute to the design of distributed systems are data fragmentation, allocation, and deduplication. The architectural design of distributed systems influences data management and retrieval. It demonstrates how data is structured and makes it simple for others to obtain and use it. A successful design must consider customer satisfaction as well as how data should be disseminated across all sites to ensure data availability, consistency, and dependability. In light of this, a thorough examination of the focus regions and a portion of the data distribution is necessary to highlight in order to take action to close existing system vulnerabilities and provide a reliable system to the dispersed environment. This chapter provides a thorough analysis of several aspects of data distribution, including fragmentation, allocation, and deduplication, which have become more advanced over time in order to manage data on a distributed platform.

2.1. Introduction

Organizations are satisfied when a system responds to their various queries, provides distributed analytical services, and assists in the building of appropriate application interfaces for a variety of scenarios. Fragmentation, allocation, and deduplication processes are the first steps in developing a system that can manage a diverse application environment. Data is categorized/fragmented according to how it is accessed. Allocation is the process of dividing fragments into geographical locations. deduplication assures data quality and single data instance at each site by identifying and removal of it across several sites.

To enhance system architecture in a distributed context, several equally important variables must be addressed. Resource sharing, compute speedup, throughput, performance, dependability, and communication are only a few examples. Despite the diversity, several difficulties must be addressed, and they appear to be a roadblock in developing a distributed system. To make the distributed design more efficient and effective, the fundamental elements that generate a geographically scattered arrangement must be focused on and highlighted.

2.2. Existing Work Done in Distributed Design

Several strategies for addressing existing issues in distributed design have been explored throughout time. Proposed ideas were applied to the existing design to improve and fix the present challenges. These solutions sought to boost scalability, data processing flexibility, and communication effectiveness, as well as manage efficiency, resilience, and resource sharing.

K. Saxena et al. [11] presented a self-organizing capability architecture that may automate some grid operations and change the grid's structure or configuration. A Multi-Agent System (MAS) architecture was suggested for achieving self-organizing behavior. These agents are self-contained, loosely linked, and communicate with one another to solve any problems that the issue solver can't solve. Issues including increasing communication saturation, component up-gradation, and mapping, and slower decision rates cannot be avoided with centralized distribution systems.

Zouari M et al. [12] introduced a distributed adaption systems architecture that offered a way to boost efficiency, robustness, and scalability in a variety of situations. The adaptation engine in this system monitors the execution and initiates the active adaptation system if any deviations are detected. Adaptive distributed applications have gotten a lot of attention, but there haven't been a lot of studies done on them.

Yuan P et al. [13] presented a distributed access control architecture to handle all cloud computing security concerns. In multitenant and virtualized systems, this idea is utilized to regulate the distribution of physical resources. Untrustworthy renters can lead to unlawful information flow and side-channel assaults, according to this study.

A. Almutairi et al. [14] introduced symmetric distributed server architecture that uses workstation servers rather than huge mainframes. This method is used to handle thousands of devices/nodes in a big geographically dispersed environment. The message bus's purpose is to reduce the cost of communication and integration between devices. Using the network as a whole has already proven unsuccessful. It's ineffectual for communication in large, geographically dispersed systems, and it's expensive due to the giant mainframe's requirements.

Slimani Y. et al. [15] introduced a ReDy architecture that enables users to work efficiently in large-scale, flexible architecture. This technique provides scalability, fault tolerance, and dynamicity in a distributed system to achieve system performance. A distributed system's complexity is growing by the day due to its reliance on application integration. All processing, communication, and control technologies must be tightly integrated to ensure expandability, efficacy, trustworthiness, adaptability, and effectiveness.

Trung A. D [16] addresses the limitations of the current CAP theorem and suggested Lambda Architecture. The CAP theorem asserts that data in a distributed system can only be in two states: consistency with availability or consistency with partition tolerance, or availability with consistency, availability with fault tolerance, or fault tolerance with consistency, not all three states at once. To put it another way, existing CAP theorems only work for two of the three states.

Joe N. [17] introduced a microservices architecture that was designed to deconstruct any large system into architectural modules. Each module

communicates with one another via the Application Programming Interface (API). This architecture allows for more scalability. This architecture, unlike monolithic design, has no impact on the operation of large applications if a single module fails. Program codes are now smaller and easier to deploy. The only reason for this was to get beyond the limits of traditional monolithic structural design.

Weiss C et al. [18] highlight a variety of architectural techniques to guarantee performance, reliability, availability, efficiency, resilience, and security in a highly distributed environment. However, distributed systems become complex as networks extend and data behavior changes. Building a distributed network of self-balancing intelligent nature is essential to eliminating distributed environment challenges.

Y. Jiao et al. [19] introduced a load balancing method known as Weighted Least Connections (WLC) to increase the performance of a distributed system-based web server.

K. Iwanicki [20] highlighted the use of industrial IoT approaches with distributed systems to achieve high performance in existing designs.

H. Jiang et al. [21] highlighted the relevance of design patterns to developers in tackling reoccurring difficulties in distributed systems.

2.3. Existing Work Done in Fragmentation

Several contributions were given by the researchers to build the distributed system robust. They put all their sincere efforts to work on strategies help to split the data. It was noticed that the researchers did not stress the efficacy of the proposed data partitioning work and that it was important to strengthen it. For best results, data structures have to be organized into fragments suitable for further allocation at distributed sites. Some of the researcher's deals with the fragmentation of data also have tried to cover other aspects matched to data

allocation, security, communication cost, time, data access issues, etc. Some of their contributions are listed below.

Peng et al. [22] Work on distributed Resource Description Framework (RDF) is performed to manage the growing massive RDF. To utilize this large volume RDF is partitioned into small parts called fragments and further approaches the same for allocation in the distributed database environment. Here, the focus is given to reducing the communication cost during the query processing tasks. It also ensures maintaining data integrity and approximation ratio due to frequent access patterns from outside. Here RDF graph is divided using three fragmentation strategies namely horizontal, vertical and mixed fragmentation based on frequent access patterns. It is also focused on balancing and allocation of fragments into different sites.

H. Abdalla et al. [23] introduced a heuristic approach for fragmentation is proposed to reduce the transmission cost (TC) of queries in a distributed environment. Here, at the initial stage fragmentation is based on a cost-effective model in the context of the relational model, and at a later stage based on DDBS design. Different replication-based allocation scenarios were proposed i.e. mixed replication-based data allocation scenario (MAS), full replication-based data allocation scenario (FAS), and non-replication data allocation scenario (NAS).

Verma et al. [24] suggested an algorithm that uses a standard deviation that decreases the overall time for formulating the cluster by a simple k-mean. The proposed solution splits the gap with the standard deviation of the square root. This modified k-mean does even better with thank-mean and k-methods. Less time is needed to formulate the clusters. But neither k-means nor k-methods also work on very large-scale outcomes. Here, the authors have not used any kind of similarity measurement technique for distance calculation between different types of data which results in poor clustering performance, and it must be integrated with k-means using optimization approaches.

Z. Tao *et al.* [25] in their work suggest a deep neural network such as CODEnnn (Code Description Embedding Neural Network). CODE does not fit the textual resemblance but includes code fragments and high-dimensional vector field examples in natural language, as well as associated vectors in the code fragment and the accompanying definition. Code fragments associated with natural language questions can be obtained by the associated vector representation by their vectors. In the queries that must be handled, the task could even be semantically identified with the keywords. The researchers did not include source code management structures in this study to help symbolize, and the deep neural network is used and limited for the basic benefit of information engineering issues.

Sewisy A. *et al.* [26] introduced a revolutionary, unrivalled-in-complexity heuristic technique capable of combining multiple design-related techniques into a single work. Given the fragmentation and allocation cost models, this strategy should also be able to significantly lower the communication cost of distribution. The proposed fragmentation approach is intended to be used in relational databases.

X. Gu *et al.* [27] proposed, a new clustering algorithm for image co-segmentation tasks called the salience-guided limited cosine similarity clustering method (SGC3). In this, a one-step clustering technique extracts the usual foreground. An unsupervised method is used to direct the clustering mechanism's auxiliary partition-level information. To ensure the robustness of the noise and outliers in a given previous one the similarity between the instance level and the partition level is used for joint estimation. Eventually, the optimization of associated K-means aims to successfully solve the objective function. Experimental outcomes from two widely used data sets show that the proposed solution has achieved successful performance against the most mature distribution methods.

W. L Xiang *et al.* [28] introduced a systematic experimental analysis of twenty-four benchmark functions in a test suite. ABC (Artificial colony of bees) is a

very common and effective tool for optimization. ABC still does however have a lack of convergence. To further increase ABC convergence velocity, a new form of ABC (CosABC) is proposed to pick better neighbors based on cosine similarity. Under the direction of chosen neighbors, a new solution search equation was applied to reduce the constraint of ABC undirected search. There is a further contrast with some of the most sophisticated algorithms to check Cos-ABC supremacy. The related results of the comparison show that Cos-ABC is efficient and competitive.

Wiese et al. [29] proposed an algorithm to find similar knowledge for a user whose original query cannot be addressed precisely, clustering-based fragmentation is suggested. Approximation algorithms and lookup tables are used to give a better shape to the distributed system for supporting flexible query answering.

Ali A. Amer et al. [30] proposed an optimized fragmentation approach on each attribute to know about their retrieval and update frequencies in each site. And proposed a synchronized horizontal fragmentation approach to reduce data locality issues and total cost. In this work, if query (Q) is initiated from multiple (M) locations, this query will be interpreted as a separate query for each position with a different radio frequency.

S. I. Khan [31] proposed a new technique called Attribute Level Precedence (ALP) to partition global schema/database relations at the initial and later stage in case of non-availability of data access statistics and query execution frequencies. ALP technique is capable to take an advanced decision for fragmentation at the initial stage (knowledge gathered during the requirement analysis phase) without empirical data statistics. ALP is a table responsible to fragment a relation horizontally based on the importance of an attribute in a network site.

To increase the performance of distributed database systems, *Masood Niazi et al.* [32] suggested enhanced vertical fragmentation, allocation, and replication

strategies. To reduce query processing costs, the suggested fragmentation approach groups highly-bonded properties (i.e., those that are normally retrieved together) into a single fragment. The suggested allocation technique aims to discover an optimal allocation that reduces the round-trip response time. The replication strategy partially copies the fragments to maximize local query execution while minimizing the cost of replica transmission to the locations.

Rahimi et al. [33] proposed an algorithm to render fragments vertically with an Updated Bond Energy Algorithm (BEA). This algorithm utilizes attribute affinity and seeks to create clusters of attributes and attributes that are individually evaluated by the same query.

Lim et al. [34] proposed a hybrid fragmentation approach for deductive database systems as HFA for horizontal fragmentation and, RCA and DVF for vertical fragmentation. This is a two-phase process in which a deductive database is fragmented using variable bindings and dependency relationships represented by the dependency graph.

Khan S. I. [35] suggested MCRUD and MMF algorithms for the efficient partitioning of large datasets without query statistics. It is suggested here that earlier partitioning methods were not acceptable because there were no usable statistics at the initial stage of the implementation of distributed database query statistics. In his paper, an optimal fragmentation technique is proposed to partition global relations of a distributed database when there are no data access statistics and no query execution frequencies are available. When data access statistics and query execution frequencies are not available at the initial stage then MMF is responsible for partition relation in the distributed database. MCRUD is responsible for taking fragmentation decisions without using empirical data.

I. B. Oriji et al. [36] proposed a hybrid optimized model using the information on the type and frequency of queries for fragmentation of data horizontally and

vertically and is based on a supervised machine learning approach to produce non-overlapping fragments. These fragments are maintained by archiving process rather than deletion operation on them. These fragments are used to facilitate searching operations based on the index so that database tables are partitioned horizontally and vertically.

Work on frequent access patterns (FAP) is given to reflect the behavior workload to ensure the data integrity and ratio of approximation. *Peng et al.* [37] presented a data structure that was based on trees by utilizing the depth-first search (DFS) coding for maintaining them as well as managing newly entered queries [24].

Aloini et al. [38] presented a fragmentation mechanism that has recently been shown to hurt the performance of negatively exported processes. Finally, by merging Process Mining (PM), Social Network Analysis (SNA), and Text Mining the fragmentation process improved knowledge sharing among port Community System (PCS) actors so that process efficiency can be achieved.

Memmi et al. [39] highlighted a study on data fragmentation in the public and private sectors to hold information in a structural archive to attain data security.

Raouf et al. [40] proposed a distributed database vertical fragmentation, allocation, and replication strategy named (VFAR). The suggested technique vertically separates distributed database relations in the initial stage of distributed database architecture, eliminating the requirement for frequent user queries, which are not available at this time. It also allocates and copies the resultant fragments to the distributed database's sites, utilizing each site to manipulate and read actions on each fragment. First, the suggested scheme decreases the overheads and complexity of existing vertical partitioning approaches' expensive computations, according to experimental data. Second, it addresses the issue of a high volume of user questions that aren't available at the outset. It also addresses the issue of iterative development. Third, it

generates all fragments of one iteration to solve the problem of repeated binary partitioning in the case of n-ary partitioning.

Ahmed E. A. Raouf et al. [41] introduced a cloud-based complete vertical fragmentation, allocation, and replication (FVFAR) system that addresses the limitations of earlier vertical fragmentation solutions while simultaneously providing vertical fragmentation, allocation, and replication as a service. The FVFAR scheme begins with the requirement analysis phase of the system development life cycle and partitions the distributed database relations vertically at the first stage of developing the distributed database, eliminating the necessity for frequent user queries that aren't available at this time. It also distributes and duplicates the resulting fragments to the distributed database system's sites, improving system performance, increasing availability, and lowering the communication cost of database relations access via the cloud. The FVFAR system addresses the limitations of earlier vertical fragmentation solutions, according to experimental results. Furthermore, compared to earlier methodologies, it results in a significant reduction in total communication costs while executing distributed database system queries. It also demonstrated the ability of the FVFAR approach to perform vertical partitioning of distributed database relations during the first stages of database architecture. As a result, before partitioning a database, database designers do not need to wait for empirical data on query frequencies.

Abdel Raouf A.E. et al. [42] demonstrated a cloud-based enhanced dynamic distributed database system. Fragmentation, allocation and replication decisions can all be made dynamically at runtime with the proposed approach. Users can also access the distributed database from any location. Furthermore, this study proposes an improved allocation and replication technique that may be used early in the distributed database design process when no information about query execution is known.

Haughlid et al. [43] proposed DYFRAM, a decentralized solution for dynamic table fragmentation and allocation in distributed database systems

based on on-site access patterns. The method performs fragments, replicates, and reallocates data based on recent access history, with the goal of increasing the number of local accesses over remote accesses.

2.4. Existing Work Done in Allocation

The way the data is spread across several geographical regions determines the success of a distributed system. Easy availability of data at a low cost depends on the strategies applied for the placement of data fragments remotely. Some of the efforts adopted by researchers are discussed as below:

Tosun U. et al. [44] employ the well-known Quadratic Assignment Problem solution algorithms to handle the fragment allocation problem in distributed databases. This significant problem is solved using a new collection of Genetic, Simulated Annealing, and Fast Ant Colony techniques. The execution times and quality of the fragment allocation possibilities are explored in the tests. Even for a vast number of fragments and sites, the results are quite promising. Only one fragment is assigned to each site by the model used to determine where each fragment will be placed.

H. I. Abdalla [45] proposed a new dynamic deallocation approach for a given fragment as Update Matrix (UM) and Distance Cost Matrix (DM). It works based on changing data access patterns in replicated and non-replicated distributed database systems. It was assumed that fragments are allocated on the network site are based on the applied frequency value of the database data items. Reallocation of data fragments on the remote sites is planned based on communication and update cost value. Each fragment is having an updated cost value. Fragment having maximum update cost value is considered for reallocation and chosen candidate site to store fragments to minimize the communication cost. UM is defined as the value getting after issuing of update query at a particular site for the manipulated fragment. In this approach when the same query is applied at more than one site, then queries can be treated differently from each other and have different frequency values.

Singh, A. et al. [46] proposed a biogeography-based optimization algorithm (BBO) for non-replicated static data allocation in the distributed database system. It reduces the data transfer cost while executing a set of queries.

N. K. Z. Lwin et al. [47] introduced a non-redundant dynamic fragment allocation technique that is based on the changing access pattern at different sites to improve the performance. Here fragments reallocation is dependent on access made on each fragment data volume based on a defined time constraint and the threshold value. This proposed technique changes the reallocation strategy by modifying the read and write data volume factor and introducing threshold time volume and Distance Constraints Algorithm. Write data volume is considered for the reallocation process when more than one sites approach the fragments. This ensures the overall improvement of distributed system performance.

H. I. Abdalla [48] proposed an algorithm called Simulates Annealing with Genetic Algorithm (SAGA) is used for optimal allocation of fragments in a distributed environment. Here, allocation of data depends on access patterns for fragments and focused on reducing the allocation cost during the movement of data fragments from one site to another.

Guo et al. [49] were introduced an energy-efficient dynamic loading and resource scheduling method that includes reducing energy usage and decreasing application times. The method also successfully decreases energy efficiency by modifying the CPU clock frequency of SMDs to the optimum in local computing and adjusting the communication energy of wireless channels in cloud computing.

Liu et al. [50] were worked for the placement of virtual machines in cloud computing and energy-efficient EMACS was created. The intended virtual machine placement was achieved with the fewest number of active machines and by turning off idle nodes. According to experimental investigations,

OEMACS aimed to minimize the number of active servers, increase resource use, balance diverse resources, and reduce power consumption.

Malekloo et al. [51] was presented a multi-target colony optimization technique that offers energy and service-sensitive performance in the placement and consolidation of virtual machines. The results demonstrate that this technique outperforms the other ways in terms of energy consumption, limiting CPU waste, lowering energy communications costs caused by traffic sharing across virtual machines, and reducing the number of virtual migrations to system and SLA violations.

Vakilinia et al. [52] were proposed his work to "minimize all of cloud data center power consumption" a platform for virtual machine placement.

Sharma et al. [53] discussed an evaluation of current reliability and energy management strategies and their effect on cloud computing. There were debates on the classifications of resources loss, failure tolerance mechanisms, and mechanisms for energy conservation in cloud systems. Different problems and study gaps have been established in the balance between energy reliability and quality.

Long et al. [54] introduced a strong immune clonal optimization method based on the dynamic load balance approach and immune clonal selection theory in green cloud computing has solved the problem of high energy consumption and reduced cloud utilization. In terms of solution efficiency and processing costs, the experimental findings show that the method outperforms clonal selection techniques.

Kaur et al. [55] demonstrated the need for energy management when addressing the dual position of cloud computing as a significant contributor to rising energy use and reducing energy waste. The research provided an in-depth analysis of current energy management methods in cloud computing. It also supplies taxonomies for the assessment of current work in the area of science.

Lee et al. [56] addressed the consolidation of tasks as an efficient way to maximize resource usage and minimize energy use. The research focused on two energy-conscious energy consolidation heuristics intended to optimize the use of resources and take both active and idle energy use directly into account. The heuristics suggested assigning each job to the resource, which minimizes the energy needs explicitly or indirectly, without degradation in performance.

Beloglazov et al. [57] suggested an energy-efficient cloud computing architectural structure and concepts. The study identified open analysis problems, the provision of infrastructure as well as algorithms to handle cloud computing environments effectively. The conclusions indicate that the suggested model of cloud storage makes substantial cost savings and has a high capacity in complex workload environments for energy efficiency improvements.

Scionti et al. [58] introduced an algorithm for allocating the total energy consumption. The effects of simulations were also viewed on a state-of-the-art platform. The suggested solution results in tangible energy conservation, showing energy efficiency dominance in comparison with well-known and widely accepted allocation methods.

Khan et al. [59] conducted their research on maximizing physical and virtual machines' capacity and energy consumption in a cloud computing system. Findings offered a good understanding of how power and energy usage were affected by various workloads. The tools and structure presented can be used for research and improving energy efficiency in any cloud environment and of any scale.

Tang et al. [60] addressed a problem of energy optimization that has been modeled whereas the task dependence, transfer of data, and some constraints such as response time, and cost have been considered and solved by genetic algorithms. A series of simulation trials have been carried out to assess the

algorithm efficiency and the findings suggest that the proposal is more effective than the benchmark method.

Liu et al. [61] proposed an optimal paradigm for work schedules to decrease energy usage in cloud data centers. The proposed solution was designed as an integer programming problem to reduce the energy consumption of a cloud-based data center by organizing activities for a small number of servers and adhering to task response time constraints. As a realistic program, the authors have developed the most effective initial task-programming method for the server to decrease energy expenses. A data center planning system with diverse tasks is modeled and simulated. The study findings reveal that the recommended work scheduling strategy reduces server power consumption by more than 70 times on average when compared to a random job scheduling system.

Zhao et al. [62] suggested a power-aware scheduling approach for a heterogeneous cloud network to solve the issue of high energy consumption. The results show that the average power consumption in this system is 23.9 - 6.6% lower than in modern technology.

Li R. et al. [63] proposed an abstract model that uses piecewise linear functions to handle data analytics workload in a distributed cluster architecture. This is responsible to reduce the makespan time to handle cost issues.

Chung-Chi H. et al. [64] proposed a hybrid heuristic genetic algorithm and the steepest descent methods to achieve optimal task allocation with a reduction in hardware policies to reduce system cost.

Kashish Ara Shakil et al. [65] developed a latency-aware max-min algorithm (LAM) for resource allocation in cloud infrastructures. The suggested method was developed to handle resource allocation challenges such as changing user requirements and on-demand access to infinite resources. It may allocate resources in a cloud-based environment to enhance infrastructure performance and increase revenue.

2.5. Existing Work Done in Deduplication

P. G. et al. [66] proposed a new way for constructing safe deduplication systems in the cloud and fog using Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms. The proposed method concentrates on the two main objectives of such systems. On the one hand, data redundancy must be reduced to a bare minimum, while on the other, a solid encryption strategy must be created to assure data security. The proposed method is well suited for tasks such as a user uploading new files to the fog or cloud storage. The file is first encrypted using the Convergent Encryption (CE) method, and then re-encrypted using the MECC algorithm.

Ahmed Sardar M. Saeed et al. [67] introduced a technique to detect substantial redundancy and content-defined chunking (CDC) has been employed in data deduplication. They optimized the deduplication system by tweaking relevant parameters in CDC to determine chunk cut-points and provided an efficient fingerprint using a novel hash function in the proposed study. They developed a new low-cost hashing function and a novel byte's frequency-based chunking (BFBC) approach.

A Vijayakumar et al. [68] introduced a service named Key Generation and Token Maintenance (KGTMAAS) to manage cloud storage and minimize duplicated data. It can detect duplicate data at both the block and file levels.

Dinesh Mishra et al. [69] proposed a framework for role and attribute-based de-duplication in the cloud that supports diverse content types. De-duplication has been done on a role-by-role basis. That is, distinct material should be kept for each role type, or de-duplication should be performed. It will handle a variety of content types, including text, photos, and video. Users with the same function may have several versions of the same content that they share or trade. These can be text, photos, or any other sort of data, hence de-duplication is necessary at the role level rather than at the system level to decrease memory and storage requirements.

Duaa S. Naji et al. [70] developed a novel de-duplication methodology to work on the contents of large data sets. Different dictionary indexing approaches will be utilized to de-duplicate the field's contents that have bounded variability. In addition, for fields with lengthy strings, a set of computationally low-cost hash algorithms will be utilized to speed up deduplication.

S. Ruba et al. [71] in their work detected that string-to-string comparison results are a less effective identification architecture to detect data duplicity. For the removal of redundant data stored on the disk fingerprint index detection and prefetching techniques were introduced. It was based on reinforcement learning and helped to remove 2% to 10% duplicity in comparison to the sparse indexing method.

Guangpin X et al. [72] proposed data deduplication work to reduce the duplicate data in scale-out distributed storage systems. To achieve this more scalable and efficient deduplication process mechanism as SEP-D for locating metadata information using content-based hashing to reduce I/O operations and CRUSH for the application of placement strategies.

H. Fingler et al. [73] introduced the sub-file-level chunking deduplication system as an example of system deduplication since it divides the incoming data stream into several data "chunks" and uses comparing methods to find duplicates. De-duplication systems delete duplicate chunks, allowing only one copy of these chunks to be stored or sent to achieve the storage space or network bandwidth savings goal. It's worth noting that de-duplication systems have a long runtime and require a lot of CPU resources to function.

H. A. S. Jasim et al. [74] introduced the genetic evolution-based data deduplication methodology has a greater deduplication ratio than existing systems, allowing it to detect duplicated data even quicker. The divisor D values in prior methodologies are static; however, in global optimization using a genetic algorithm, we dynamically choose the optimum divisor D value to uncover maximal redundancy. Bucket indexing based on genetic evolution

deduplication is 16 times quicker than Rabin CDC, 5 times quicker than AE, and 1.6 times quicker than Fast CDC.

In Bigdata storage, a large amount of data is stored in digital form without using a structured approach. So, to identify duplicate data and create unstructured data into structured data a new technique was presented by *N. Kumar et al.* [75] to increase the efficiency of Bigdata storage. Based on hash values and MD5 methods a bucket-based deduplication system with a fixed-size chunking level is proposed. To execute this strategy, the researchers employed the Destor open-source application and HDFS (Hadoop Distributed File System).

N. Kumar et al. [76] proposed the AR-Dedup cluster deduplication system to achieve high data deduplication rates and minimal overhead communication while maintaining load balancing. In their program, they adopt an application-aware method for deduplication. AR-Dedup makes use of cluster deduplication routing.

Y. Xuan Xing et al. [77] presented an artificial neural network deduplication methodology to discover duplicate data. The suggested system takes as input a set of data created by various similarity measurements. The ANN is used to process the model parameters that have been computed from similarity functions i.e. dice coefficient, Damerau-Lavenshtein distance, and Tversky index. The ANN has two stages, one for training and the other for testing data.

Y. Zhang et al. [78] introduced a method to achieve maximum disc storage savings, it spreads data blocks over numerous storage nodes and performs rapid compression following data deduplication. Droplet improves disc IO by combining write requests for tiny data blocks into big chunks. Droplet has demonstrated great deduplication performance in tests. Droplet is a high-throughput and scalability distributed deduplication storage solution.

S. S. Sengar et al. [79] proposed an in-line data deduplication distributed architecture that follows an intelligent storage balancing strategy to use storage

nodes for storing data to enhance deduplication efficiency. This architecture was capable to perform deduplication with high throughput and ratio and was found effective in comparison to performing duplication on a single system. A technique was proposed called sampled hashing to strengthen the scalability of the distributed architecture.

Paulo, J. et al. [80] presented DEDIS, a dependable and distributed system that conducts offline deduplication across VMs' primary storage volumes in a cluster-wide approach. DEDIS is designed to be completely decentralized, avoiding any single point of failure or contention and allowing for safe scaling out. As long as it exports a shared block device interface, the concept is compatible with any storage backend, distributed or centralized. It also included new optimizations for enhancing deduplication efficiency and reliability while lowering the storage request effect.

Singhal, S. et al. [81] proposed an approach that focuses mostly on chunking and indexing in distributed data deduplication systems. The suggested method uses GA to determine the value of the chunk's dynamic single divisor D . GA's primary goal is to get a high deduplication ratio, although this requires some additional time. To find the value of the divisor, a novel model is proposed by associating GA. This is the first time GA has been used to deduplicate data. The BST index tree is used to index the fingerprints, which is the proposed system's second innovation. On VMDK, Linux, and Quanto datasets, the suggested system's performance is examined, and the deduplication ratio is improved significantly.

Shengmei Luo et al. [82] proposed a distributed deduplication cloud storage solution. Using several storage nodes to deduplicate in parallel, it enables scalable throughput and capacity with low deduplication ratio loss. First, Boafft uses a data-similarity-based efficient data routing algorithm that not only reduces network bandwidth overhead but also quickly calculates data storage location. Second, each data server keeps a similarity index table in memory that can be used to partially deduplicate data and avoid a significant number of

random disc reads/writes. Finally, using a cache container of hot fingerprints based on access frequency, we improve the data deduplication ratio in a single node.

2.6 Problems Statement

The researchers devised many solutions to data management issues. When data is sufficiently formatted, prepared, or regularly arranged, it functions well as a crucial component of information construction. However, changes in data behavior in terms of quantity, cost, and quality have an influence on worker capabilities and performance. When users send data access requests from any activity center, they confront a number of challenges. Data extraction delays, transmission or data loss, availability of partial data at the node level, high data access prices, data quality, and inconsistency are some of the problems.

In distributed data management, empirical data was employed. When applied to any NP-hard problem solution, such drawing solutions to such problems look radical or fundamental. It is tough to operate efficiently on many networks while accessing their data. Such outcomes are not useful and do not work for progressive results in study domains during studies. Data is being created at an increasing rate due to the importance of data in business. Every user seeks data for a variety of reasons; some are required to change existing data patterns, while others use data for business analysis. Various social networking groups and corporate web programs generated equal volumes of data, resulting in a large storage space full of duplicated data.

Very little work had been done in the domain of textual data context encouraging to work for a distributed architecture to handle it properly. The amount of text data generated by social media groups such as WhatsApp, and Facebook is constantly growing. This data has a storage problem, which increases data complexity when many data-access activities are performed at the same time. It necessitates the management of multiple concurrent users accessing their data from various locations. It demonstrated a system that

enables users to perform transactions on shared data simultaneously and enhance performance on the distributed ground.

2.7 Research Questions

To work upon such hurdles several questions are active in mind. Some of the research questions are depicted below:

- How to manage textual data in a large distributed network to achieve distributed performance?
- How can a distributed network accomplish an optimum data allocation that reduces overall costs?
- How do large amounts of data stored on dispersed machines become redundant-free, hence improving data quality?

2.8 Research Gaps

Based on the literature review outlined in the previous chapter, the following gaps in the existing literature have been identified:

- Usage of relative similarity is found in most of the studies in modern-day fragmentation techniques.
- The allocation model lacks the supplementary load architecture and also ignores the network cost.
- Data duplication identification methods can be improved by the application of modern-day machine learning which is not monitored.

2.9 Objectives of Research Work

- Design a new data fragmentation architecture based on a similarity-based threshold segmentation method.
- To introduce a novel data allocation scheme to optimize the allocation process based on Swarm Intelligence.

- To improve the existing data duplication scheme with Machine Learning architecture.
- To evaluate and analyze the proposed methodology with the existing scheme using a simulation framework.

2.10 Scope of the proposed work

Every company now uses data repository systems such as relational databases, transactional databases, and multimedia databases to retain data from various sources. Researchers make every effort to work with massive amounts of data to address issues such as delays, inconsistent data, and missing data, among others. To accommodate ever-increasing data, the emphasis is on improving data management through data fragmentation, allocation, and deduplication. Data, such as audio, video, and textual content, is growing in size as it is generated from all directions. The goal of this research is to handle textual data using appropriate distribution methods, allocation algorithms, and data quality improvement while maintaining a single data instance.

2.11 Proposed Methodology

This section explains the proposed methodology for the implementation of the work and as depicted in figure 1.5.

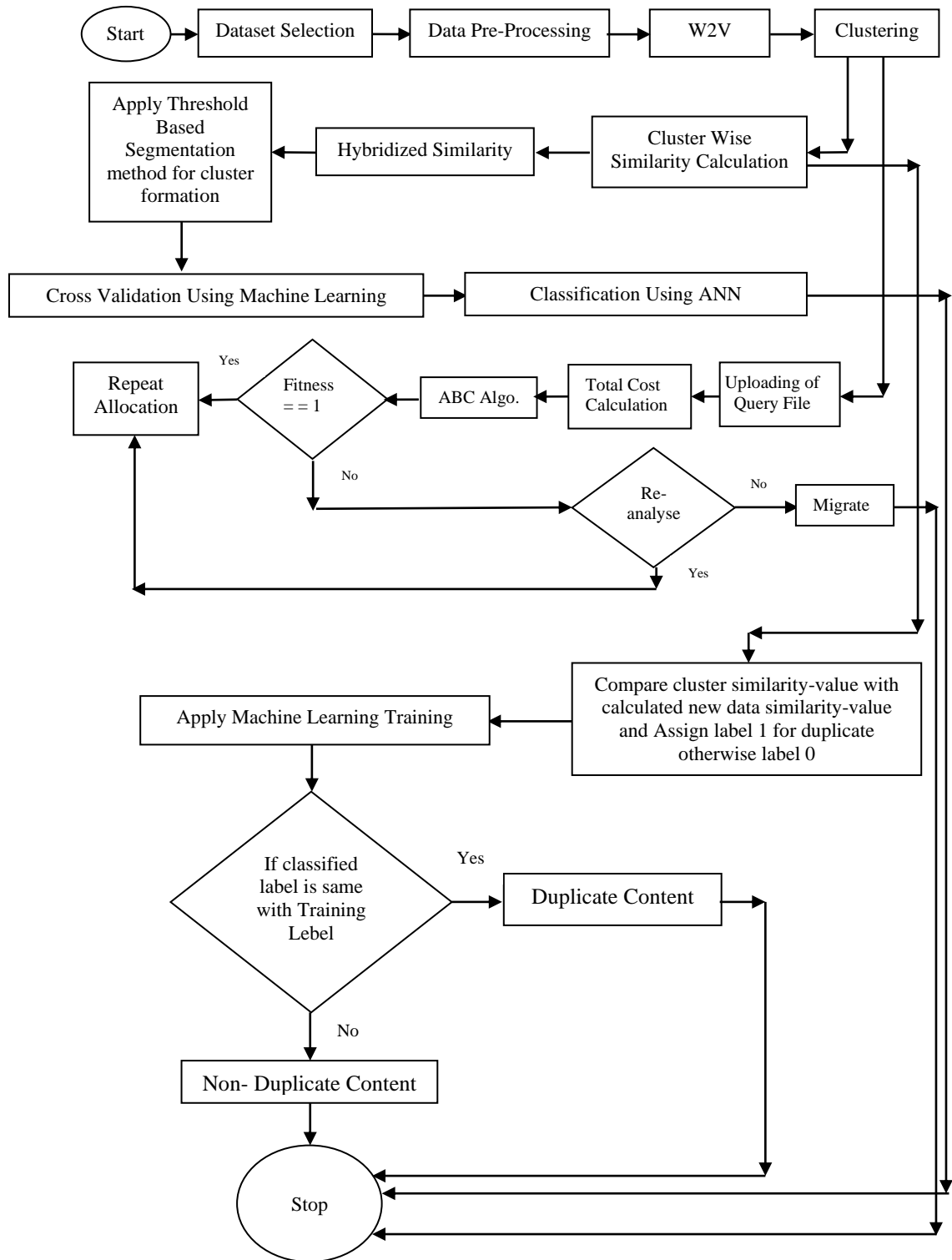


Figure 1.5 Research methodology flow

2.11.1 Pseudo-code for the Data Fragmentation

The following steps are accomplished to achieve this work and are as follows:

Step 1: Loading of data for fragmentation process

Step 2: Selection of data and its representation in a list, row-by-row

Step 3: Applying data preprocessing tasks i.e. stop word removal

Step 4: Filtered data generation and its conversion into Vector Model

Step 5: Cluster formation based on the vector data model

Step 6: Applying similarity index calculations clusters-wise (cosine, soft-cosine similarity)

Step 7: Hybridize the similarity

Step 8: Apply threshold-based data fragmentation based on the similarity index

Step 9: Cross-validation to authenticate fragments

Step 10: Validate the fragments by using machine learning and classifying it using an artificial neural network (ANN).

2.11.2 Pseudo-code for the Data Allocation

The steps are as follows:

Step 1: Loading of Query File

Step 2: Total cost calculation

Step 3: Level-wise estimation of execution patterns

Step 4: Execution of cost patterns task-wise

Step 5: Applying the proposed artificial bee colony (ABC) algorithm

Step 6: Re-Analyse Optimal Result using Upgraded Threshold

2.11.3 Pseudo-code for the Data Deduplication

The steps are as follows:

Step 1: Begin uploading saved data and queries.

Step 2: Remove stop words from the dataset using text pre-processing.

Step 3: Convert textual information into vectors.

Step 4: Using k-means to categorize existing or new data into clusters, find indices (indicating the number of probable clusters) and centroids positions (the center point of each cluster).

Step 5: Indexes for training and testing are created.

Step 6: The cluster-wise similarity index calculation.

Step 7: Check duplicate data in testing data using the ground truth (*gt*) test. Label 1 should be used for duplicate data entry; otherwise, label 0 should be used.

Step 8: Perform a similarity test on the test data.

Step 9: To find duplicate figures, the results of similarity tests are compared to the similarity of each cluster according to set threshold values.

Step 10: Calculation of performance parameters such as sensitivity, specificity, accuracy, and fmeasure to assess the efficacy of a suggested deduplication system.

Design a New Data Fragmentation Architecture Based on a Similarity-Based Threshold Segmentation Method

A distributed system is defined as the use of separate computers to share various resources over linked networks. A properly distributed architecture may fulfill each data item need specified by users. Users employ a variety of query methods to locate desired data or information. Some of them focus on data items from a single table or a combination of tables. User queries are categorized as fine or coarse-grained, single or multi-database, and use a collective and selective approach to retrieve needed data from several viewpoints [83]. If knowledge grows at an exponential pace on a daily basis, design architecture must be scalable in order to handle vast amounts of data in the future. In this work, an effort for fragmentation of data is given to handle data at large. To achieve a similarity-based threshold segmentation approach is applied to make clusters or groups represented as fragments.

3.1 Introduction

One of the most important aspects of any serving network is data distribution. The organization of data in a dispersed context has a significant impact on data storage and retrieval. The demands of consumers have changed in tandem with the rapid growth of technology. A user who was once immobile has evolved into a mobile user who requires data access from everywhere on the planet. Users may migrate from one service provider to another as a result of an unstructured data structure causing network output delays. When it comes to data storage, data fragmentation is a crucial component. Others will be able to use organized data more easily. The extraction of information on time is quite difficult due to a large amount of data. As a result, to accomplish performance in a distributed system, an optimal strategy is required to overcome earlier shortcomings and serve the greatest number of consumers possible across a large geographical network.

The fragmentation, allocation, and deduplication of data mechanisms are linked to data improvement in a dispersed network. By breaking huge global data into small independent portions termed fragments or segments [84-91] it is possible to make the best use of available storage space and efficiency. When opposed to retrieving data from a single huge data structure, these discrete segments help to reduce demand in a globally distributed context. Scalability, data availability, security, searching speed, and data inconsistency may all be easily managed with distributed systems. Working with a single massive data system and entertaining millions of users at the same time while maintaining high data accuracy has become difficult.

Parallel technology, software technology, and network infrastructure advances have all contributed to the emergence of cloud computing, VANETs, and OPPNETs [92],[93]. This is a novel computer model that offers users data, programs, and various IT resources via the network as a service, with no latency in information transmission [94]. Cloud computing is a type of infrastructure management tool that includes resource management via virtualization technologies and a big capacity resource pool. Users of the cloud will make requests via the network and subsequently receive the service. The resource pool can be dynamically deployed, modified, and reconfigured, and the operation can be canceled, among other things [95].

The main goal of this research is to develop a novel relative-based fragmentation architecture for scalable query addressing in the context of textual facts in a distributed network environment. Previously, data fragmentation was based on empirical data and was broad in scope to get the desired result. This method applies similarity calculations on textual data are used to arrive at a definitive result utilizing vector calculation. This study uses a combination of cosine, soft cosine, and hybrid similarity as a means of distinguishing between data items for partitioning.

Earlier strategies focused on discovering similarities between several documents, but the proposed methodology is responsible for determining the

similarity in the connection itself by comparing one row to the next and applying similarity calculations to vector values. As a result, it is necessary to rely on the required strategy that is appropriate for a diversified environment with an adaptive character.

3.1.1 Segmentation Technique

Segmenting data or splitting cloud data into smaller, coherent, and interconnected sections is a method of collecting data with similar features. Text segmentation is the process of breaking down a document into smaller chunks, or segments. In word processing, it is widely utilized. Depending on the goal of the text analysis, these segments are classed as words, phrases, subjects, sentences, or any other unit of data. Text segmentation is a technique for deleting logical sections of text. The passage or boundary portion is the name given to this part. There are a variety of reasons why a divided document might be useful for text analysis. This could be because it is smaller and more coherent than the entire document. When it comes to getting information, text segmentation is a major issue. Its goal is to partition a text into homogeneous segments, which are segments that have the following characteristics: **(a)** each segment has a distinct topic, and **(b)** adjacent segments cover diverse themes. From a large base of unformatted or loose material, these pieces can be tracked as suitable to a query [96].

3.1.2 Similarity Measures

When there is ground truth for clustering, the similarity value is derived using the ground truth of the cluster or region; however, when there is no ground truth for the region or cluster, the ground truth becomes radical, and similarity measures are generated using vector calculations. Discourse is the measurement of the equivalent of two pieces of evidence. An agreement is generally characterized as a gap along with the dimensions representing the objects' attributes in the context of data mining. If the distance is little, the degree of similarity will be high; if the distance is big, the degree of similarity

will be below. Plagiarism, asking for a similar question previously answered on Quora, collaborative filtering in recommendation systems, and so on all employ the similarity metric. The distance between different points of data is a similar measurement. The difference is between the two data items measuring divergence, whereas similarity is a variable that shows the strength of the association between two data items. In this work, three similarity metrics are combined: Cosine, Soft Cosine similarity, and hybrid similarity [97].

3.1.2.1 Cosine Similarity

In general, similarity is a measure of how similar things are when compared to other comparable things. One involved the use of vectors, which are computer equations. A vector is a quantity that is both large and has a direction. In computer science, a vector is a one-dimensional sequence. The angle of cosine between them is calculated using the similarity of the cosine approach. Finding the angle between the two vectors requires the point product of the two vectors, as shown in **Figure 3.1**. We will effectively attempt to get the cosine of the angle between the two lines by establishing the cosine relation. The 0° cosine is 1, and every other variable is less than 1. As a result, it's an orientation judgment rather than a magnitude judgment: two vectors with the same direction have a cosine similarity of 1, two vectors with a 90° similarity of 0, and two vectors with the same direction have a similarity of -1, regardless of magnitude [98].

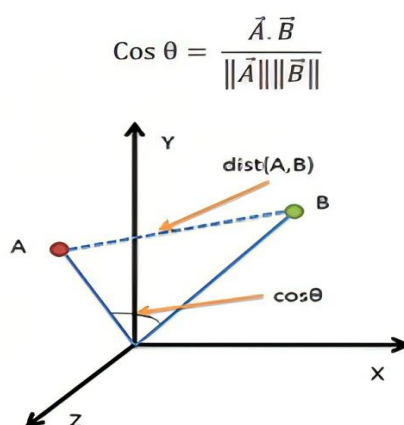


Figure 3.1 Cosine Similarity

3.1.2.2 Soft Cosine Similarity

A soft cosine agreement test evaluates the agreement's attributes. The traditional Cosine conformity criterion judged similarity based on features determined by the vector space model (VSM), which are fundamentally distinct from one another. Soft cosine similarity, on the other hand, can be a huge benefit if you need to apply a criterion of an agreement to help with document grouping or classification [99].

3.1.2.3 Hybrid Similarity

The qualities of cosine similarity and soft cosine similarity are merged in this similarity metric.

3.1.3 Machine Learning

Machine learning (ML) is an artificial intelligence (AI) technology that allows devices to learn and strengthen themselves without the need for explicit programming. Machine learning is concerned with the creation of computer systems that can observe and learn information by themselves. The learning process, according to the explanations we provide, begins with insights or data, such as examples, direct knowledge or input, data patterns, and making informed future judgments. The main goal is to allow computers to learn on their own and alter their behavior accordingly without the need for human interaction or assistance [100].

The following is how ML categorization is always done:

- A supervised machine learning algorithm
- Machine Learning Unsupervised Algorithm
- Deep learning algorithm that is semi-supervised
- A strengthening algorithm based on machine learning [101]

The relative information model is used entirely by the machine learning architecture (illustrated in **Figure 3.2**).

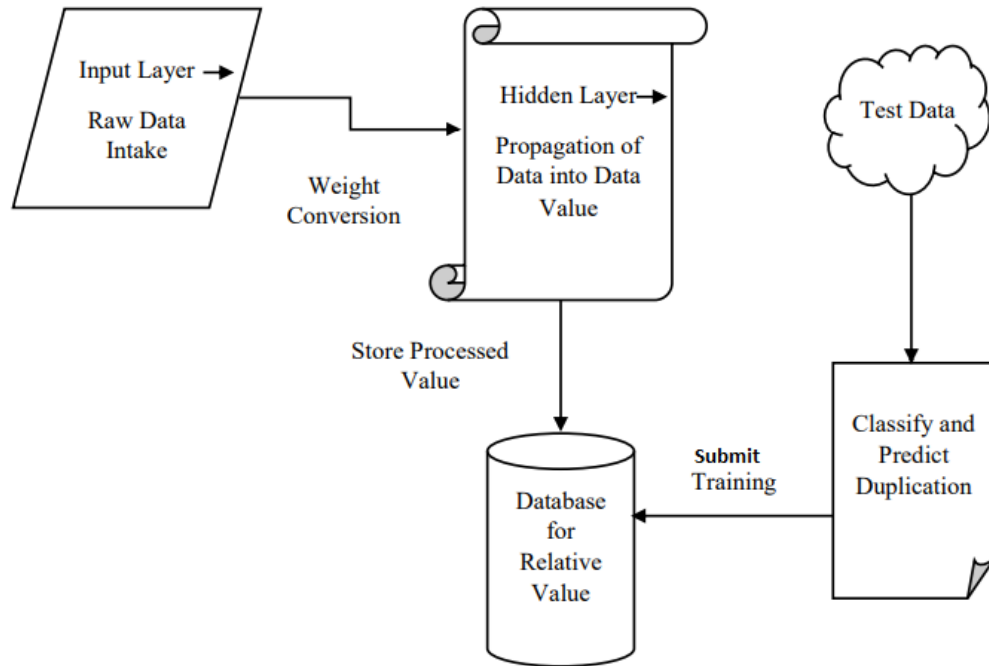


Figure 3.2 Machine Learning Architecture

It takes in what is given to it and learns from it. It is divided into three parts: the input layer, the hidden layer, and the output layer. The input layer takes the raw data and changes it into a more intelligible form by comparing it to the defined goal label. The data's meaning is propagated by the hidden layer, which generates the cross-validation training platform. AI and machine learning as a solution to problems involved in the process of validating the data received as output from the proposed method applied.

3.2 Methodology and Implementation

This section covers steps involving the processing of data fragmentation implementation discussed below and graphically depicts in Figure 3.3:

- Start by uploading Twitter data that hasn't been labeled.
- Apply Data Preprocessing
 - Remove English Stop Words from the data to filter it.
- Apply word-to-vector on filtered data.
- On the set of row lists, perform similarity calculations.

- Identifying the data's initial centroid.
- The Euclid distance between each tweet was calculated.
- Counting how many centroids there are.
- Using K-Means to generate fragments
- Use Machine Learning (ML) Training to validate the fragments and a neural network to classify them.

To do so, we went through a series of procedures. The following sections contain step-by-step descriptions of each process. In MATLAB R2016a, each step is carried out separately.

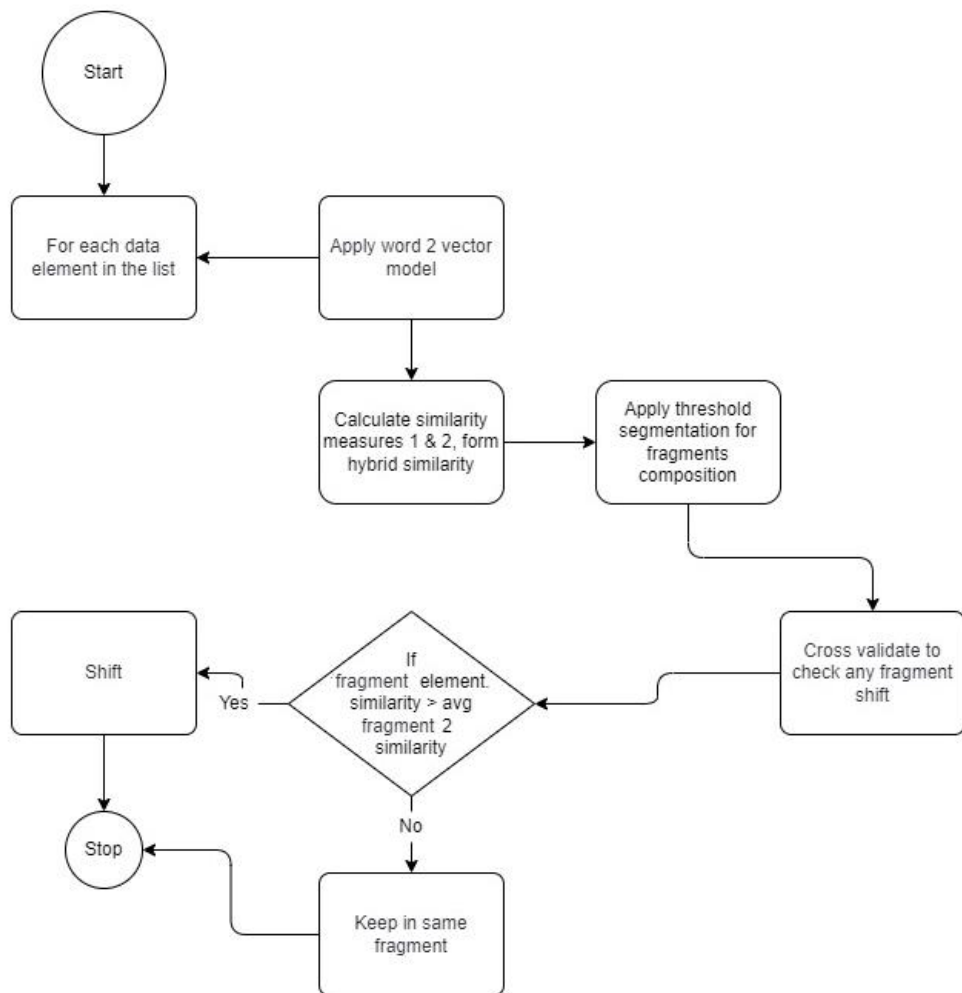


Figure 3.3 Data flow structure for fragmentation

3.2.1 Uploading of Twitter-Dataset

The dataset used in the proposed study comes from a sample of data and was accessed at [102]. The above-described dataset contains unlabeled tweets that must be segmented and for segmentation. The "text" column in this dataset contains 1048576 data instances, of which 93 are used to continue the relative fragmentation experiment as shown in **Figure 3.4**.

1	2	3	4	5	6	7	8
tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id	
119237	105834	1	'Wed Oct 11 06:55:44 +0000 2017'	'@AppleSupport causing the reply to be disregarded and the tapped notification under the keyboard l...	119236	NaN	
119238	'ChaseSu...	0	'Wed Oct 11 13:25:49 +0000 2017'	'"@105835 Your business means a lot to us. Please DM your name, zip code and additional details abo...	NaN	119239	
119239	105835	1	'Wed Oct 11 13:00:09 +0000 2017'	'@76328 I really hope you all change but I'm sure you won't! Because you don't have to!	119238	NaN	
119240	'VirginTra...	0	'Tue Oct 10 15:16:08 +0000 2017'	'"@105836 LiveChat is online at the moment - https://t.co/SY94VU8Kq or contact 03331 031 031 optio...	119241	119242	
119241	105836	1	'Tue Oct 10 15:17:21 +0000 2017'	'@VirginTrains see attached error message. I've tried leaving a voicemail several times in the past wee...	119243	119240	
119243	'VirginTra...	0	'Tue Oct 10 15:25:14 +0000 2017'	'@105836 Have you tried from another device, Miriam ^MM'	119244	119241	
119244	105836	1	'Tue Oct 10 15:26:44 +0000 2017'	'@VirginTrains yep. I've tried laptop too several times over the past week and again today. I've tried d...	119245	119243	
119245	'VirginTra...	0	'Tue Oct 10 15:33:22 +0000 2017'	'@105836 It's working OK from here, Miriam. Does this link help https://t.co/0m2mpH15eh ? ^MM'	NaN	119244	
119242	105836	1	'Tue Oct 10 15:09:00 +0000 2017'	'"@VirginTrains I still haven't heard & the number I'm directed to by phone is a dead end & ampr...	119240	119246	
119246	'VirginTra...	0	'Tue Oct 10 10:13:19 +0000 2017'	'@105836 That's what we're here for Miriam ðŸˆŠ The team should send you an email shortly ^HP'	119242	119247	
119248	'AppleSu...	0	'Wed Oct 11 13:38:29 +0000 2017'	'"@105837 We can help. Which version of iOS are you on? You can find that in Settings > General &...	NaN	119249	
119249	105837	1	'Wed Oct 11 07:37:27 +0000 2017'	'@105838 @AppleSupport Me too am suffering , hope the can find a solution'	119248	119250	
119250	105838	1	'Wed Oct 11 05:33:17 +0000 2017'	'"@AppleSupport hi #apple, lâ€™ve a concern about the latest ios is too slow on #iphone6 and i am n...	1.1925e+11	NaN	
119252	'AppleSu...	0	'Wed Oct 11 13:40:27 +0000 2017'	'"@105839 Thanks for reaching out to us. We are always happy to help. Send us a DM so we can look i...	NaN	119253	
119253	105839	1	'Wed Oct 11 07:21:34 +0000 2017'	'I just updated my phone and suddenly everything takes ages to load wtf @76099 this update sux I hat...	119252	NaN	
119254	'SpotifyC...	0	'Wed Oct 11 13:41:25 +0000 2017'	'"@105840 Hi there! What device is this happening on? If you could also let us know the Android and S...	119255	119256	
119255	105840	1	'Wed Oct 11 13:45:59 +0000 2017'	'"@SpotifyCares Thanks! Version 8.4.22.857 arm7 on anker bluetooth speaker on Samsung Galaxy Tab ...	119257	119254	
119257	'SpotifyC...	0	'Wed Oct 11 14:00:48 +0000 2017'	'"@105840 Thanks. The distance could possibly affect playback. Does logging out & restarting your ...	119258	119255	
119258	105840	1	'Wed Oct 11 14:01:58 +0000 2017'	'"@SpotifyCares No, but I've moved speaker to about 1 metre away and it's not skipping at the mo - it...	1.1925e+11	119257	
119259	'SpotifyC...	0	'Wed Oct 11 14:20:00 +0000 2017'	'@105840 That's great to hear. If anything comes up, just let us know. We'll carry on helping out ðŸ™, /...	119260	119258	
119260	105840	1	'Wed Oct 11 14:22:05 +0000 2017'	'@SpotifyCares Brilliant thanks ðŸˆŠ'	119261	119259	
119261	'SpotifyC...	0	'Wed Oct 11 14:41:35 +0000 2017'	'@105840 You're welcome! If there's anything else we can help with, just give us a shout. We're here f...	NaN	119260	
119256	105840	1	'Wed Oct 11 12:53:29 +0000 2017'	'@76495 @91226 Please helo! Scootiv Premium skioino through sonos constantiv on android tablet ...	119254	NaN	

Figure 3.4 Metadata of Twitter Dataset [102]

3.2.2 Removal of English Stop Words

A stop list [103] can be used to get a list of those stop words. The first step is to eliminate English stop words from unlabeled tweets. The technique of deleting English stop words from tweets (as illustrated in **Figure 3.5** and **Figure 3.6**) helps to reduce the dimension of the data accessible. The most prevalent words

in textual (tweet) files tend to be prepositions, articles, and nouns. These words don't add anything to the content. Because some words in information retrieval (IR) software are not named keywords, stop list words were deleted from the text [105]. To achieve this, data column 5 contain textual data used as data for experiment purpose. In this procedure, all rows are examined one by one till the conclusion of the records and compared for the elimination of stop words. Filter words are traced and turned into vectors at the end of the operation.

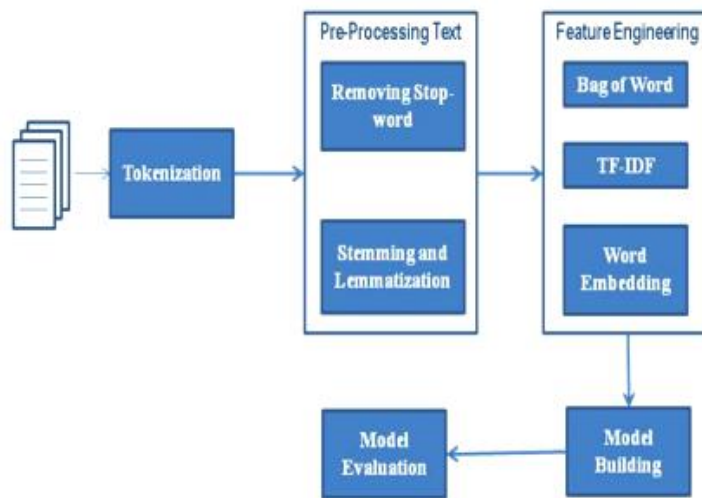


Figure 3.5 Stop Word Removal Process

```

datacol=5;
for i=2:r
    rawdata{i-1,1}=data{i,datacol};
end
stopwords=[];
stfilename='englishST.txt';
stopwords=textscan(stfilename,'%s');
[r,c]=size(rawdata);
for i=1:r
    currentdata=rawdata{i,1};
    words=getwords(currentdata);
    allwords{i}=words;
    filteredwords=filterdata(words,stopwords);
    allfiltered{i}=filteredwords;
    for wd=1: numel(filteredwords)
        w2v(i,wd)=sum(double(filteredwords{wd}));
    end
end
end
    
```

Figure 3.6 Stop Word Removal and W2V Conversion Code Snippet

3.2.3 Applying Word to Vector on Filtered Data

The similarity is commonly measured by comparing how similar or similar two objects are. One approach to calculating similarity is to use vectors. A vector is essentially a number with both magnitude and direction. In computer science, a one-dimensional sequence is referred to as a vector. Word2vec is a tool for creating a compact space of word vectors. It takes a large text corpus as input (tweets with stop words deleted) and allocates a vector to each word in the tweet.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1327	746	321	556	227	199	1150	307	321	638	1287	542	321	849	220	0
2	463	876	532	97	335	227	232	634	209	463	461	339	411	307	1049	742
3	105	649	428	349	313	614	331	253	447	349	528	728	349	476	420	0
4	848	220	645	213	321	656	45	831	1229	225	748	250	148	148	665	93
5	1312	317	830	554	741	363	536	742	97	953	754	546	215	321	440	428
6	420	349	536	436	753	668	639	0	0	0	0	0	0	0	0	0
7	1312	378	363	536	656	338	754	546	444	321	440	428	307	512	545	363
8	375	769	218	436	464	639	427	440	430	425	831	1096	63	0	0	0
9	1312	105	552	685	516	415	321	649	253	836	227	219	538	220	97	398
10	587	436	474	420	327	639	1580	321	423	655	426	349	207	520	789	0
11	220	306	425	531	774	213	331	312	349	284	349	306	417	433	215	881
12	1327	210	338	206	969	44	428	321	306	417	97	0	0	0	0	0
13	1327	209	609	17396	97	744	539	321	653	331	220	338	453	221	732	307
14	649	327	833	344	227	232	220	312	657	546	227	425	426	232	97	209
15	105	454	743	230	538	307	872	1093	536	416	227	416	337	440	643	352
16	209	569	436	624	220	440	954	284	207	349	535	431	325	232	447	321
17	1308	682	774	56	52	100	164	493	221	529	982	747	221	766	646	311
18	649	321	843	535	885	617	839	427	743	344	316	1091	463	624	316	743
19	1308	265	331	363	539	747	227	539	49	541	434	307	375	337	869	213
20	587	531	227	416	207	866	535	273	454	325	232	447	475	545	221	743
21	1308	961	649	0	0	0	0	0	0	0	0	0	0	0	0	0
22	603	781	207	690	866	425	220	306	425	488	454	427	232	97	563	474
23	634	458	782	767	869	769	554	1103	221	737	636	415	982	747	536	0
24	359	438	227	458	634	209	232	307	325	232	447	328	559	607	536	530
25	1327	530	321	98	48	50	230	538	454	553	451	213	321	436	312	685

Figure 3.7 Vector Representation of Filtered Words

The vector representation of the terms is obtained after generating a dictionary. Each row of words in column 5 is represented in the dataset by the values of the vector after approximation (as shown in **Figure 3.7**). As a result, the following stage is to assess the data's similarity index using three different similarity measures: Cosine, Soft Cosine, and hybrid similarity index. The next sections provide a full description of these three measurements. In word2vec, the two main models are Skip-gram and Continuous Bag of Words. In the Skip-gram model, terms are predicted from a word in their context, whereas in the CBOW model, the most likely word is chosen based on the context. The output layer uses a softmax feature or a combination of softmax features to get

the output of each term's probability distribution. In these models, input and output words are encoded in one-hot encoding, which means that when the matrix W connecting the input and hidden layers is multiplied by one, a single row is picked W . Because its lines contain vector representations of terms, dimension N is the algorithm hyperparameter and the qualified W -output matrix. [106].

Modifications to softmax, such as hierarchical softmax and negative sampling, are used to speed up the training of Skip-gram and CBOW models, allowing the probability distribution to be calculated faster than in linear time from dictionary size [107-109].

3.2.4 Applying Similarity Calculations

In the vector model, a document is defined as an unordered set of terms. The words that make up the text to obtain essential or useful information are referred to as retrieval terms. Here, the similarity index for the uploaded document to the rest of the text in the set is calculated using cosine similarity. For example, if the tweet contains 100 rows, the similarity (either cosine or soft cosine) is determined by comparing the word vector to the remaining 99 rows.

Input: Vector data
Output: simvalue=Calculatecossim(v1, v2)

1. Calculatecossim(v1, v2) = [];
2. nume = 0; //numerator
3. deno=0; //denominator
4. deno1=0;
5. deno2=0;
6. for I = 1 → v1.length
7. nume=nume+v1(I)*v2(I);
8. End for
9. for J = 1 → v1.length
10. deno1=deno1+v1(J)^2;
11. deno2=deno2+v2(J)^2;
12. End for
13. deno=sqrt(deno1)*sqrt(deno2);
14. simvalue=nume/deno;
15. Return: simvalue as output
16. End function

Pseudocode 1. Cosine Similarity on Vectors

The pseudocode 1, depicts the operation of cosine similarity by computing the cosine angles between two vectors v_1 and v_2 . To do this, each row in relation is compared to other rows using a vector list with the numerator and denominator as variables. It is calculated by multiplying each vector row by row consecutively and storing the result in the *nume* variable. *Deno* is computed by multiplying *deno1* and *deno2* by their square roots. *deno1* and *deno2* are the squares of v_1 and v_2 . Finally, *simvalue* is determined by dividing *nume* and *deno*.

Input: Word to Vector data

Output: $sc = \text{Calculatesoftcosine}(v_1, v_2)$

1. $\text{Calculatesoftcosine}(v_1, v_2) = []$
 2. $sc = 0;$
 3. $num = 0;$
 4. for $I = 1 \rightarrow v_1.length$
 5. for $J = 1 \rightarrow v_2.length$
 6. $num = num + v_1(I) * v_2(J);$
 7. End for
 8. End for
 9. $avalue = 0;$
 10. $bvalue = 0;$
 11. for $I = 1 \rightarrow v_1.length$
 12. for $J = 1 \rightarrow v_1.length$
 13. $avalue = avalue + v_1(I) * v_1(J);$
 14. $bvalue = bvalue + v_2(I) * v_2(J);$
 15. End for
 16. End for
 17. $avalue = \text{sqrt}(avalue);$
 18. $bvalue = \text{sqrt}(bvalue);$
 19. $deno = avalue * bvalue;$
 20. $sc = num / deno;$
 21. $sc = sc / (\max(v_1) / \max(v_2));$
 22. End function
-

Pseudocode 2. Soft Cosine Similarity on Vectors

An enhanced technique known as soft cosine similarity is developed to attain precision in the outcome of cosine similarity. Soft cosine is calculated using pseudocode 2 by dividing the numerator and denominator. For each row, the numerator is the multiplication of v_1 and v_2 with other available rows in the relations. On the other hand, the denominator for each row and column is the square root of v_1 and v_2 .

Input: calculated results of cosine similarity and soft cosine similarity

Output: allhybrid=hybridsim()

1. [r,c]=size(w2v);
2. allcossim=[];
3. allsoftcossim=[];
4. allhybrid=[];
5. for i=1→r
6. v1=w2v(i,:);
7. simvalue=0;
8. softvalue=0;
9. counter=0;
10. for j=i+1→r
11. v2=w2v(j,:);
12. simvalue=simvalue+Calculatecossim(v1,v2);
13. softvalue=softvalue+Calculatesoftcosine(v1,v2);
14. counter=counter+1;
15. End for
16. allcossim(i)=simvalue/counter;
17. allsoftcossim(i)=softvalue/counter;
18. allhybrid(i)=allcossim(i)+allsoftcossim(i);
19. End for
20. End function

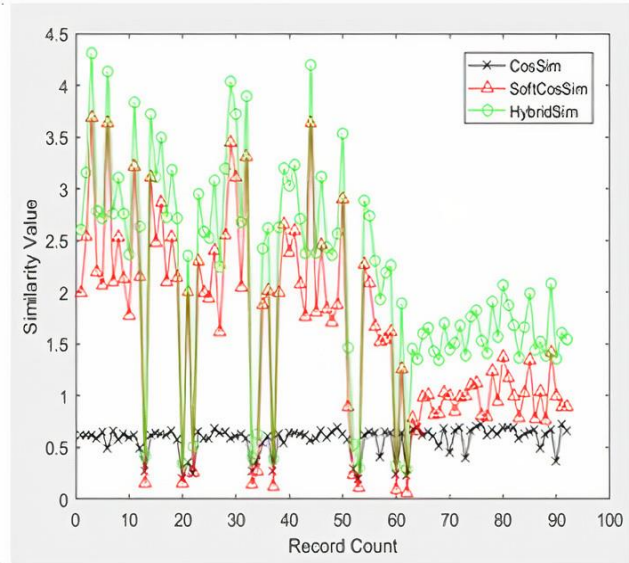
Pseudocode 3. Hybrid Similarity

The findings of cosine and soft-cosine similarity are used in **pseudocode 3**. To begin, determine the size of the vector to determine the number of rows and columns. Then, using $w2v$, get the values of the vectors $v1$ and $v2$ from each word in the rows. Call *Calculatecossim()* on the $v1$ and $v2$ variables to get the similarity value. Call *Calculatesoftcosine()* on the $v1$ and $v2$ variables to get the Soft Cosine similarity value. Finally, for each row, add the average of cosine and soft cosine similarity to get the hybrid similarity.

Columns 1, 2, and 3 of (a) in **Figure 3.8** reflect the results of cosine, soft-cosine, and hybrid similarity calculations, respectively. And (b) shows the similarity index graphical representation after applying Cosine, Soft Cosine, and hybrid similarity measures.

	1	2	3
1	0.6100	1.9951	2.6051
2	0.6171	2.5406	3.1577
3	0.6187	3.6904	4.3091
4	0.5833	2.1993	2.7827
5	0.6422	2.0723	2.7145
6	0.4920	3.6406	4.1326
7	0.6599	2.1021	2.7621
8	0.5732	2.5343	3.1074
9	0.6296	2.1303	2.7599
10	0.5888	1.7783	2.3671
11	0.6188	3.2159	3.8346
12	0.4858	2.1490	2.6349
13	0.2629	0.1535	0.4164
14	0.6124	3.1095	3.7219
15	0.6318	2.4824	3.1142
16	0.6272	2.8681	3.4953
17	0.6190	2.1062	2.7253
18	0.6536	2.5292	3.1828
19	0.5773	2.1383	2.7156

(a)



(b)

Figure 3.8 (a) Outcome after similarity measure (b) Similarity index graphical representation

The average similarity index is calculated by evaluating the final output for all tweets in the row. The following equations reflect the cosine, soft cosine, and hybrid similarity index formulas:

$$\text{Cosine Similarity} = \sum_{i=1}^n \text{cosinesimilarity} / n \quad (1.1)$$

$$\text{Soft Cosine Similarity} = \sum_{i=1}^n \text{Softcosinesimilarity} / n \quad (1.2)$$

$$\text{Hybrid Similarity} = \text{Cosine} + \text{Soft Cosine} \quad (1.3)$$

$$\text{Set} = [\text{Cos}_i, \text{soft}_i, \text{hybrid}_i] \quad (1.4)$$

3.2.5 Finding Initial Centroid (IC) of Data

The Initial Centroid (IC) of the tweet data is determined next, which is derived as the average of each similarity measure acquired from each similarity index (Cosine, soft cosine, and hybrid) independently. This value is used as the tweet's IC. The formula for calculating the IC is given by equation (1.5).

$$\text{Initial Centroid (IC)} = \left[\sum_{i=1}^k \frac{\text{All Cosine}}{K}, \sum_{i=1}^k \frac{\text{All softCosine}}{K}, \sum_{i=1}^k \frac{\text{All hybrid}}{K} \right] \quad (1.5)$$

Where k = Total number of tweets

The "X" sign represents the centroid of each cluster, as seen in **Figure 3.9**. The available data is divided into three clusters, which are labeled cluster1, cluster2, and cluster3, and are represented by distinct colors, blue, red, and orange, respectively.

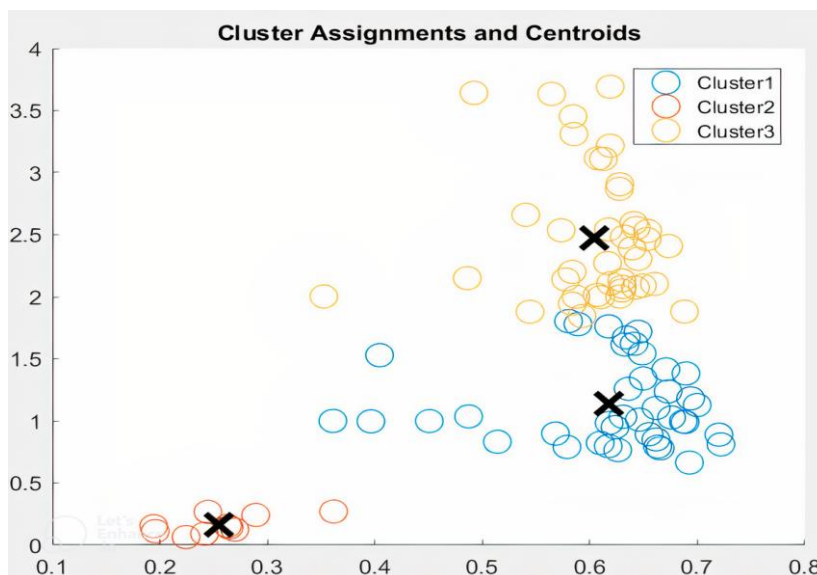


Figure 3.9 Representation of Clusters and Centroids

3.2.6 Calculation of Euclid Distance

The geometric distance in multidimensional space is known as the Euclidean distance. The following formula is used to compute the Euclidean distance between points T1 and T2 in n-dimensional space (1.6). The equations below reflect the formula used to determine the Euclidian distance of each tweet from the set (ST), which is calculated using equation (1.4) for cosine data, soft cosine data, and hybrid data:

$$D1 = ECU1(IC,ST) \quad (1.6)$$

$$D2 = Squared ECU1(IC,ST) \quad (1.7)$$

$$D3 = (D1 + D2) / 2 \quad (1.8)$$

It is worth noting that the Euclidean distance (and its square) are calculated using the Tweet data received in the previous stage. The total number of centroids in the given data was then calculated to determine the number of fragments by measuring the average of D1, D2, and D3.

D is the total number of centroids.

$$D = \frac{D1 + D2 + D3}{2} \tag{1.9}$$

Figure 3.10 depicts an example of a distance calculated from ST that is D1, D2, D3.

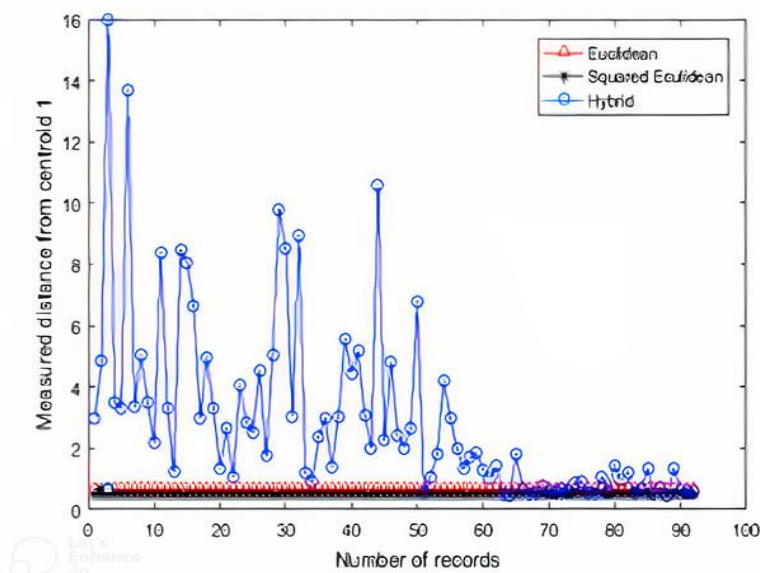


Figure 3.10 Measures Distance from the Centroid

3.2.7 Fragment Generations

Data fragmentation is the process of splitting data into fragments to make storage easier. The formula used to determine the number of pieces from the supplied data is expressed as equation (1.10).

$$p = \frac{\sqrt{D \times K}}{c} \tag{1.10}$$

where K = total number of tweets (rows)

C = number of centroids

and D = Total numbers of fragments

The fragmented data is fed into the Artificial Neural Network (ANN) alongside the original word 2 vector data. **Table 3.1** shows the classified fragments for the 100, 200, 300, 400, and 500 rows. It shows fragments distributions individually for each row after classified by ANN.

For 100 Rows	For 200 Rows	For 300 Rows	For 400 Rows	For 500 Rows
1	1	2	4	3
1	1	2	2	2
2	1	1	1	1
1	2	3	3	2
1	2	1	1	3
1	1	2	2	2
2	1	1	2	4
1	3	2	3	4
1	2	3	3	3
3	1	4	4	3
2	2	3	2	2

Table 3.1 Fragments Classification Row-Wise

3.3 Results and Discussions

A MATLAB program with 4 GB RAM, a 64-bit operating system, and a 2.30 GHz processor was used to implement the fragmented architecture. In terms of categorized accuracy, the performance has been evaluated. For 100, 200, 300, 400, and 500 rows, the results were examined individually.

Table 4.2 shows the results of five experiments that were conducted to measure detection accuracy. The accuracy of the designed fragmented structure's classification is shown in **Figure 3.11**. The simulations were run five times to ensure that the submitted data, which may comprise rows (100, 200, 300, 400, and 500), was as accurate as possible. The accuracy of classification increases as the number of rows grows, as shown in the graph. This is because as the amount of data grows, so does the ability to train an ANN structure, resulting in better classification of fragmented data. For 100, 200, 300, 400, and 500 rows, the classification accuracy averages 63.81, 76.28, 81.52, 83.58, and 92.078, respectively.

Table 3.2 Classification Accuracy for Each Row

Iterations	100	200	300	400	500
1	62.45	75.89	82.15	84.57	91.04
2	63.57	76.28	81.27	83.57	92.57
3	62.78	76.18	79.68	84.25	93.17
4	64.28	75.12	80.15	82.37	92.14
5	65.97	77.94	84.36	83.14	91.47

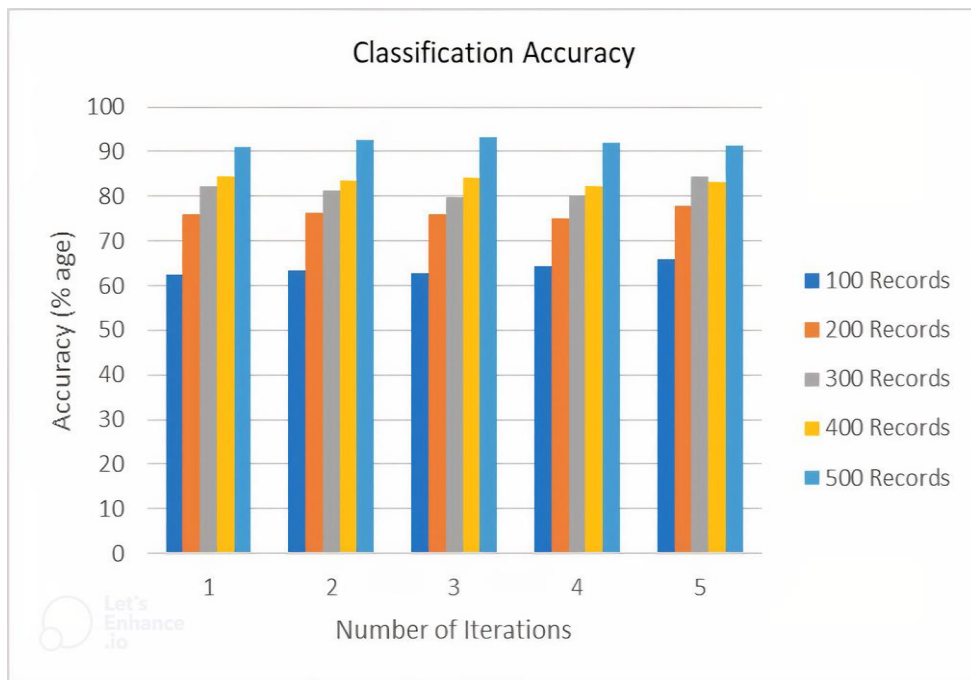


Figure 3.11 Classification Accuracy

The average percentage of iterations is derived to evaluate precision and recall, as shown in Table 3.3. Table 3.4 shows the fragmented architecture developed after the suggested work was analyzed using parameters such as True Positive, True Negative, False Negative, and False Positive. Table 3.5 shows the precision and recall.

$$\text{True Positive (Tp)} = \text{Total true selected elements} / \text{Total sample size} \quad (1.11)$$

$$\text{False Positive (Fp)} = \text{False selected elements} / \text{Total sample size} \quad (1.12)$$

$$\text{True Negative (Tn)} = \text{True left samples} / \text{Total sample Size} \quad (1.13)$$

$$\text{False Negative (Fn)} = \text{False left sample} / \text{Total sample Size} \quad (1.14)$$

Table 4.5 displays the precision and recall, which are nearly the same for each row, and **Figure 3.11 and 3.13** shows the graphical depiction of the Tp, Tn, Fp, Fn evaluation and precision and recall respectively.

The Fp value reveals that the components are not placed in clusters that are suitable for them. In this study, the Fp results are lower. Tn displaying the remaining bad samples. It shows a decent search response if it is high. Precision and recall values are calculated based on the values calculated for Tp, Tn, Fp, and Fn. The simulation of Tp, Tn, Fp, Fn is shown in Figure 3.12.

Table 3.3 Average Percentage of Iterations

No. of Rows/Records	Average of all five iterations in %
100	64
200	76
300	82
400	84
500	92

Table 3.4 Evaluation of Tp, Fp, Tn, and Fn

Total Passed Query	100	200	300	400	500
Total true selected samples (Tp)	0.45	0.53	0.57	0.59	0.64
True left samples (Tn)	0.25	0.17	0.13	0.11	0.11
False left samples (Fn)	0.44	0.53	0.57	0.58	0.64
False selected sample (Fp)	0.25	0.08	0.04	0.03	0.01

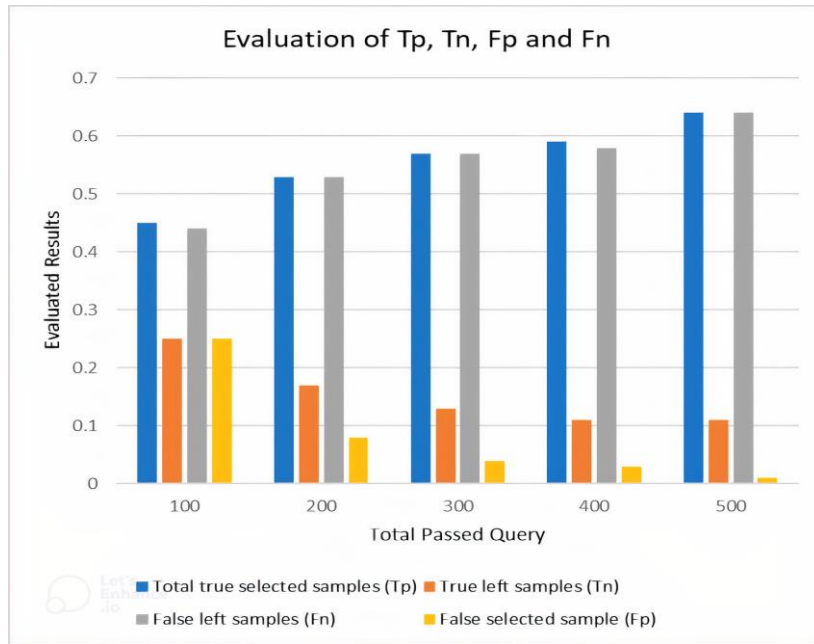


Figure 3.12 True Positive-False Positive-True Negative-Fast Negative

Total Passed Rows	100	200	300	400	500
Precision	0.64	0.87	0.93	0.95	0.98
Recall	0.64	0.76	0.82	0.84	0.85

Table 3.5 Evaluation of Precision and Recall



Figure 3.13 Precision and Recall

3.4 Comparative Analysis

Different fragmentation algorithms are proposed by the researchers to work for enhancing performance. These algorithms work for fragmentation and work for controlling transfer cost (TC), communication Cost (CC), and access time (AT). and response time (RT). The proposed work used performance parameters including precision and recall to achieve optimal results. The parameters of different techniques are hereby discussed below in Table 3.6.

Table 3.6 Comparative analysis of parameters followed by different fragmentation techniques

References	Algorithms/ Techniques Used	Cost parameters used					Performance parameters	
		Storage Cost	TC	CC	AT	RT	Precision	Recall
[109]	K-Means and SMOTE	X	✓	✓	✓	✓	✓	✓
[110]	Hybrid Similarity based partitioning	X	✓	✓	✓	✓	✓	✓
[33]	Modified Bond Energy algorithm	X	✓	✓	✓	✓	X	X
[129]	Hybrid Ant Colony Algorithm (HACA)	X	✓	✓	✓	✓	✓	X
Proposed Work	The similarity- based threshold segmentation technique	✓	✓	✓	✓	✓	✓	✓

For total passed rows of 500, the maximum precision achieved is 0.98. Every row passed has a recall value of 0.85. It is discovered that the proposed work has improved in terms of precision and recall. Researchers previously employed the k-means dependent cosine similarity measurement method to estimate feature similarity between cluster centroids and data points to quantify the similarity between the clustering output and side information. A data partitioning clustering technique based on the notion of a learning approach has been suggested. The suggested work's fundamental flaw is that it only uses cosine similarity-based k-means for dividing big data sets [110]. On the contrary proposed approach uses a similarity-based threshold segmentation method clustering-wise to reach optimal precision and recall results.

To increase precision and recall parameters during partitioning, an effort is also made to hybridize cosine and soft-cosine [111]. We offered cosine, soft cosine, and hybrid similarity as an increased method in this study, and the result was a rise of 0.98 precision and 0.85 recall values. As a result, we rely on this technique to fragment textual data for a variety of systems.

In his research, *Huaping Guo* [109] suggested a classification system based on data-partitioning and SMOTE for unbalanced learning, achieving precision and recall rates of 0.47 and 0.54 in percentage, respectively. On the other hand, the author in [110] offered a strategy to develop a partitioning method based on the combination of cosine and soft cosine similarities to increase the data partitioning performance with improved portioning speed and accuracy. It shows 0.69 and 0.67 precision and recalls in percentage.

The suggested approach's computed accuracy and recall are based on the classification technique's estimated precision and recall of 0.98 and 0.85 for the proposed method, respectively after the data are classified into clusters. In **Table 3.7**, the comparisons with various methodologies are shown or distinguished. The Graphical Views of Precision and Recall are shown in **Figure 3.14**.

Table 3.7 Comparison of calculated Precision and Recall

Parameters	Huaping Guo et al. [109]	Kiranjeet Kaur et al. [110]	Proposed Work
Precision	0.47	0.69	0.98
Recall	0.54	0.67	0.85

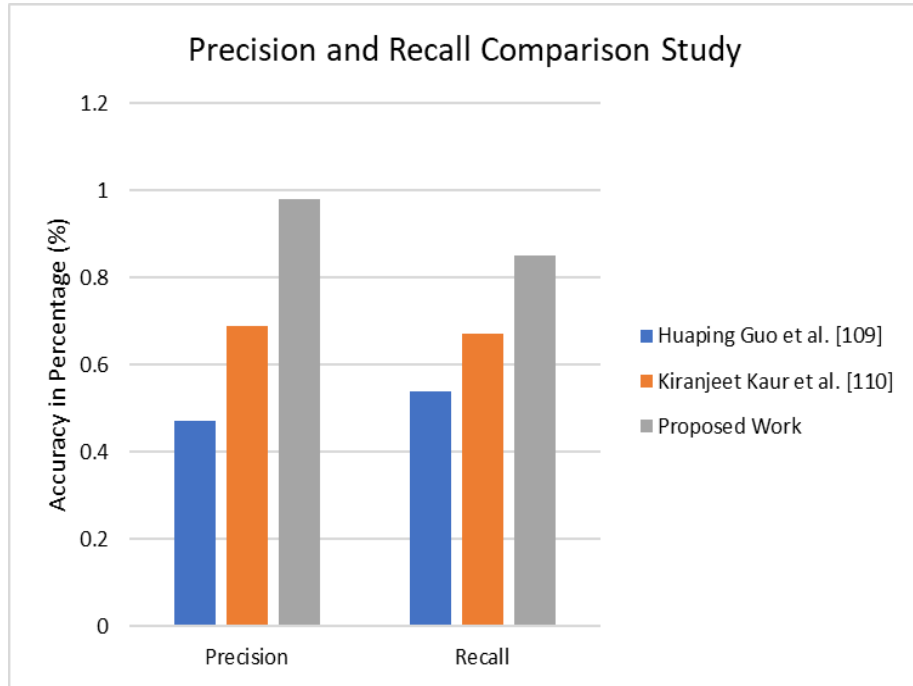


Figure 3.14 Precision and Recall Comparison

The following are some of the benefits/differences of the proposed work over existing implemented methodologies:

- It aids in the classification of fragmented data with high precision and recalls indicating maximum coverage, and accuracy, and reducing overall processing time.
- Previous clustering strategies based on cosine-based k-means, cosine, and soft cosine hybridization failed due to a lack of balance in the quality and efficiency of clustering in categorical data sets.
- Existing techniques' concepts are fine in certain circumstances, but not in others. As a result, algorithm hybridization is the optimal strategy. To ensure enhanced efficiency and the ability to cope with big data sets, cosine and soft cosine similarity notions are used to compute hybrid similarity in this research work.

3.5 Conclusion

A novel relative data fragmentation architecture is proposed in this study to partition a huge dataset into many fragments. Twitter data is used in this project, and it is turned into vectors to be used for fragmentation. Calculations of cosine, soft cosine, and hybrid similarity are performed, and centroid positions are discovered. To find clusters, the K-Mean technique is utilized to calculate the distance between data points with each centroid. Finally, validation and performance are checked using ANN to ensure accuracy. The goal of this study is to implement a unique similarity-based data fragmentation architecture in an unsupervised learning environment. When compared to previously proposed approaches, the comparison yielded good precision and recall.

Introduce a Novel Data Allocation Scheme to optimize the Allocation Process Based on Swarm Intelligence

Efforts to increase the flexibility and efficacy of distributed computing systems are continuing. These achievements are the result of extensive research in the domains of connection technologies, network programs, high-performance computer components, and storage. However, delays in answers, long execution times, and long completion times have been highlighted as stumbling blocks that impair performance and need more attention. Such shortcomings raised the total cost of the system and impeded data dissemination in a geographically dispersed structure. Data allocation performance might be improved by improving the load-based architectural model.

4.1. Introduction

In this objective, an abstract job model is used to accomplish this, and a data query file containing input data is processed on a directed acyclic graph. The task is run on the processing engine with the lowest execution cost, and the total cost of the system is computed. Communication, computation, and network costs are all added together to get the overall cost. The way data is fragmented and dispersed across multiple geographically dispersed sites determines the success of a distributed system [99]. Overcoming the escalating labor load, and reducing data allocation costs are two of the most difficult components of distributed architecture. A higher load means a longer query completion time, which impacts operational costs and raises the overall system execution cost. Data pieces must be split down into little sub-tasks and scheduled to be handled in parallel to keep the overall system execution cost low. The goal of this form of heuristic division is to cut down on the amount of time it takes to complete all tasks.

The query file has details about tasks, communication, and computation cost incurred by each processing engine. It determines how tasks are executed among an available number of processing engines. For estimating task execution costs, each engine has its own computing cost. The execution cost is estimated by adding the processing engine's lowest execution cost to the communication and network costs incurred during execution.

A directed acyclic graph (DAG) comprising vertices and edges, as shown in **Figure 4.1**, was used to parallelize all of the jobs. Each job/task is represented in a DAG as a vertex or node, with the links between them reflecting the communication overhead and connectivity between tasks. When data is exchanged between nodes, the cost of communication is incurred. The DAG is in care of three different types of costs: communication, network, and computing. Each expense was taken into account while calculating the total cost of system execution. For total cost calculations, only the calculation cost with the shortest execution time is considered.

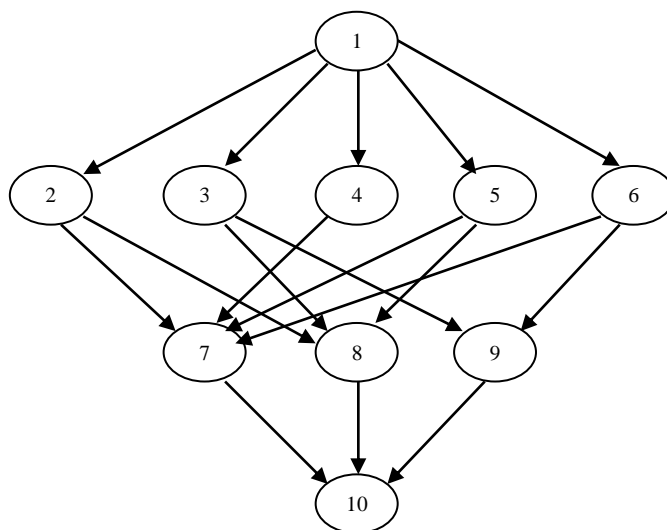


Figure 4.1. Directed Acyclic Graph

In this research work, we look at a distributed system with heterogeneous processors and system tasks. A distributed system is made up of $P = P_1, P_2, \dots, P_n$ heterogeneous processors coupled by communications links and $T = t_1, t_2, \dots, t_m$ system tasks that collectively express a purpose. As shown in **Tables 4.1**

and 4.2, the execution cost matrix (ECM) is an asymmetrical matrix of order $m*m$ that represents the cost of executing tasks on different processors, while the network cost matrix (NCM) is an asymmetrical matrix of order $m*m$ that represents the cost of communication between multiple tasks. The suggested ABC algorithm is used to compute the best total cost by combining both.

Table 5.1 Execution Cost Matrix (ECM)

Query/Tasks	Processors		
	P1	P2	P3
1	0.81	0.16	0.66
2	0.91	0.97	0.04
3	0.13	0.96	0.85
4	0.91	0.49	0.93
5	0.63	0.80	0.68
6	0.10	0.14	0.76
7	0.28	0.42	0.74
8	0.55	0.92	0.39
9	0.96	0.79	0.66
10	0.96	0.96	0.17

Table 4.2 Network Cost Matrix (NCM)

Query/Tasks	Processors		
	P1	P2	P3
1	0.71	0.44	0.28
2	0.03	0.38	0.68
3	0.28	0.77	0.66
4	0.05	0.80	0.16
5	0.10	0.19	0.12
6	0.82	0.49	0.50
7	0.69	0.45	0.96
8	0.32	0.65	0.34
9	0.95	0.71	0.59
10	0.03	0.75	0.22

The challenges with data allocation in a distributed system are response time, execution time, and completion time [112]. It gradually raises system expenses and impedes workforce advancement. The use of multiple processors directly impacts network and communication costs during task execution, which affects the overall system cost. The researchers previously utilized methodologies that eliminated network costs from overall system cost assessments. In this case, the previously obtained study results appear to be incorrect and useless in many trials conducted in a dispersed setting. The suggested research focuses on a swarm intelligence-based artificial bee colony method that can be used to address and resolve present problems as well as quickly work on earlier errors. Previous difficulties are discussed in [114-118]. The proposed strategy is based on bees' learning and adaptive behavior, which could help with performance issues. It controls bee degradation loss as a result of the expensive cost of storing the bees from the site. To balance loss, such costs are removed from the overall system cost. In a distributed computing system, it contributes to the enhancement of data allocation.

4.2 Proposed Methodology and Implementation

The processing engine determines the data allocation technique. If the cost parameter is not properly regulated, the data allocation procedure consumes more network costs. In this instance, the cost of doing actions on several processors varies. The processing cost of two dependent jobs running on the same processor is the same, and communication between them is considered zero. To compute execution cost, the intended operations are repeated on the specific processing engines. The lowest cost of query execution is applied to the communication cost to determine the task execution cost. The following steps are followed to complete the scheduled activity, as shown in **Figure 4.2**:

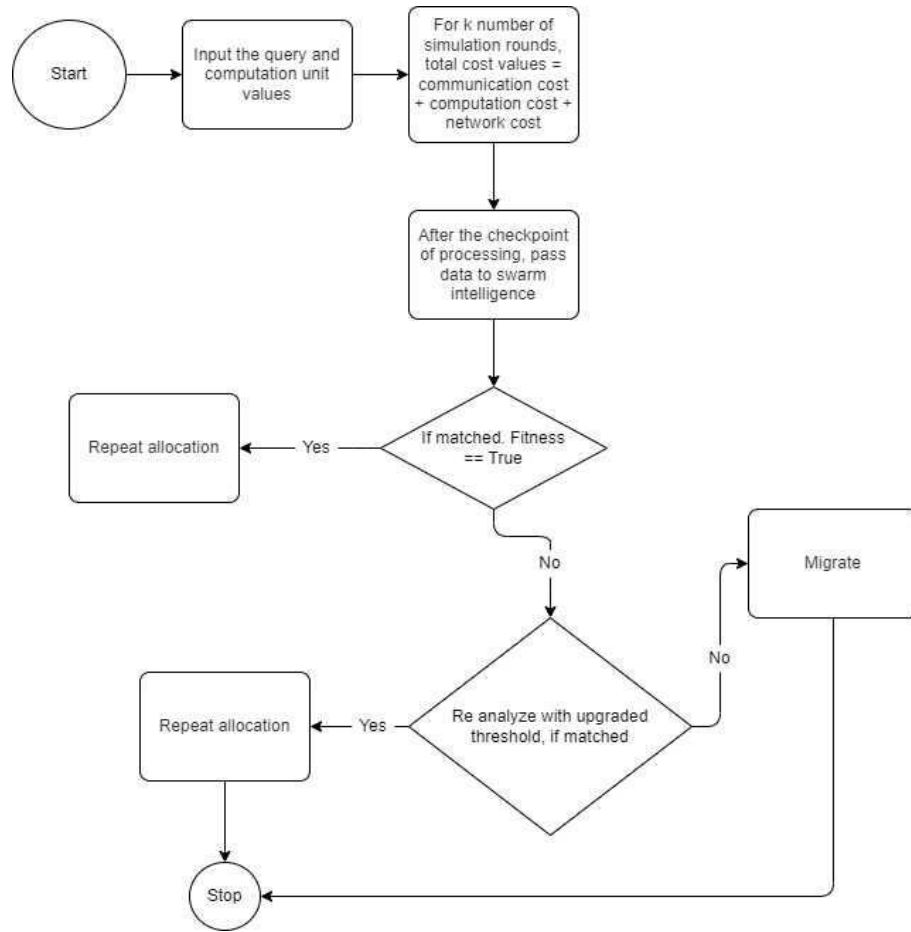


Figure 4.2 Proposed Model Flow

4.2.1 Loading of Query File

The query model flow provides a complete list of all jobs, their flows, and the communication and computes charges associated with each processing engine, as illustrated in **Table 4.3** below. It is a tabular representation of a directed acyclic graph (DAG). Query tasks are represented as flows from the root node to their connected nodes, and so on. The communication cost is the cost that appears after two tasks have been completed and communication or interaction has occurred between them. And, when several processing engines are used, the engine with the lowest computing cost is chosen for continued execution. The least expensive processing engine fit for the tasks at hand is chosen in this case.

Table 4.3 Query Model Flow

Query/Tasks flow		Communication Cost	Computation Cost		
T1	T2		PE1	PE2	PE3
1	2	12	12	10	14
1	3	15	8	9	15
1	4	17	14	18	15
1	5	14	11	14	9
1	6	16	12	14	17
2	7	14	11	15	17
2	8	15	12	17	19
3	8	18	12	17	19
3	9	17	17	18	13
4	7	18	11	15	17
5	7	17	11	15	17
5	8	25	12	17	19
6	9	10	17	18	13
6	7	14	11	15	17
7	10	12	14	15	13
8	10	13	14	15	13
9	10	14	14	15	13

4.2.2 Total Cost Calculation

The overall cost is equal to the total amount of energy used to complete all tasks. Overall expenses are calculated at the node level using a level-by-level cumulative computation. The total cost of communication, computation, and network access for all jobs is added up at each level to determine load. All three parameters combine to reflect the total cost of selecting processors for task execution. Each CPU has its own set of characteristics, and they all do tasks within the operating system's priority limits. It means that we won't be able to interrupt the workflow. After each of its parents has completed their jobs, sub-tasks are carried out. Instead of prioritizing tasks at each node, task assessment is prioritized for processors with lower execution costs.

4.2.2.1 Level-Wise Estimation of Execution Patterns

To calculate the total cost, jobs at various levels are processed in topological order, and their associated communication, network, and computing costs from

the communicating node are calculated and added. This method is used for all jobs and lies on different levels (0-nth levels) depending on DAG size. The overall execution cost is calculated by summing the results computed at each level.

To implement calculations at each node level-wise following pseudocodes are used and shown below:

4.2.2.1.1 Pseudocode to Compute Level-Wise Execution-Cost

Level-I:

Input: *entry_query, query_comp_cost, myminvalue, myminpos*

Output: *total_engine_value*

1. `execution_pattern=[];`
2. `total_engine_value=zeros(1,3);`
3. `[myminvalue,myminpos]=min(query_comp_cost(1,:));`
4. `execution_pattern(1,1)=entry_query;`
5. `execution_pattern(1,2)=0;`
6. `execution_pattern(1,3)=myminvalue;`
7. `execution_pattern(1,4)=myminpos;`
8. `total_engine_value(1,myminpos)=total_engine_value(1,myminpos)+myminvalue;`

Level-II:

Input: *last_engine, total_query_count, lvcount2, current, parent, query_engine*

Output: *comm_costt, total_cost, total_engine_value*

1. `last_engine=myminpos;`
2. `total_cost=[];`
3. `total_query_count=2; % disp(execution_pattern);`
4. `for i=1:lvcount2 % return level-2 queries count`
5. `current=level(2,i);`
6. `parent=entry_query;`
7. `for j=1:query_engines`
8. `comp_current=query_comp_cost(current,j);`
9. `if j~=last_engine`
10. `comm_costt=0;`

```

11. sd=find(query_comm_cost(:,1)==entry_query ); % return no. of queries related
    to 1
12. for k=1:numel(sd)
13. kp=query_comm_cost(sd(k),2);
14. if kp==current
15. comm_cost=query_comm_cost(k,3);
16. end
17. end
18. total_cost(j)=comm_cost+comp_current+total_engine_value(1,j);
19. else
20. total_cost(j)=comp_current+total_engine_value(1,j);
21. end
22. end

```

Level-III:

Input: lvcounter, query_data, execution_pattern

Output: parentcurrent, parentfinishtime, totalcost

```

1   for i=1:lvcounter           % Tally execution pattern of level 3
2   current=lv3jobs(i);
3   parentcurrent=[ ];
4   parentfinishtime=[ ];
5   counter=0;
6   [dp,pos]=find(query_data(:,2)==current);
7   for j=1:numel(dp)
8   parentcurrent(j)=query_data(dp(j),1);
9   currentparent=parentcurrent(j);
10  m=find(execution_pattern(:,1)==currentparent);
11  currentparentfinishtime=execution_pattern(m,3);
12  parentfinishtime(j)=currentparentfinishtime;
13  parentprocessor(j)=execution_pattern(m,4);
14  end
15  [maxval,maxpos]=max(parentfinishtime);
16  minstarttime=maxval;
17  parentp=parentprocessor(maxpos);

```

```
18 totalcost=[ ];
19 for j=1:3
20 [p,k]=find(execution_pattern(:,4)==j);
21 lasttime=execution_pattern(p(numel(p)),3);
22 if lasttime<minstarttime
23 lasttime=minstarttime;
24 end
25 totalcost(j)=lasttime;
26 end
```

Level -IV

Input: lvcounter, query_data, execution_pattern

Output: parentcurrent, parentfinishtime, totalcost

```
1. for i=1:lvcounter
2. current=lv4jobs(i);
3. parentcurrent=[ ];
4. parentfinishtime=[ ];
5. counter=0;
6. [dp,pos]=find(query_data(:,2)==current);
7. for j=1:numel(dp)
8. parentcurrent(j)=query_data(dp(j),1);
9. currentparent=parentcurrent(j);
10. m=find(execution_pattern(:,1)==currentparent);
11. currentparentfinishtime=execution_pattern(m,3);
12. parentfinishtime(j)=currentparentfinishtime;
13. parentprocessor(j)=execution_pattern(m,4);
14. end
15. [maxval,maxpos]=max(parentfinishtime);
16. minstarttime=maxval;
17. parentp=parentprocessor(maxpos);
18. totalcost=[ ];
19. for j=1:3
20. [p,k]=find(execution_pattern(:,4)==j);
21. lasttime=execution_pattern(p(numel(p)),3);
```

22. if lasttime<minstarttime
23. lasttime=minstarttime;
24. end
25. totalcost(j)=lasttime;
26. end

The cost of the processing engine's execution for each task is calculated using a formula (1). The formula is used to get the lowest execution cost (2). The overall cost of the system is computed by calculating the lowest execution cost of all tasks by the formula (3). **Table 4.4** shows the practical implementation calculation for each processing engine.

$$\text{PEs Execution Cost} = \text{communication cost} + \text{computation cost} + \text{network cost} \quad (1)$$

$$\text{Least Execution Cost} = \min(\text{execution cost of each PEs}) \quad (2)$$

$$\text{Total System Execution Cost} = \text{sum of the least execution cost of all tasks} \quad (3)$$

Table 4.4 Estimation of Total System Cost

Query/ Tasks/Jobs	Computation Cost of Tasks on PEs			Execution Cost of Each PEs			Least Execution Cost	Total System Cost
	P1	P2	P3	P1	P2	P3		
1	9	8	7	0	0	7	7	
2	12	10	14	24	22	21	21	
3	8	9	15	23	24	43	23	
4	14	18	15	54	35	43	35	
5	11	14	9	48	63	37	37	
6	12	14	17	51	65	82	51	396
7	11	15	17	53	57	60	53	
8	12	17	19	53	45	37	45	
9	17	18	13	63	60	61	60	
10	14	15	13	64	72	70	64	

4.2.2.2 Execution Cost Patterns Task-Wise

As indicated in **Table 4.5**, the execution pattern for each processing engine task is computed using the ECM and NCM results.

4.2.2.2.1 Pseudocode for the calculation of execution pattern task-wise

Input: execution_pattern, network_cost, currentpt, total_query_count, ex_pt

Output: ecost

```

1.   ex_pt=[ ];
2.   ex_pt{1,1}='Query No';
3.   ex_pt{1,2}='Starting Consumption Unit';
4.   ex_pt{1,3}='Ending Consumption Unit';
5.   ex_pt{1,4}='DB Engine ID';
6.   total_query_count=10;
7.   for i=1:total_query_count
8.     ex_pt{i+1,1}=execution_pattern(i,1);
9.     ex_pt{i+1,2}=execution_pattern(i,2);
10.    ex_pt{i+1,3}=execution_pattern(i,3);
11.    ex_pt{i+1,4}=execution_pattern(i,4);
12.  end
13.  for i=1:total_query_count
14.    currentdiff=execution_pattern(i,3)-execution_pattern(i,2);
15.    currentp=execution_pattern(i,4);
16.    ecost=energypattern(i,currentp);
17.    ecost=ecost+networkcost(i,currentp);
18.    execution_pattern(i,5)=ecost;
19.  end

```

Table 4.5 Execution Cost Pattern for Each Processing Tasks

Tasks	Starting Consumption Unit	Ending Consumption Unit	Processing Engines ID	Execution Cost Pattern (mJ)
1	0	7	3	6.5224
2	7	21	3	0.7154
3	0	23	1	0.4039
4	0	35	2	1.2806
5	28	37	3	0.7977
6	23	51	1	0.9210
7	51	53	1	0.9733
8	37	45	2	1.5620
9	51	60	2	1.5016
10	60	64	1	0.9993

4.3 Proposed Artificial Bee Colony (ABC) Algorithm

The swarm intelligence algorithm is a step toward solving problems that traditional numerical approaches can't solve. Honey bees are an example of a rapid social collective behavior that can adapt, learn, and update itself. The majority of the researchers were inspired to use it to improve their outcomes. This algorithm is based on the behavior of bee colonies. Employed bees (those responsible for food collection), observer bees (those responsible for food monitoring), and scout bees (those are in rest) are the three categories of bees found here. The ABC algorithm is used to optimize the total execution cost.

4.3.1 Artificial Bee Colony pseudocode

Input: Food Source

Output: [scout, beedegradation]=beefitness(employed_bee, energypattern, networkcost, timemodel, currentprocessor, taskname)

```

1. scout=0;
2. beedegradation=0;
3. restprocessors=[ ];
4. rc=1;
5. for i=1:3
6.   if i~=currentprocessor
7.     restprocessors(rc)=i;
8.     rc=rc+1;
9.   end
10. end
11. for i=1:numel(restprocessors)
12.
13.   onlooker_bee_value(i)=timemodel*(energypattern(taskname,restprocessors(i)), networkcost(taskname,restprocessors(i)));
14. end
15. selected_food_source=min(onlooker_bee_value);
16. onlooker_bee_selection=selected_food_source;
17. employed_bee=employed_bee*timemodel;
18. natural_change_onlooker=rand;
19. natural_change_employed=rand;
20. if onlooker_bee_selection*natural_change_onlooker > employed_bee *
21.   natural_change_employed
22.   scout=0;
23.   beedegradation=0;
24. else

```

```

23.  scout=1;
24.  beedegradation=((employed_bee*natural_change_employed) -
    (onlooker_bee_selection * natural_change_onlooker ) ) / timemodel;
25.  end
26.  end

```

The fitness function above lines 19-25 aids in measuring the overall execution cost in distributed systems to achieve low-cost data allocation. It compares and ensures that the outputs, as well as the bee deterioration component, are optimal once the processes are completed.

4.3.2 Re-Analyse of Optimal Result using Upgraded Threshold

Re-Analyse of optimal result is applied to an upgraded threshold value. This is accomplished by subtracting the cost of bee degradation from the total system cost.

Input: *scout, beed, timemodal*

Output: *Optimal*

```

1.  if scout>0
2.  optimal(i,5) =abs(optimal(i,5)-beed/timemodel);
3.  end

```

The measurement results of fitness value along with the re-analyzing process are carried out to achieve optimal results using the ABC algorithm. The calculated result values are evaluated and results are discovered as shown in **Table 4.6**.

Table 4.6 Fitness Value and Re-Analyse Process for Optimal Result

Tasks	Current Processor	Rest Processor	Onlooker_ bee_value	onlooker_ bee_selection (A)	natural_ change_ onlooker (B)	employee d_ bee (C)	natural_ change_ employed (D)	If A*B > C*D	Scout	beed	Optimal Result After re-analyze (mJ)																																																																																																																				
1	3	1	10.6454	4.1745	0.7513	45.6565	0.2551	False	1	1.2158	6.3487																																																																																																																				
		2	4.1745									2	3	1	13.1267	13.1267	0.5060	140.2212	0.6991	False	1	6.5274	9.5496	2	18.9301	3	1	2	39.6217	34.5972	0.8909	213.6683	0.9593	False	1	7.5716	8.9607	3	34.5972	4	2	1	33.5842	38.3812	0.5472	44.8201	0.1386	False	1	5.6881	44.6576	3	38.3812	5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863	2	8.8844	6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727	3	35.1709	7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567
2	3	1	13.1267	13.1267	0.5060	140.2212	0.6991	False	1	6.5274	9.5496																																																																																																																				
		2	18.9301									3	1	2	39.6217	34.5972	0.8909	213.6683	0.9593	False	1	7.5716	8.9607	3	34.5972	4	2	1	33.5842	38.3812	0.5472	44.8201	0.1386	False	1	5.6881	44.6576	3	38.3812	5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863	2	8.8844	6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727	3	35.1709	7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800				
3	1	2	39.6217	34.5972	0.8909	213.6683	0.9593	False	1	7.5716	8.9607																																																																																																																				
		3	34.5972									4	2	1	33.5842	38.3812	0.5472	44.8201	0.1386	False	1	5.6881	44.6576	3	38.3812	5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863	2	8.8844	6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727	3	35.1709	7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																		
4	2	1	33.5842	38.3812	0.5472	44.8201	0.1386	False	1	5.6881	44.6576																																																																																																																				
		3	38.3812									5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863	2	8.8844	6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727	3	35.1709	7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																
5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863																																																																																																																				
		2	8.8844									6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727	3	35.1709	7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																														
6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727																																																																																																																				
		3	35.1709									7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467	3	3.4058	8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																																												
7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467																																																																																																																				
		3	3.4058									8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348	3	5.8609	9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																																																										
8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348																																																																																																																				
		3	5.8609									9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642	3	11.1667	10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																																																																								
9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642																																																																																																																				
		3	11.1667									10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852	3	1.5800																																																																																																						
10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852																																																																																																																				
		3	1.5800																																																																																																																												

4.4 Simulation Results and Discussion

The ABC technique is written in MATLAB and runs on a 3.00 GHz Intel Core i3 processor with 4 GB of RAM. The quantity of energy consumed is measured in megaJoules (mJ). **Table 4.7** highlighted algorithms used in the allocation in distributed systems and found that network cost is not used to compute the system total cost. **Table 4.8** compares the planned work to other methods. Previously suggested methodologies failed to account for network costs in their computations, resulting in anomalies that increased overall system costs. In the proposed work, all parameters, including communication cost, computation cost, network cost, and bee degradation are used to produce the best least cost.

Table 4.7 Comparative Analysis of Techniques with their Cost Usability

References	Algorithms	Communication Cost	Computation Cost	Network Cost
[113]	List based HEFT (Heterogeneous Earliest Finish Time)	✓	✓	✗
[119]	Communication Link Sum (CLS)	✓	✓	✗
[64]	Enhanced PSO	✓	✓	✗
[117]	Communication and Computation Aware task scheduler framework	✓	✓	✗
[130]	Simplified biogeography-based optimization (SBBO) & Genetic algorithm	✓	✓	✗
[131]	Computational Algorithm	✓	✓	✗
[132]	Fuzzy C-means Clustering Technique	✓	✓	✗
Proposed Work	Artificial Bee Colony (ABC)	✓	✓	✓

Table 4.8 Comparative Studies of Proposed and Existing Approaches

Existing Approaches	References	Purpose	Assumption	Drawbacks
Artificial Intelligence	[113]	This used list-based (Heterogeneous Earliest Finish time) algorithm to reduce cost by minimizing energy consumption rate.	Assume to reduce system cost	Network cost is not used during the computation of cost.
Communication Link Sum (CLS)	[119]	To reduce the inter-processor communication to reduce the system cost for task allocation in distributed computing systems.	Assume to reduce system cost	Network cost is not used during the computation of system cost. On the other side, this approach follows a static task allocation policy.
Enhanced PSO	[64]	Proposed a Load Balancing Mutation Particle Swarm Optimization (LBMPZO) to allocate the best resources to tasks for maintaining execution time, transmission cost, makespan, etc.	Assume to improve efficiency by allocating data with all resources at a low cost.	network cost parameter is avoided here in the data allocation perspective
Proposed Approach	--	Work to reduce overall system cost using artificial bee colony approach for DAG	Assume to reduce system cost by learning, adaptive, and updating behavior to achieve performance in distributed computing.	does not consider the fault tolerance part to adjust the load.

All activities are carried out in parallel on processing engines, improving performance and boosting distributed job allocation. Low-cost processing units can complete jobs in huge numbers and in a short amount of time. The overall incurred cost on task execution is used to calculate system cost, as shown in **Table 4.9**. As stated in **Table 4.10**, the proposed work is compared to other current strategies, and it is determined that the ABC algorithm assists in obtaining optimal system cost to develop a robust distributed environment.

Table 4.9 Optimal Task Allocation

Optimal Allocation		Total Execution Cost	System Cost
Tasks	Processing Engines		
t ₃ , t ₆ , t ₇ , t ₁₀	PE1	191	
t ₄ , t ₈ , t ₉	PE2	140	396
t ₁ , t ₂ , t ₅	PE3	65	

Table 4.10 Evaluation of System Cost with Existing Methods

Proposed Algorithm		System Cost	A.Y., Hamed Algorithm [120]		System Cost	P.K., Yadav Algorithm [119]		System Cost
Tasks	Processing Engines		Tasks	Processing Engines		Tasks	Processing Engines	
t ₃ , t ₆ , t ₇ , t ₁₀	PE1		t ₄ , t ₇	PE1		t ₅ , t ₇	PE1	
t ₄ , t ₈ , t ₉	PE2	396	t ₂ , t ₃ , t ₈ , t ₉	PE2	459	t ₂ , t ₃ , t ₈ , t ₉	PE2	528
t ₁ , t ₂ , t ₅	PE3		t ₁ , t ₅ , t ₆	PE3		t ₁ , t ₄ , t ₆	PE3	

Table 4.11 compares the findings of total energy usage during task execution before and after. It demonstrates that the proposed ABC algorithm reduces total cost. **Figure 4.3** depicts the simulation results before and after optimization graphically, as stated below.

Table 4.11. Before and After Total Cost Results in mJ

Query/Tasks	Total Cost Before Optimization in mJ	Total Cost After Optimization in mJ
1	6.5224	6.3487
2	10.0158	9.5496
3	9.2899	8.9607
4	44.8201	44.6576
5	7.1796	6.9863
6	25.7880	25.5727
7	1.9467	1.9467
8	12.4964	12.0348
9	13.5141	13.1642
10	3.9973	3.5852
Average Cost of Tasks	13.56	13.28

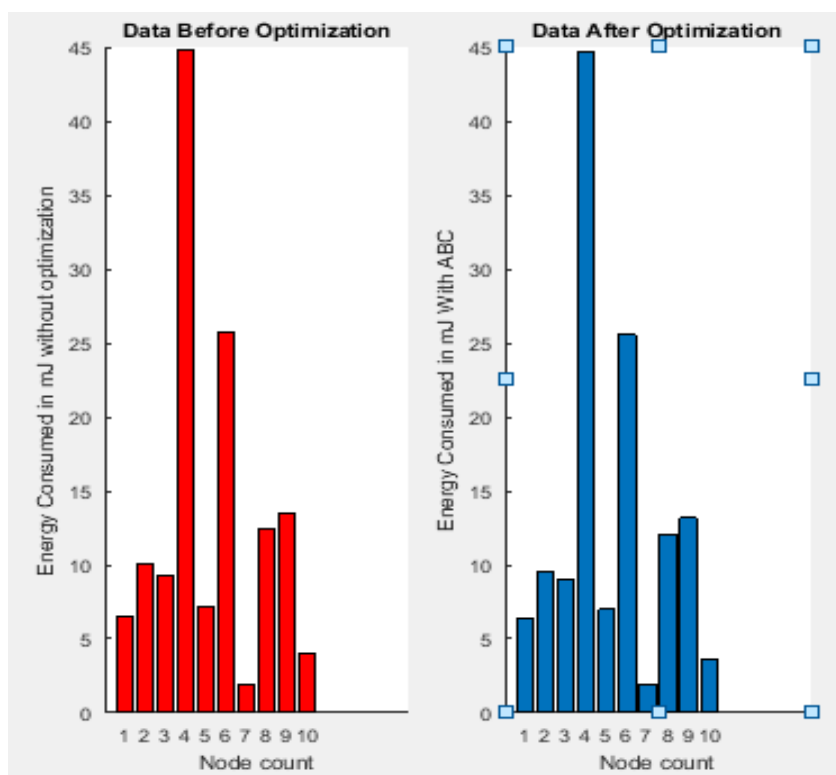


Figure 4.3 Results Before and After Optimization of Cost Using ABC

Table 4.12 compares the findings to a list-based work scheduling technique using artificial intelligence [113] and a task allocation model for system cost analysis using communication link sum (CLS) [119]. Except for network cost, these strategies take into account all indicators. As illustrated in **Figure 4.4**, the suggested approach saves 13.28 in overall expenses when compared to previous techniques. This approach makes it simple to allocate big data chunks and execute them quickly and cheaply. There are no long waits, delays, or completion periods with this strategy, which lowers the distributed system's performance.

Table 4.12 Comparison of Methods Based on Reduced Total Cost

Research Technique Used	Reduced Total Cost (%age)
Proposed Work using Artificial Bee Colony (ABC)	13.28
Existing Work using AI [113]	60.6
Existing Work using Communication Link Sum (CLS) [119]	24

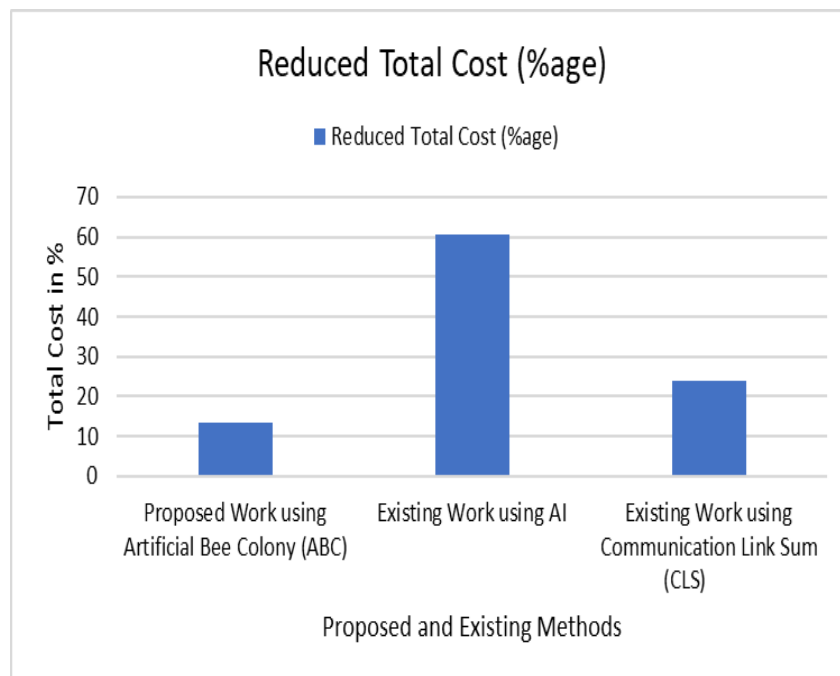


Figure 4.4 Reduced Total Cost in % age

4.5 Conclusion

In this paper, we describe a swarm intelligence-based artificial bee colony (ABC) method for lowering system execution costs and improving data distribution in distributed systems. By removing equivalent cost units from the entire cost, it is also easy to trace the degrading loss of bees. The ABC method was found to significantly reduce total execution costs and enhance system efficiency when compared to earlier approaches. According to prior studies, network costs are not used to assess system costs. As a result, test findings based on previous results are incorrect. This cost allocation model accounts for all costs spent in a distributed system's data processing.

Improve the Existing Data Deduplication Scheme with Machine Learning Architecture

Data as a resource has its own value in a distributed architecture, however, it appears that continuous integration of vast volumes of data across several sites without cross-verification to retain a single instance data pattern is unachievable. As a result, systems have run across roadblocks that have a direct impact on distributed workforce efficiency and performance. Users require high-quality data or information to continue operating as enhanced data services and to identify future trends. Duplicate data entries in storage repositories, on the other hand, are seen as a severe fault or stumbling block in data analysis and query activities. As a result, organizations have spent a lot of time and money detecting duplicate data throughout the duplicate entry detection process.

5.1. Introduction

This research work on a cutting-edge machine learning framework for detecting duplicate data in existing and new data entries. This technique imports textual data inputs or queries into memory to pre-process them, then converts them to a vector space model. A clustering K-means technique is used to group data into groups of equivalent capacity. Similarity computations were done cluster-by-cluster rather than on a large dataset to save time and money during the discovery phase.

The distributed environment satisfies company needs by allowing individuals to store, transfer, and secure their data on a regular basis, lowering operational expenses. Researchers have worked on [67-69] in the domain of distributed data. As a result of businesses' efforts to boost storage capacity at dispersed sites, data quality issues have been observed. Workers are accountable for developing data-related difficulties when a growing focus on data consumption

includes digital social media content, transactional data, archived data, and frequent data backups, all of which have developed as data problems. It reduces the efficiency of dispersed systems by increasing data storage capacity (due to redundancy). It lowers data quality by slowing query responses, increasing computation expenses, wasting excess bandwidth during data transmission, and slowing query responses. Such bottlenecks diminish distributed workforce efficiency and put data analysts and data engines at risk of not being able to extract data from dispersed massive data volumes in order to complete their responsibilities. It is crucial to identify duplicate data before disseminating data to dispersed sites. The deduplication process is pictorially shown in **Figure 5.1**.

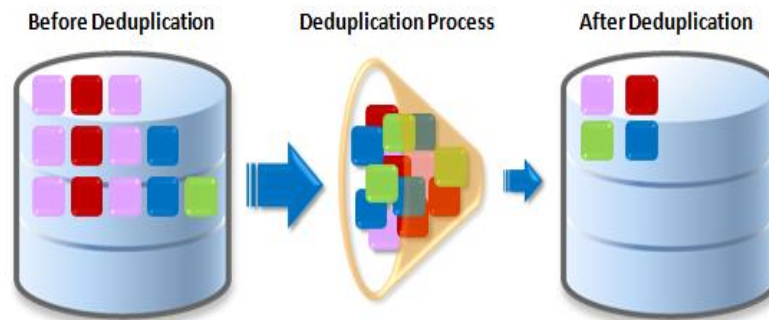


Figure 5.1 Data Deduplication Process

For distributed data structures, the researchers have presented a number of data deduplication strategies. Several strategies, such as attribute-based deduplication, attribute, and role-based deduplication, ANN, clustering algorithms, hash indexing, MapReduce and HDFS, bucket-based deduplication, Fingerprint clustering, Sampling method, Token generation techniques, and block-level deduplication, have been proposed to address existing challenges. Data uniqueness aids in the creation of error-free, consistent, and unambiguous data. Data redundancy is created by overlapping data from several sources, which provides an erroneous scenario for those who value performance attributes like data reliability, precision, and consistency. As a result, before being allowed to write to current data storage, frequent data

requests must be reviewed first. It only accepts non-conflict data that would be considered duplicates otherwise.

Businesses spend a large amount of money securing their data by performing periodic data backups and storing it on offsite servers, according to the findings. It causes data challenges such as increased data duplication ratios, delays in finding and retrieving data, muddled data issues, and other hidden costs. The process of recognizing duplicate data in storage units is known as data deduplication. When uploading and transferring data across a distributed network, the proposed approach detects duplicate data copies to save time and money.

This study proposes an advanced machine-learning framework for the deduplication process. The experiment uses the "user's sentimental analysis data" and "tweets" datasets, each comprising 1048576 data instances. With 2000x20 and 2001x30 records/instances, respectively, columns 6 and 4 are used as rawdata or inputs in the deduplication process, which are then used for text pre-processing or refinement by deleting stop words to get filteredwords. Each word in the filterdata is converted to a vector space model and given a value based on the sum of each character's ASCII values. Data clustering is a method of grouping data into groups in order to reduce computation costs and find duplicate data more effectively.

5.2 Proposed Methodology

MATLAB 2016a is used to accomplish the recommended methodology. **Figure 5.2** shows a design for detecting/identifying duplicate data.

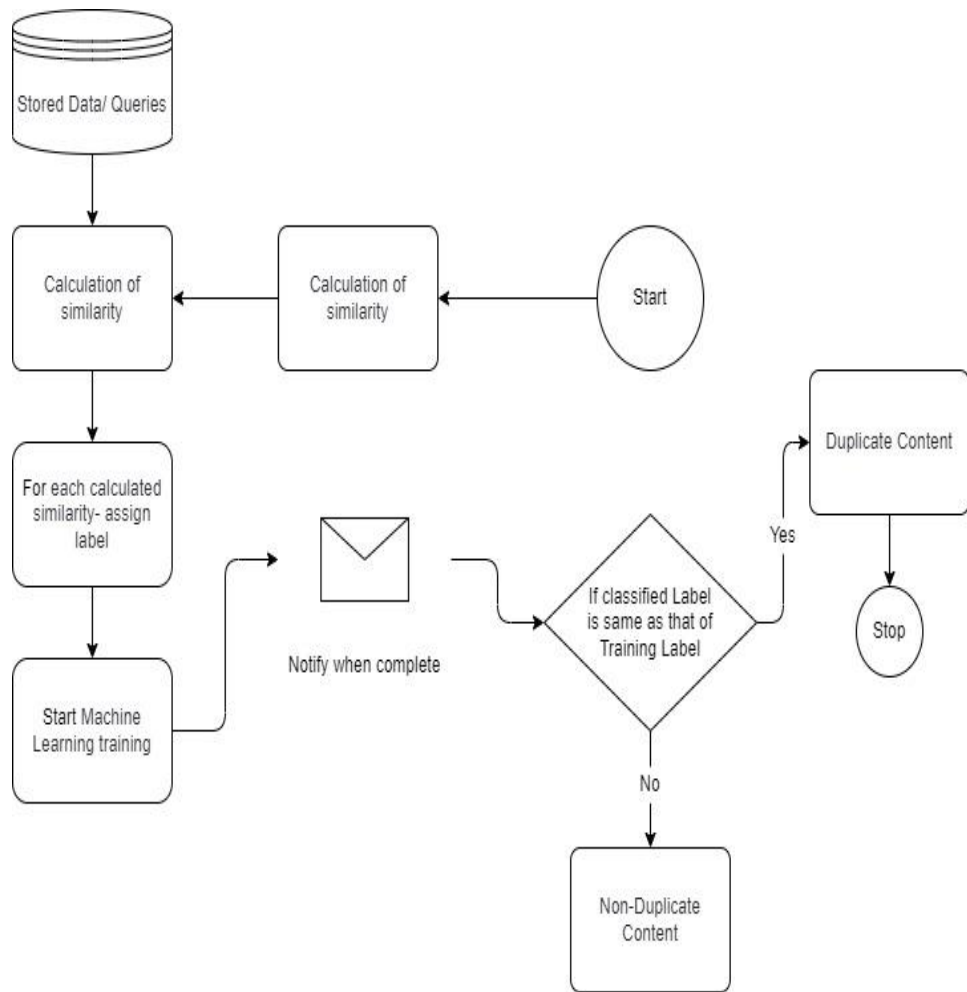


Figure 5.2 Work Flow of Proposed Methodology

5.2.1 Pre-Processing of Data

It refers to the practice of removing common terms from a dataset in a textual context utilizing stop-word lists that include articles, prepositions, and nouns, among other things. By removing stop words from the available dataset text records, the dimension of the dataset text records must be reduced. This is due to the fact that it contains no useful information. The stop list in [103] can be used to find these stop words. The sixth column from the dataset in [102] and the fourth column from the dataset in [104] are used as data inputs (rawdata) to the code lines below to apply to pre-process:

1. *Datacol = 6;*
2. *for i=1:r*
3. *rawdata{i,1}=data{i,datacol};*
4. *end*

At each cycle, data-input requests (rawdata) are selected row-by-row from the storage and set as currentdata, which is then parsed by the *getwords()* method. Individual words from currentdata are obtained using the *getwords()* method and stored in the array list of *allwords*. These words are compared to stop word lists to generate a *filteredwords* list. After the pre-processing procedure, *filteredwords* containing suppressed words are retrieved.

5.2.1.1. Pseudocode for Pre-Processing Task

Input: *rawdata*

Output: *filteredwords*

1. *Load rawdata as input % Select text column row-wise*
2. *[r,c]= size(rawdata);*
3. *for i=1:r*
4. *currentdata=rawdata{i,1};*
5. *words=getwords(currentdata);*
6. *allwords{i}=words;*
7. *filteredwords=filterdata(words,stopwords);*
8. *End*

5.2.2 Word-To-Vector (W2V) Conversion

Comparing how similar or similar two objects are is a typical way to gauge similarity. Using vectors as a method of calculating similarity is one option. A vector is a number that has both a magnitude and a direction. It takes a big text corpus (tweets with stop words removed) as input and assigns a vector to each word. Depending on the size of the characteristics or *filteredwords*, such vectors are expressed in rows and columns, which are then utilized to cluster the data. Each characteristic or *filteredwords* is replaced by a value, which is the total of each character's ASCII value.

5.2.2.1. Pseudocode for W2V Translation

Input: *filteredwords*

Output: *w2v*

1. *for wd=1:numel(filteredwords)*
2. *w2v(i,wd)=sum(double(filteredwords{wd}));*
3. *End*

5.2.3 Clustering of Data

Textual data samples are turned into vector data models for deduplication, which is then used to create clusters. The data instances in these datasets are used to train machine learning models. As indicated in **Table 5.1**, datasets are divided into three clusters, each with its unique collection of data instances.

Table 5.1 Cluster-Wise Distribution of Datasets

Dataset	Dataset Size Used	Clusters Indexes	Data-Instances	
			Rows	Cols.
		1	724	20
Sentiment Analysis with Tweets [102]	2000x20	2	701	20
		3	575	20
Tweets-about-the-top-companies-from-2015-to-2020 [104]	2001x30	1	35	30
		2	1935	30
		3	31	30

Clustering is a machine learning technique that aids in the discovery of groups in unlabelled data. It aids in the examination and categorization of fresh data patterns into their related groupings. During the deduplication process, it preserves intra-cluster similarity and assists in identifying duplicate data contents at the cluster level. Rather than applying to the huge dataset values, group-data analysis improves similarity measure outcomes. The computing cost of data analysis interpretation is also reduced when done in groups.

To begin the process of creating data clusters, the dataset is divided into indexes and centroids. Based on a vector data model, K-means uses the closest neighbouring technique to allocate data to cluster categories and estimate centroid positions for each cluster. During the cluster identification procedure, these centroid coordinates aid in the categorization of vector data. As a result, rather than comparing dataset values, the deduplication procedure is applied directly at the cluster level.

5.2.4 Training and Testing Indexes Generation

The dataset for evaluating machine learning models is divided into training and testing data. For training and testing data, a ratio of 80 percent to 20 percent is employed. In order to create the training index (index) and testing index (ti), a randomly generated integer is chosen in each iteration until the training and testing counts are not equal. These indexes are saved in the training indexes and test index lists, which are then utilized to extract data rows. These indexes are also used to calculate similarity based on the vector values of data rows. Equations I and (ii) are used to estimate training count and testing count, respectively:

$$training_count = round(training_rate/100*rrc) \quad (i)$$

$$test_count = rrc - numel(training_indexes) \quad (ii)$$

5.2.4.1. Pseudocode for Generating Training and Testing Indexes

Input: training_count, rrc, test_count

Output: training_indexes, test_index

1. *for kki=1: training_count*
2. *tindex=round(rrc*rand);*
3. *if tindex==0*
4. *tindex=1;*
5. *end*
6. *training_indexes(kki)=tindex; % generate training index*
7. *end*
8. *for kkm=1:test_count*
9. *ti=round(rrc*rand);*

10. *if* $ti==0$
11. $ti=1$;
12. *end*
13. $test_index(kkm)=ti$; *%generating testing indexes*
14. *end*

5.2.5 Similarity Calculation Cluster-Wise

A similarity computation is done independently on each cluster to detect duplicate data records. Calculating the similarity index for each data point in the cluster takes a lengthy time. To save time, we construct cluster indexes rather than individual indexes. Cluster indexes are built using a random evaluation approach to save processing time. Ten randomly applied indices are chosen independently to describe each clustered index. These validations are a set of data row values picked at random and used to determine cosine similarity to the test data. The previous result of the similarity value is maintained and added to the most current result during the computation.

This technique is repeated based on the characterization set-value (which represents row attributes) and the validation set-value (which represents row attributes) (which represents random data values). The cluster-wise similarity is used to identify duplicate data received as testing data or new data entries to be stored as a whole in data storage. As a result, all data contents that fall within the range of the cluster similarity index could be regarded duplicate material. The results of cluster-wise similarity values are provided in **Table 5.2**.

5.2.5.1 Pseudocode For Similarity Calculation Cluster-Wise

Input: rowvalue, test_index

Output: simvalue=calculatecossim(v1,v2)

1. $dfa=[]$;
2. *for* $i=1:3$
3. $dtcluster=[]$;
4. $dtcluster=w2v1(ind1==i, :)$;
5. $smvalue=0$;
6. $[rcl,cls]=size(dtcluster)$;

```

7. total_no_of_validations=10;
8. total_characterization=10;
9. counter=0;
10. for dd=1:total_characterization
11. for kd=1:total_no_of_validations
12. rowvalue=round(rcl*rand);
13. if rowvalue==0
14. rowvalue=1;
15. end
16. counter=counter+1;
17. dtver(counter,:)=dtcluster(rowvalue,:);
18. end
19. end
20. datatovaildate=w2v1(test_index,:);
21. try
22. smvalue=smvalue+calculatecossim(dtver(1:numel(test_index,:),datatovaildate);
23. catch
24. smvalue = smvalue + calculatecossim(datatovaildate, dtver);
25. end
26. dfa(i)=smvalue/numel(test_index);    % clusters wise similarity index
27. end

```

Table 5.2 Cluster-Wise Similarity Calculations

Datasets References	Cluster Indexes	Similarity Value (Cluster-wise)
	1	0.0012
[102]	2	0.0009
	3	0.0014
	1	0.2222
[104]	2	0.2077
	3	0.2579

5.2.5.2. Pseudocode to Calculate Cosine-Similarity

Input: test_index, row_value

Output: simvalue = simvalue + calculatecossim(v1,v2)

```

1. nume=0;           %numerator
2. deno=0;           %denominator
3. for i=1:numel(v1)
4.   try
5.     nume=nume+v1(i)*v2(i);
6.   catch
7.   end
8. end
9. deno1=0;
10. deno2=0;
11. for j=1:numel(v1)
12.   try
13.     deno1=deno1+v1(j)^2;
14.     deno2=deno2+v2(j)^2;
15.   catch
16.   end
17. end
18. deno=sqrt(deno1)*sqrt(deno2);
19. simvalue=nume/deno;
20. end

```

5.2.6 Applying of Ground Truth Test (gt)

The *gt* test command is used to detect duplicate data on the testing data. To do so, 1x400 and 1x1600 data-size test and training indexes are employed. Records are detected as duplicate if vectors on training indexes are matched and determined to be identical to vectors on test indexes, otherwise, they are not. The *gt* test elements list is updated to 1 (if a match is discovered) or 0 (if no match is found) during the comparison (for matching not found). This operation is repeated until all elements on the training data are not compared to the test data as a whole, and the *gt* test element list is populated.

5.2.6.1. Pseudocode To Apply Ground Truth Test

Input: training_indexes, test_index

Output: gt_test

1. $w2v_test=[];$
2. $w2v_test=w2v1(test_index,:);$
3. $[rtest,ctest]=size(w2v_test)$
4. $gt_test=[];$
5. $gt_test=zeros(1,numel(test_index));$
6. *for* $kdi=1:numel(gt_test)$
7. $ssd=[];$
8. $ssd=find(training_indexes==test_index(kdi));$
9. *if* $\sim isempty(ssd)$
10. $gt_test(kdi)=1;$
11. *end*
12. *end*

5.2.7 Similarity Calculation on Testing Data

The test data and test value are subjected to a similarity test in this step. Test values are 400x20 vector values of testing data that are based on randomly generated test indices. In this section, the set value for test indexes is shown at line 5 in pseudocode. To reduce computing expense, each test value of testing data is compared with randomly produced test data to compute a cosine similarity test. The similarity test is calculated by multiplying test indexes by the sum of previously computed *simtest* values.

5.2.7.1. Pseudocode For Testing Data Similarity Calculation

Input: test_data, test_value

Output: sim_test

1. $simtest=0;$
2. $[rtest,ctest]=size(w2v_test)$
3. *for* $ii=1:rtest$
4. $test_value=w2v_test(ii,:);$

```

5. test_indexes=5;           % generations of test index
6. for kk=1:test_indexes
7. test_index=round(rtest*rand);
8. if test_index==0
9. test_index=1;
10. end
11. test_data=w2v(test_index,:);
12. simtest=simtest+calculatecossim(test_data,test_value);
13. end
14. simtest=simtest/test_indexes;

```

5.2.8 Detecting Duplication by Comparing Similarity-Test Results with Clusters-Similarity

Threshold limitations are determined by adding and subtracting 10% from cluster similarity values after the similarity test has been computed. Setting up threshold as described in Pseudocode lines 4-5 defines the threshold range limit. The decision boundary (threshold limit) is compared to the similarity test (*simtest*) to validate if the data is duplicated (1) or not. Each testing data test value (400x1) is compared to threshold limits comparable to cluster similarity index numbers. If the results of a similarity test are close to or within threshold limits, the input is classified as duplicate data, and the result set list is updated with duplication (1) for each comparison performed. Because duplicate data inputs may be limited or monitored, data is sometimes not allowed to be stored in current storage.

5.2.8.1. Pseudocode for comparison of Similarity test and Cluster Similarity

Input: dfa, simtest, duplicate

Output: result_test

```

1. result_test=[];
2. duplicate=0;
3. for kd=1: numel(dfa)
4. th1=dfa(kd)+dfa(kd)*.10;
5. th2=dfa(kd)-dfa(kd)*.10;

```

```

6. if simtest<th1 && simtest>th2
7. duplicate=1;
8. else
9. end
10. end
11. result_test(ii)=duplicate;
12. end

```

5.3 Performance Parameters Estimation

The sensitivity, specificity, fmeasure, and accuracy metrics are used to test the performance of the machine learning model. When the results of the resulting test and the ground truth test (gt test) are almost the same, the accuracy percentage rate rises. For each result test and gt test element, frequencies of 1 and 0 are computed independently. According to coding lines 24 and 25, the percentages of sensitivity (for frequency 1) and specificity (for frequency 0) are calculated.

5.3.1. Pseudocode to Compute Performance Parameters

Input: result_test, gt_test

Output: function [sensitivity, Specificity, fmeasure] = parameters (output, actual)

```

1. ar=actual;
2. actual=[];
3. actual=ar;
4. fone=0;           % one in test result
5. ft=0;           % one in test gt
6. for i=1:numel(actual)
7. if actual(i)==1
8. ft=ft+1;
9. if output(i)==1
10. fone=fone+1;
11. end
12. end
13. end
14. fzero=0;       % zero in test result

```

```

15. fz=0;           % zero in test gt
16. for i=1:numel(actual)
17. if actual(i)==0
18. fz=fz+1;
19. if output(i)==0
20. fzero=fzero+1;
21. end
22. end
23. end
24. sensitivity=(fone/ft)*100;
25. Specificity=(fzero/fz)*100;
26. acflag=zeros(1,numel(output));
27. for kk=1:numel(output)
28. if output(kk)==actual(kk)
29. acflag(kk)=1;
30. end
31. end
32. match=0;
33. for k=1:numel(output)
34. if output(k)==actual(k)
35. match=match+1;
36. end
37. end
38. tempv=sensitivity;
39. sensitivity=Specificity;
40. Specificity=tempv;
41. accuracy=((((numel(find(acflag==1))/numel(acflag)))+((numel(find(acflag=
    =0))/numel(acflag))))/2)*100;
42. accuracy=(match/numel(output))*100;
43. fmeasure=2*sensitivity*Specificity/(sensitivity+Specificity);
44. fmeasure=fmeasure/100;
45. end

```

As indicated in **Table 5.3** and evaluated at code lines 24-44 in the above-concerned pseudocode, four elements are examined for the assessment of the proposed deduplication approach: sensitivity, specificity, accuracy, and

fmeasure. The sensitivity and specificity values are calculated using data from *fone* and *fzero*. The predictions of duplicate records as duplicate and *fzero*, which mistakenly suggest a duplicate record, distinguish *fone*. The fraction of correct test data predictions is defined as accuracy. The total number of forecasts can easily be used to calculate the number of correct predictions. Fmeasure displays the precision of a model on a given dataset.

5.4 Results and Analysis

The performance of the suggested deduplication approach is evaluated using a variety of parameters. The approaches are implemented in MATLAB 2016a with an Intel Core 2 Processor, 2.1MHz clock speed, and 4GB of RAM.

5.4.1 Evaluation of Performance

Based on the datasets in [95],[97], this section gives a performance analysis of the proposed deduplication method. The performance of the proposed machine learning architecture is examined on both datasets, and it is discovered that it is effective in achieving the accuracy performance criterion. The evaluation parameters used for both datasets are specificity, sensitivity, accuracy, and fmeasure, as shown in **Figure 5.3**. **Table 5.3** shows the results of parameter measurements used to estimate performance specifications. The proposed sophisticated machine learning architecture achieves an average accuracy of 99.7%.

Table 5.3 Performance Parameters

Dataset Used	Sensitivity	Specificity	Fmeasure	Accuracy
Sentiment Analysis with Tweets [102]	1	0.004545	0.00904	100
Tweets about the top companies from 2015-to-2020 [104]	0.99408	0	0	99.4083
Average Accuracy				99.7

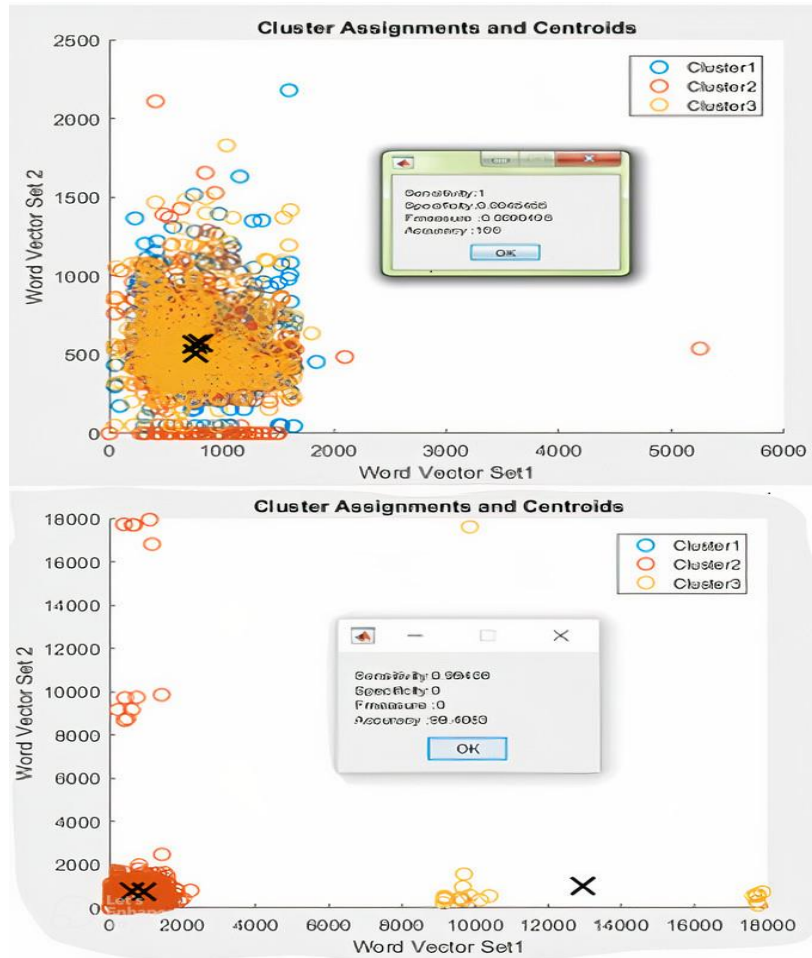


Figure 5.3 Graphical Representation of clusters and centroids with performance parameters

5.5 Comparative Study

The dependency on storage services like Dropbox [121], OneDrive, Google Drive [122], Mozy [123], and Spideroak [124] are used to maintain voluminous data. Researchers used a variety of strategies to ensure a high deduplication ratio to reduce storage and associated cost using the deduplication process. The deduplication techniques continuously work for achieving performance in a distributed environment and work for different parameters. Some of them are depicted below in **Table 5.4** as applied by different algorithms.

Table 5.4 Comparative Analysis of performance parameters in data deduplication

Deduplication Algorithms	Performance Parameters					
	Storage	Communication	Computation	Reliability	Privacy	Security
[27-30]	✓	✗	✗	✗	✗	✗
[133]	✓	✓	✓	✓	✓	✓
[134]	✓	✗	✗	✓	✗	✗
[135]	✓	✗	✗	✓	✗	✓
Proposed work	✓	✓	✓	✓	✗	✗

In this proposed scheme, an advanced machine learning architecture is applied, and a comparison with existing Support Vector Machine, Neural Network, and Fuzzy techniques is undertaken, and it is discovered that our scheme has a high accuracy rate of 100% (approx..). To depict metrics findings, the performance study of the deduplication process is done out on the accuracy parameter, as shown in **Table 5.5** and bar-graph in **Figure 5.4**. The comparison is made between the planned work and the existing methodology in terms of accuracy.

Table 5.5 Results of Proposed and Existing Work

Proposed and Existing Data Deduplication Schemes	Accuracy %age
Advance Machine Learning Architecture (Proposed Model)	99.7
FEM (Fuzzy Expectation Maximization) Clustering [125]	97.98
Enhanced Fuzzy Ontology Based Record Deduplication [126]	91.5
Support Vector Model [127]	88
Artificial Neural Network Model [128]	79.8

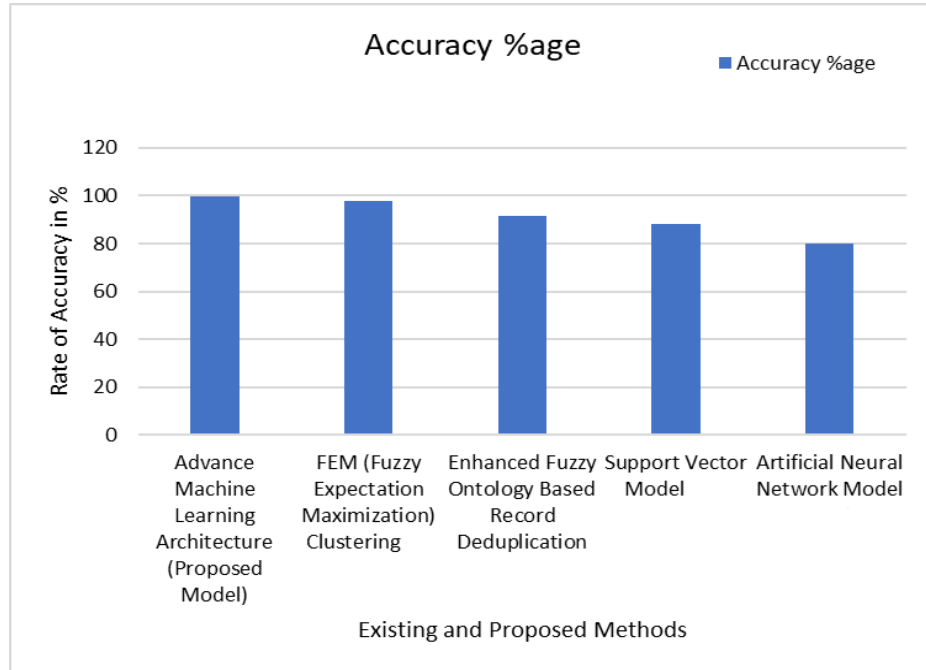


Figure 5.4 Comparison of Proposed and Existing Techniques

5.6 Conclusion

This research proposes a sophisticated machine learning architecture for detecting duplicate data on distributed storage as well as monitoring fresh data inputs. It emphasizes the importance of using an effective deduplication technique to detect duplicate data entries. We suggested an advanced approach based on a machine learning model that can be used to detect duplicate data entries and improve labor efficiency and system performance. It uses an unsupervised learning approach to detect duplicate data in a textual environment using a relative approach. In comparison to existing SVM, NN, and Fuzzy model accuracy results, the proposed methodology has obtained 100% accuracy. On the basis of the used datasets, evaluations of performance characteristics like sensitivity, specificity, fmeasure, and accuracy are also generated. Performance is assessed in this case using two distinct datasets. It is discovered that experiments using the proposed methods produce better outcomes than those using the existing models.

Chapter 6

Conclusion, Contributions and Applications, and Future Work

The data storage maintains data from varied sources including database users, web applications, web blogs, forums containing users' feedback, and messaging apps including WhatsApp, Facebook, Twitter, and Instagram. Data size is scaled up and affects the storage capacity. It was found that researchers contributed very little to handle the data storage part. This research effort is an initiative towards handling textual data, that enables users to retrieve information easily and control storage by appropriate data distribution techniques at geographical sites. This can be achieved by distributing the data at the repository or storage level into manageable pieces called fragments. Another aspect of a distributed system is to reduce total costs during data access operations. This can be fulfilled by applying all cost parameters during the calculation of the total cost during data allocation measures. Besides, it raised high chances of the occurrence of repeated data problems arising due to the availability of multiple data instances at sites. This can be resolved by identifying multiple data entries at distributed data stores and controlling during the arrival of a new entry at the beginning.

The conclusion and future perspectives of the projected study are described in detail further in this study. The conclusion is presented in section 6.1, contributions and application in section 6.2, and the planned work for the future is presented in section 6.3.

6.1 Conclusion

This research work explores a novel scheme for data fragmentation, allocation, and deduplication to cope with various distributed systems data hurdles and performance issues. This work introduces three major aspects of a distributed systems design concerning textual data or information: a) Fragmentation of data for controlling data storage. b) Cost-based data allocation for distributed performance. c) Deduplication of data for handling redundant data instances.

Some research gaps were found in earlier techniques or methodologies adopted by researchers and motivated us to eliminate them to make distributed system design robust.

In this work, a novel relative data fragmentation architecture is proposed for partitioning a large dataset into smaller fragments. This experiment makes use of Twitter data, which has been converted into vectors for fragmentation purposes. Cosine, soft cosine, and hybrid similarity are calculated, as well as centroid placements. The K-Mean approach is used to estimate the distance between data points and each centroid to locate clusters. Finally, ANN is used to assess the validity and performance of the model. The purpose of this research is to introduce a novel data fragmentation architecture based on similarity into an unsupervised learning environment. The comparison revealed remarkable accuracy and recall when compared to other proposed alternatives. It aids in the accurate and precise categorization of fragmented data, as well as indicating maximum coverage and accuracy while reducing total processing time. Due to a lack of balance in the quality and efficiency of clustering in categorical data sets, earlier strategies that relied on cosine-based k-means, cosine, and soft cosine hybridization for clustering were ineffective. Existing techniques' principles are good in certain cases, but not in others. As a result, algorithm hybridization is the best option. To provide enhanced efficiency and the ability to cope with big data sets, cosine and soft cosine similarity notions are employed to calculate hybrid similarity in this study effort.

Secondly, we use a swarm intelligence-based artificial bee colony (ABC) approach to reduce system execution costs and enhance data allocation in distributed systems. It's also simple to measure bee degradation losses when comparable cost units are removed from the total cost. When compared to previous methodologies, the ABC algorithm was proven to considerably lower total execution costs and increase system efficiency. According to previous studies, network prices are not taken into account when calculating system costs. As a result, earlier test results are untrustworthy. This cost allocation

model accounts for all expenses incurred during data processing in a distributed system.

This research proposes a sophisticated machine learning architecture for detecting duplicate data on distributed storage as well as monitoring fresh data inputs. It emphasizes the importance of using an effective deduplication technique to detect duplicate data entries. We suggested an advanced approach based on a machine learning model that can be used to detect duplicate data entries and improve worker efficiency and system throughput. It uses an unsupervised learning approach to detect duplicate data in a textual environment using a relative approach. In comparison to existing SVM, NN, and Fuzzy model accuracy results, the proposed methodology has obtained 100% accuracy.

All above schemes are implemented using MATLAB and performances are compared with existing techniques and observed that proposed techniques are optimal and achieve performance.

6.2 Contribution and Applications

The main objective of this research work is to introduce data fragmentation, allocation, and deduplication techniques for building a robust distributed system. To achieve this a comprehensive literature study is carried out that covers various fragmentation, allocation and deduplication strategies involved in the distribution design.

In today's world, everyone strives for a high level of consistency and quality in their data. On the one hand, the company is reliant on a centralized data management system for its desired data. This allows users to access it from within the organization's network rather than having to seek help from across the globe. On the other hand, to keep up with global business, companies must manage their data in a global context so that it can be accessed by anyone, anywhere, at any time.

Businesses make every effort to protect and organize their data to fulfill their transactions. These transactions may be used for data analysis, which can aid in recognizing current business trends, resolving data anomalies detected during data insertion, updating, and deletion, and computations on stored data. All of the following objectives would be met if all the relevant ideas were merged to improve existing design architecture and build new approaches.

This research work effectively handles growing data issues by controlling storage and partitioning them into fragments. Once data is fragmented, is allocated remotely based on the least total cost paradigm. Another hurdle as duplicate data in distributed sites is tackled using machine learning that sustains data quality and focuses on maintaining a single data instance at each site.

Hence, the research carried out in this thesis will help in the fields of software development, business data analytics, and other areas including database performance, medical, and data investigations.

6.3 Future Work

In this research work, efforts are made to provide novel data fragmentation, allocation, and deduplication strategies to build a reliable, high-performance distributed operational environment. When dealing with textual data, future work will focus on optimizing work results by introducing strategies to control ever-increasing data storage, reduction of data distribution costs, and enrich data quality by preserving a single data instance in a distributed architecture.

References

- [1] Andrews, Gregory R. Foundations of Multithreaded. Parallel and Distributed Programming. Addison– Wesley. 2000. p. 8–9 291. ISBN 978-0-201-35752-3.
- [2] Ghosh, Sukumar. Distributed Systems – An Algorithmic Approach. Chapman & Hall/CRC. 2007. p. 3. ISBN 978-1-58488-564-1.
- [3] Andrews, Gregory R. Foundations of Multithreaded. Parallel and Distributed Programming. Addison– Wesley. 2000. p. 291. ISBN 978-0-201-35752-3
- [4] Ghosh, Sukumar. Distributed Systems – An Algorithmic Approach. Chapman & Hall/CRC. 2007. p. 3. ISBN 978-1-58488-564-1
- [5] Lynch, Nancy A. Distributed Algorithms. Morgan Kaufmann. 1996. ISBN 978-1-55860-348-6.
- [6] Ranichandra, C., Tripathy, B.K. Architecture for distributed query processing using the RDF data in a cloud environment. Evol. Intel. 2019. <https://doi.org/10.1007/s12065-019-00315-5>
- [7] Weiss C, Karras P, Bernstein A. Hexastore: sextuple indexing for semantic web data management. Proc VLDB Endow. 2008. 1(1). p.1008–1019.
- [8] Das S, Agrawal D, El Abbadi A. G-store: a scalable data store for transactional multi key access in the cloud. In Proceedings of the 1st ACM symposium on cloud computing. ACM. 2010. p.163–174.
- [9] Lu X.. Symmetric Distributed Server Architecture for Network Management System. In: Zhou X.. Xu M., Jahnichen S., Cao J. (eds) Advanced Parallel Processing Technologies. APPT 2003. Lecture Notes in Computer Science. 2003. vol. 2834. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-540-39425-9_50

- [10] A. Valadares, C. V. Lopes. A Framework for Designing and Evaluating Distributed Real-Time Applications. 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real-Time Applications. Toulouse. 2014. p. 67-76. <https://doi.org/10.1109/DS-RT.2014.17>
- [11] K. Saxena, A. R. Abhyankar, Distributed architecture for self-organizing smart distribution systems, in IET Smart Grid. 2018. vol. 1. no. 4. p. 113-122. <https://doi.org/10.1049/iet-stg.2018.0029>
- [12] Zouari M., Segarra MT., André F., Thépaut A. An Architectural Model for Building Distributed Adaptation Systems. In: Brazier F.M.T., Nieuwenhuis K., Pavlin G., Warnier M., Badica C. (eds) Intelligent Distributed Computing V. Studies in Computational Intelligence. 2011. vol 382. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-642-24013-3_15
- [13] Yuan P, Liu P, Wu B, Jin H, Zhang W, Liu L. TripleBit: a fast and compact system for large-scale RDF data. Proc VLDB Endow. 2013. 6(7). p.517–528.
- [14] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref and A. Ghafoor. A Distributed Access Control Architecture for Cloud Computing. IEEE Software. 2012. March-April. vol. 29. no. 2. p. 36-44. <https://doi.org/10.1109/MS.2011.153>
- [15] Slimani Y., Najjar F., Mami N. An Adaptive Cost Model for Distributed Query Optimization on the Grid. In: Meersman R., Tari Z., Corsaro A. (eds) On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. OTM 2004. Lecture Notes in Computer Science. 2004. vol. 3292. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-540-30470-8_26
- [16] Trung A. D. Big Data: Lambda Architecture in a nutshell. gitconnected. 2020. Oct 9. <https://levelup.gitconnected.com/big-data-lambda-architecture-in-a-nutshell-fd5e04b12acc>

- [17] Joe N. What are microservices?. cloudacademy. 2019. April 10. <https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/>
- [18] Weiss C, Karras P, Bernstein A. Hexastore: sextuple indexing for semantic web data management. Proc VLDB Endow. 2008. 1(1). p.1008–1019.
- [19] Y. Jiao, W. Wang. Design and Implementation of Load Balancing of Distributed-system-based Web Server. 2010 Third International Symposium on Electronic Commerce and Security. Guangzhou. 2010. p. 337-342. <https://doi.org/10.1109/ISECS.2010.81>
- [20] K. Iwanicki. A Distributed Systems Perspective on Industrial IoT. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). Vienna. 2018. p. 1164-1170. <https://doi.org/10.1109/ICDCS.2018.00116>
- [21] H. Jiang, D. Liu, X. Chen, H. Liu, and H. Mei. How Are Design Patterns Concerned by Developers?. 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Montreal. QC. Canada. 2019. p. 232-233. <https://doi.org/10.1109/ICSE-Companion.2019.0009>
- [22] Peng Peng, Lei Zou, Lei Chen, Dongyan Zhao, “Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload”, IEEE Transactions on Knowledge and Data Engineering, Vol. 31, No. 4, pp. 670-685, April 2019
- [23] H. Abdalla, A. M. Artoli, “Towards an Efficient Data Fragmentation, Allocation, and Clustering Approach in a Distributed Environment”, Information, Vol 10, No.112, 2019
- [24] Verma and A. Kumar, “Performance Enhancement of K-Means Clustering Algorithms for High Dimensional Data sets”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, No. 1, pp.5-9, 2014

- [25] Z. Tao, H. Liu, H. Fu and Y. Fu, “Image Co-segmentation via Saliency-Guided Constrained Clustering with Cosine Similarity”, AAAI, pp. 4285-4291, 2017
- [26] Sewisy A, Amer A, Abdalla H (2017) A novel query-driven clustering-based technique for vertical fragmentation and allocation in distributed database systems. *Int J Semant Web Inf Syst* 13(2):27–54
- [27] X. Gu, H. Zhang and S. Kim, “Deep code search”, In Proceedings of the 40th International Conference on Software Engineering, ACM, pp. 933-944, 2018
- [28] W. L Xiang, Y. Z. Li, R. C. He, M.X. Gao, M.Q An, “A novel artificial bee colony algorithm based on the cosine similarity”, *Computers & Industrial Engineering*, Vol. 115, pp.54-68, 2018
- [29] Wiese, L. (2014). Clustering-based fragmentation and data replication for flexible query answering in distributed databases. *Journal of Cloud Computing* 3, 18. <https://doi.org/10.1186/s13677-014-0018-0>
- [30] Ali A. Amer, Adel A. Sewisy, Taha M.A. Elgendy. (2017). An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs). *Heliyon* 3 e00487. doi: 10.1016/j.heliyon.2017. e00487
- [31] Khan S. I., (2016). Efficient Partitioning of Large Databases without Query Statistics. *Database System Journal*, pp. 34-53
- [32] Masood Niazi Torshiz, Azadeh Salehi Esfaji, Haleh Amintoosi, Enhanced Schemes for Data Fragmentation, Allocation, and Replication in Distributed Database Systems. *International Journal of Computer Systems Science & Engineering*. 2020. 2. pp. 99-112
- [33] Rahimi, H., Parand, F. A., & Riahi, D. (2018). Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases. *Applied computing and informatics*, 14(2), pp. 127-133. <https://doi.org/10.1016/j.aci.2015.03.001>

- [34] Lim, S., Ng, Y. A Hybrid Fragmentation Approach for Distributed Deductive Database Systems. *Knowledge and Information Systems* 3, pp. 198–224, 2001. <https://doi.org/10.1007/PL00011666>
- [35] Shahidul Islam Khan, A. S. M. Latiful Hoque, “A New Technique for Database Fragmentation in Distributed Systems”, *International Journal of Computer Applications (0975 – 8887)*, Volume 5– No.9, August 2010
- [36] I. B. Oriji, I.C. Ejiofor, “A Hybrid Model for Data Fragmentation in Distributed System”, *International Research Journal of Computer Science*, Issue 04, Volume 5, pp. 186-192, 2018
- [37] Peng, P., Zou, L., Chen, L., & Zhao, D. Adaptive distributed RDF graph fragmentation and allocation based on query workload. *IEEE Transactions on Knowledge and Data Engineering*, 31(4), pp.670-685, 2019. <https://doi.org/10.1109/TKDE.2018.2841389>
- [38] Aloini, D., Benevento, E., Stefanini, A., & Zerbino, P. Process fragmentation and port performance: Merging SNA and text mining. *International Journal of Information Management*, 2020, 51,101925. <https://doi.org/10.1016/j.ijinfomgt.2019.03.012>
- [39] Memmi, G., Kapusta, K., & Qiu, H. Data protection: Combining fragmentation, encryption, and dispersion. In *2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)* (pp.1-9). IEEE. <https://doi.org/10.1109/SSIC.2015.7245680>
- [40] Raouf, A.E.A., N.L. Badr, and M. Tolba. An optimized scheme for vertical fragmentation, allocation and replication of a distributed database. in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*. 2015. IEEE.
- [41] Ahmed E. Abdel Raouf , Nagwa L. Badr, M. F. Tolba. An Enhanced CRUD for Vertical Fragmentation Allocation and Replication Over the Cloud Environment. *INFOS '16: Proceedings of the 10th International*

- Conference on Informatics and Systems. 2016. pp. 146-152.
<https://doi.org/10.1145/2908446.2908480>
- [42] Abdel Raouf A.E., Badr N.L., Tolba M.F. Distributed Database System (DSS) Design Over a Cloud Environment. In: Hassanien A., Mostafa Fouad M., Manaf A., Zamani M., Ahmad R., Kacprzyk J. (eds) *Multimedia Forensics and Security. Intelligent Systems Reference Library*, vol 115. 2017, Springer, Cham. https://doi.org/10.1007/978-3-319-44270-9_5
- [43] Hauglid, J.O., Ryeng, N.H., Nørnvåg, K.: DYFRAM: dynamic fragmentation and replica management in distributed database systems. *Distrib. Parallel Databases* **28**(2–3), 157–185 , 2010
- [44] Tosun U, Dokeroglu T, Cosar A, Heuristic algorithms for fragment allocation in a distributed database system. In: Gelenbe E, Ricardo L (eds) *27th international symposium on computer and information sciences (ISCIS). Computer and information sciences III*. Springer, pp 401–408, 2013
- [45] H. I. Abdalla, “A New Data Re-Allocation Model for Distributed Database System”, *International Journal of Database Theory and Application*, Vol. 5, No. 2, pp. 45-60, 2012
- [46] Singh, A., Kahlon, K. S., & Virk, R. S. Nonreplicated Static Data Allocation in Distributed Databases Using Biogeography-Based Optimization. *Chinese Journal of Engineering*, 2014, 1–9. doi:10.1155/2014/785321
- [47] N. K. Z. Lwin, T. M. Naing, “Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System”, *ICIIBMS*, Bangkok, Thailand, pp.300-305, 2018
- [48] H. I. Abdalla, “An Efficient Approach for Data Placement in Distributed Systems”, *Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering*, pp. 297-301, 2011

- [49] Guo, S.; Liu, J.; Yang, Y.; Xiao, B.; Li, Z., “Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing”, *IEEE Transaction Mobile Computing*, 18, 319-333, (2019). doi:10.1109/TMC.2018.2831230
- [50] Liu, X.F.; Zhan, Z.H.; Deng, J.D.; Li, Y.; Gu, T.; Zhang, J., “An energy efficient ant colony system for virtual machine placement in cloud computing”, *IEEE Trans. Evol. Comput.*, 22, 113–128, (2018) doi:10.1109/TEVC.2016.2623803.
- [51] Malekloo, M.H.; Kara, N.; El Barachi, M., “An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments”, *Sustain. Comput. Informatics Syst.*, 17, 9–24, (2018) doi:10.1016/j.suscom.2018.02.001.
- [52] Vakiliinia, S.; Heidarpour, B.; Cheriet, M., “Energy efficient resource allocation in cloud computing environments”, *IEEE Access*, 4, 8544–8557, (2016). doi:10.1109/ACCESS.2016.2633558
- [53] Sharma, Y.; Javadi, B.; Si, W.; Sun, D., “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy”, *J. Netw. Comput. Appl.*, 74, 66–85, (2016). doi:10.1016/j.jnca.2016.08.010
- [54] Long, Z.; Ji, W., “Power-efficient immune clonal optimization and dynamic load balancing for low energy consumption and high efficiency in green cloud computing”, *J. Commun.* 2016, 11, 558–563, doi:10.12720/jcm.11.6.558-563
- [55] Kaur, T.; Chana, I., “Energy efficiency techniques in cloud computing: A survey and taxonomy”, *ACM Comput. Surv.*, 48, (2015). doi:10.1145/2742488
- [56] Lee, Y.C.; Zomaya, A.Y., “Energy efficient utilization of resources in cloud computing systems”, *J.Supercomput.* , 60, 268–280, (2012). doi:10.1007/s11227-010-0421-3
- [57] Beloglazov, A.; Abawajy, J.; Buyya, R., “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”, *Future Gener. Comput. Syst.*, 28, 755–768, (2012). doi:10.1016/j.future.2011.04.017.

- [58] Scionti, A.; Goga, K.; Lubrano, F.; Terzo, O., “Towards energy efficient orchestration of cloud computing infrastructure”, In *Complex, Intelligent, and Software Intensive Systems. CISIS 2018. Advances in Intelligent Systems and Computing*; Barolli, L., Javaid, N., Ikeda, M., Takizawa, M., Eds.; Springer: Cham, 2019; Vol. 772, pp. 172–183, ISBN 9783319936581.
- [59] Khan, N.; Shrestha, R., “Optimizing power and energy efficiency in cloud computing”, In *Proceedings of the 9th International Conference on Cloud Computing and Services Science. CLOSER 2019*; 2019.
- [60] Tang, C.; Xiao, S.; Wei, X.; Hao, M.; Chen, W., “Energy-efficient and deadline satisfied task scheduling in mobile cloud computing”, In *Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018*; 2018; pp. 198–205.
- [61] Liu, N.; Dong, Z.; Rojas-Cessa, R., “Task scheduling and server provisioning for energy-efficient cloud- computing data centers”, In *Proceedings of the International Conference on Distributed Computing Systems*; 2013; pp. 226–231.
- [62] Zhao, H.; Qi, G.; Wang, Q.; Wang, J.; Yang, P.; Qiao, L., “Energy-efficient task scheduling for heterogeneous cloud computing systems”, In *Proceedings of the 21st IEEE International Conference on High-Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019*; 2019; pp. 952–959.
- [63] Li R., Mi N., Riedewald M., Sun Y., Yao Y., “A Case for Abstract Cost Models for Distributed Execution of Analytics Operators”, In: Bellatreche L., Chakravarthy S. (eds) *Big Data Analytics and Knowledge Discovery. DaWaK 2017. Lecture Notes in Computer Science*, 2017, vol 10440. Springer, Cham. https://doi.org/10.1007/978-3-319-64283-3_11

- [64] Chung-Chi H., Y-Che H., “Reliability and cost optimization in distributed computing systems”, *Computers & Operations Research*, Volume 30, Issue 8, 2003, pp. 1103-1119, ISSN 0305-0548, [https://doi.org/10.1016/S0305-0548\(02\)00058-8](https://doi.org/10.1016/S0305-0548(02)00058-8)
- [65] Kashish Ara Shakil, Mansaf Alam, Samiya Khan, “A latency-aware max-min algorithm for resource allocation in cloud”, *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 11, No. 1, February 2021, pp. 671-685, ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp.671-685
- [66] P. G., S., R. K., N., Menon, V.G. *et al.* A secure data deduplication system for integrated cloud-edge networks. *J Cloud Comp* **9**, 61 (2020). <https://doi.org/10.1186/s13677-020-00214-6>
- [67] Ahmed Sardar M. Saeed and Loay E. George, “Data Deduplication System Based on Content-Defined Chunking Using Bytes Pair Frequency Occurrence”, *Symmetry*, 2020, 12(11), 1841. <https://doi.org/10.3390/sym12111841>
- [68] A Vijayakumar and A Nisha Jebaseeli, “Pioneer approach of data deduplication to remove redundant data from cloud storage”, *International Journal of Advanced Research in Engineering and Technology (IJARET)*, Volume 11, Issue 10, October 2020, pp. 535-544, DOI: 10.34218/IJARET.11.10.2020.057
- [69] Dinesh Mishra, Sanjeev Patwa, “Attribute and Role-Based Deduplication”, *International Journal of Advanced Science and Technology*, vol. 29(7), pp. 14597 – 14606, 2020.
- [70] Duaa S. Naji and Loay E. George, “A Technique for Big Data Deduplication based on Content Attributes and Dictionary Indexing”, *IOP Conference Series: Materials Science and Engineering*, Volume 928, 2nd International Scientific Conference of Al-Ayen University (ISCAU-2020) 15-16 July 2020, Thi-Qar, Iraq

- [71] S. Ruba, A.M. Kalpana, "Machine Learning Techniques Used to Store Efficient Cloud Data Through Chunking And Data Deduplication Process", *Proteus Journal*, 2020, pp. 60-75, Vol.11, Issue 12. <https://doi.org/10.37896/PJ11.12/043>
- [72] Guangpin X, Bo T, et al., "LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication", 35th Symposium on Mass Storage Systems and Technologies (MSST), IEEE, pp. 299-310, 2019.
- [73] H. Fingler, M. Ra and R. Panta, "Scalable, Efficient, and Policy-Aware Deduplication for Primary Distributed Storage Systems," 2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2019, pp. 180-187, doi: 10.1109/SBAC-PAD.2019.00038.
- [74] H. A. S. Jasim and A. A. Fahad, "New techniques to enhance data deduplication using content based-TTTD chunking algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 5, pp. 116–121, 2018, doi: 10.14569/IJACSA.2018.090515.
- [75] N. Kumar, S. Antwal, G. Samarthyam and S. C. Jain, "Genetic optimized data deduplication for distributed big data storage systems," 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), 2017, pp. 7-15, doi: 10.1109/ISPCC.2017.8269581.
- [76] N. Kumar, R. Rawat, and S. C. Jain, "Bucket based data deduplication technique for big data storage system," 2016 5th Int. Conf. Reliab. Infocom Technol. Optim. ICRITO 2016 Trends Futur. Dir., pp. 267–271, 2016, doi: 10.1109/ICRITO.2016.7784963.
- [77] Y. Xuan Xing, N. Xiao, F. Liu, Z. Sun, and W. hui He, "AR-dedupe: An efficient deduplication approach for cluster deduplication system," *J. Shanghai Jiaotong Univ.*, vol. 20, no. 1, pp. 76–81, 2015, doi: 10.1007/s12204-015-1591-1.

- [78] Y. Zhang, Y. Wu and G. Yang, "Droplet: A Distributed Solution of Data Deduplication," 2012 ACM/IEEE 13th International Conference on Grid Computing, pp. 114-121, 2012, doi: 10.1109/Grid.2012.21.
- [79] S. S. Sengar and M. Mishra, "E-DAID: An Efficient Distributed Architecture for In-Line Data De-duplication," 2012 International Conference on Communication Systems and Network Technologies, 2012, pp. 438-442, doi: 10.1109/CSNT.2012.101.
- [80] Paulo, J., & Pereira, J. (2016). *Efficient Deduplication in a Distributed Primary Storage Infrastructure*. *ACM Transactions on Storage*, 12(4), 1-35. doi:10.1145/2876509
- [81] Singhal, S., Kaushik, A., & Sharma, P. (2018). *A Novel approach of data deduplication for distributed storage*. *International Journal of Engineering & Technology*, 7(2.4), 46. doi:10.14419/ijet.v7i2.4.10040
- [82] Shengmei Luo, Guangyan Zhang, Chengwen Wu, Samee U. Khan, Boafft: Distributed Deduplication for Big Data Storage in the Cloud, *IEEE Transactions on Cloud Computing*, Vol. 61, No. 11, January 2015
- [83] Tarun S., Batth R. S. (2019). Distributed Database Design Challenges and its Countermeasures-A Study. *Journal of the Gujarat Research Society* 21 (6), pp. 875-886
- [84] S. Tarun, R. S. Batth and S. Kaur, "A Review on Fragmentation, Allocation and Replication in Distributed Database Systems," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2019, pp. 538-544, doi: 10.1109/ICCIKE47802.2019.9004233
- [85] Chakravarthy, S., Muthura j, J., Varadara jan, R., and Navathe, S.B. An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis, *Distributed and Parallel Databases*, Vol. 2, No. 2, New York, April 1993.

- [86] Chu Pai-Cheng, A Transaction Oriented Approach to Attribute Partitioning, Information System, Vol. 17, No. 4, London, 1992.
- [87] Chu, W., and Leong, I.T., A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems, IEEE Transaction on Software Engineering, Vol. 19, No. 8, New York, August 1993.
- [88] Cornell, D., and Yu, P., A Vertical Partitioning Algorithms for Relational Databases, Proceedings of the Third International Conference on Data Engineering, Los Angeles, CA, February 1987.
- [89] Hammer, M., and Niammir, B., A Heuristic Approach to Attributes Partitioning, Proceedings ACM SIGMOD International Conference on Management of Data, Boston, MA, 1979.
- [90] Navathe, S., Ceri, G., Wiederhold, G., and Dou, J., Vertical Partitioning Algorithm for Database Design, ACM Transaction on Database System, Vol.9, No.4, New York, December 1984.
- [91] Navathe, S. and Ra, M., Vertical Partitioning for Database Design: A Graphical Algorithm, ACM SIGMOD, Portland, OR, June 1989.
- [92] R. Singh and K. S. Mann, "Improved TDMA Protocol for Channel Sensing in Vehicular Ad Hoc Network Using Time Lay," Proceedings of 2nd International Conference on Communication, Computing and Networking Lecture Notes in Networks and Systems, pp. 303–311, 2018.
- [93] A. Nayar, R. S. Batth, D. B. Ha, and G. Sussendran, G. "Opportunistic networks: Present scenario-A mirror review" International Journal of Communication Networks and Information Security," 10 (1), pp. 223-241, 2018
- [94] G.S Shahi, R.S Batth, S. Egerton, 2020 "MRGM: An Adaptive Mechanism for Congestion Control in Smart Vehicular Network", International Journal of Communication Networks and Information Security 12 (2).

- [95] Qi, H., & Gani, A. (2012, May). Research on mobile cloud computing: Review, trend and perspectives. In 2012 Second International Conference on Digital Information and Communication Technology and its Applications (DICTAP), IEEE, pp. 195-202.
- [96] Borkar, V., Deshmukh, K., & Sarawagi, S. (2001, May). Automatic Segmentation of text into structured records. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data, pp. 175-186.
- [97] Santini, S., & Jain, R. (1999). Similarity measures. IEEE Transactions on pattern analysis and Machine Intelligence, 21(9), pp. 871-883.
- [98] Huang, A. (2008, April). Similarity measures for text document clustering. In Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, Vol. 4, pp. 9-56.
- [99] Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas, 18(3), pp. 491-504.
- [100] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning. Neural and Statistical Classification, 13(1994), pp. 1-298.
- [101] Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. WSEAS transactions on computers, 4(8), pp. 966-974.
- [102] S. AXELBROOKE, "First Inbound and Response Tweets", Kaggle.com, 2020. [Online]. Available: <https://www.kaggle.com/soaxelbrooke/first-inbound-and-response-tweets/data?select=sample.csv> [Accessed: 27-Dec- 2020].
- [103] Yencken, L., Stopwords.txt, Gist., from <https://gist.github.com/larsyencken/1440509> [Retrieved December 27, 2020]

- [104] Metin, O., *Tweets about the top companies from 2015 to 2020*, Kaggle., from <https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020?select=Tweet.csv> [Retrieved December 27, 2020]
- [105] Lende, S. P., & Raghuwanshi, M. M. (2016, February). Question answering system on education acts using NLP techniques. In 2016 world conference on futuristic trends in research and innovation for social welfare (Startup Conclave) (pp. 1-6). IEEE.
- [106] Zeyu, X., Qiangqian, S., Yijie, W., & Chenyang, Z. (2018). Paragraph vector representation based on word to vector and CNN learning. *Computers, Materials & Continua*, 55(2), pp. 213-227.
- [107] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- [108] Bartunov, S., Kondrashkin, D., Osokin, A., & Vetrov, D. (2016, May). Breaking sticks and ambiguities with adaptive skip-gram. In *artificial intelligence and statistics*, pp. 130-138.
- [109] H. Guo, J. Zhou and C.A. Wu (2018), "Imbalanced Learning Based on Data-Partition and SMOTE", *Information*, Vol. 9, No. 9, pp. 238.
- [110] Kaur K., Laxmi V. (2019), "Hierarchical Clustering Based Improved Data Partitioning using Hybrid Similarity Measurement Approach", *International Journal of Innovative Technology and Exploring Engineering*, Volume-8 Issue-8, pp. 3008-2014.
- [111] Sashi Tarun, Ranbir Singh Batth, Sukhpreet Kaur, "A Novel Fragmentation Scheme for Textual Data Using Similarity-Based Threshold Segmentation Method in Distributed Network Environment", *International Journal of Computer Networks and Applications (IJCNA)*, Vol 7, Issue 6, Nov-Dec 2020, pp. 231-242. <https://doi.org/10.22247/ijcna/2020/205322>
- [112] Ahmed Osman, Assim Sagahyroon, Raafat Aburukba, Fadi Aloul, "Optimization of energy consumption in cloud computing data centers", *International Journal of Electrical and Computer Engineering (IJECE)*,

Vol. 11, No. 1, February 2021, pp. 686-698, ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp 686-698

- [113] Akanksha, “List-Based Task Scheduling Algorithm for a Distributed Computing System Using Artificial Intelligence”, In: Abraham A., Cherukuri A., Melin P., Gandhi N. (eds) Intelligent Systems Design and Applications. ISDA 2018 2018. Advances in Intelligent Systems and Computing, vol 941. Springer, Cham. (2020). https://doi.org/10.1007/978-3-030-16660-1_98
- [114] Gandomi, A., Movaghar, A., Reshadi, M. et al., “Designing a MapReduce performance model in distributed heterogeneous platforms based on benchmarking approach”, J Supercomput, 76, 7177-7203, (2020). <https://doi.org/10.1007/s11227-020-03162-9>
- [115] Lotfi, N., “Data allocation in distributed database systems: a novel hybrid method based on differential evolution and variable neighborhood search”, SN Appl. Sci. 1, 1724, (2019). <https://doi.org/10.1007/s42452-019-1787-3>
- [116] Tariq R., Aadil F., Malik M.F., Ejaz S., Khan M.U., Khan M.F., “Directed Acyclic Graph Based Task Scheduling Algorithm for Heterogeneous Systems”, In: Arai K., Kapoor S., Bhatia R. (eds) Intelligent Systems and Applications. IntelliSys 2018. Advances in Intelligent Systems and Computing, vol 869. Springer, Cham. (2019). https://doi.org/10.1007/978-3-030-01057-7_69
- [117] Suhelah Sandokji and Fathy Eassa, “Communication and Computation Aware Task Scheduling Framework Toward Exascale Computing”, International Journal of Advanced Computer Science and Applications (IJACSA), 10(7), (2019). <http://dx.doi.org/10.14569/IJACSA.2019.0100718>
- [118] I. Hababeh, “A Method for Fragment Allocation Design in the Distributed Database Systems”, The Sixth Annual U.A.E. University Research Conference, 2005.
- [119] P K Yadav, M P Singh and Kuldeep Sharma, “An Optimal Task Allocation Model for System Cost Analysis in Heterogeneous

- Distributed Computing Systems: A Heuristic Approach”, International Journal of Computer Application, 28(4):30-37, August 2011. DOI: 10.5120/3374-4664
- [120] Ahmed Younes. Hamed, “Task Allocation for Minimizing Cost of Distributed Computing Systems Using Genetic Algorithms”, International Journal of Advanced Research in Computer Science and Software Engineering, September 2012, Vol.2, Issue 9, pp. 202-209.
- [121] Dropbox, a file-storage and sharing service. [Online]. Available: <http://www.dropbox.com>
- [122] Google drive. [Online]. Available: <http://drive.google.com>
- [123] Mozy: A file-storage and sharing service. [Online]. Available: <http://mozy.com/>
- [124] Spideroak. [Online]. Available: <https://www.spideroak.com/>
- [125] P. Selvi, D. Shanmuga Priyaa, “An Enhanced Unsupervised Fuzzy Expectation Maximization Clustering for Deduplication of Records in Big data”, International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-3, pp. 988-993, 2019. DOI:10.35940/ijrte.C1269.1083S219
- [126] R. Parimala Devi, “Enhanced Fuzzy Ontology Based Record Deduplication”, Aut Aut Research Journal, Vol. 11(9), pp. 499-515 DOI:10.0001865.Aut Aut.2020.V11I9.463782.00746
- [127] M. Padmanaban and T. Bhuvaneshwari, “A Technique for Data Deduplication using Q-Gram Concept with Support Vector Machine “, International Journal of Computer Applications, Vol. 61(12), pp. 1-9, 2013
- [128] M. Padmanaban and T. Bhuvaneshwari, “An Approach Based on Artificial Neural Network for Data Deduplication”, International Journal of Computer Science and Information Technologies, Vol. 3(4), pp. 4637-4644, 2012.

- [129] Goli, M., & Rouhani Rankoohi, S. M. T., “A new vertical fragmentation algorithm based on ant collective behavior in distributed database systems”, *Knowledge and Information Systems*, 30(2), 435–455, 2011, doi:10.1007/s10115-011-0384-6
- [130] Arjan Singh, “SBBO Based Replicated Data Allocation Approach for Distributed Database Design”, *International Journal of Engineering Research and Technology*, Vol, 13(9), pp. 2461-2473, 2020, <https://dx.doi.org/10.37624/IJERT/13.9.2020.2461-2473>
- [131] Anju Khandelwal, “Optimal execution Cost of Distributed System: Through Clustering”, *International Journal of Engineering Science and Technology (IJEST)*, Vol. 3(3), pp. 2320-2328, 2011
- [132] Seema Yadav, Rakesh Mohan, Pradeep Kumar Yadav,” Task Allocation Model for Optimal System Cost Using Fuzzy C-Means Clustering Techniques in Distributed Systems”, *Ingénierie des Systèmes d’Information*, Vol. 25(1), pp. 59-68, 2020. <https://doi.org/10.18280/isi.250108>
- [133] J. Paulo and J. Pereira, “A survey and classification of storage deduplication systems,” *ACM Comput. Surv.*, vol. 47, no. 1, pp. 11:1–11:30, 2014. [Online]. Available: <http://doi.acm.org/10.1145/2611778>
- [134] J. Li, et al., “Secure distributed deduplication systems with improved reliability,” *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3569–3579, Dec. 2015. [Online]. Available: <http://dx.doi.org/10.1109/TC.2015.2401017>
- [135] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, “Secure deduplication with efficient and reliable convergent key management,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014. [Online]. Available: <http://dx.doi.org/10.1109/TPDS.2013.284>

Research Publications

This research work contributes following publications objective-wise:

Journal Publications:

[1] Sashi Tarun, Ranbir Singh Batth, Sukhpreet Kaur, "A Novel Fragmentation Scheme for Textual Data Using Similarity-Based Threshold Segmentation Method in Distributed Network Environment", International Journal of Computer Networks and Applications (IJCNA), 7(6), pp: 231 - 242, 2020, DOI: 10.22247/ijcna/2020/205322.

[2] Sashi Tarun, Mithilesh Kumar Dubey, Ranbir Singh Batth, Sukhpreet Kaur, "An Optimized Cost-Based Data Allocation Model for Heterogeneous Distributed Computing Systems", International Journal of Electrical and Computer Engineering (IJECE), ISSN 2088-8708, e-ISSN 2722-2578, Q2, Vol. 12, No. 6, December 2022, pp. 6373~6385, DOI: 10.11591/ijece.v12i6.pp6373-6385

Conference Publications:

[3] S. Tarun, R. S. Batth and S. Kaur, "A Review on Fragmentation, Allocation and Replication in Distributed Database Systems," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 2019, pp. 538-544, doi: 10.1109/ICCIKE47802.2019.9004233.

[4] Sashi Tarun, Ranbir Singh Batth, Sukhpreet Kaur, "Distributed System Design Issues, Challenges and Weaknesses: A Review", First International Conference on Recent Trends in Parallel and Distributed Processing (RTPDP-2021), GCET, Greater Noida, IOS Press Conference Series, Scopus Indexed, July 2021

[5] S. Tarun, R. S. Batth and S. Kaur, "A Scheme for Data Deduplication Using Advance Machine Learning Architecture in Distributed Systems," 2021 International Conference on Computing Sciences (ICCS), 2021, pp. 53-60, DOI: 10.1109/ICCS54944.2021.00019

A Review on Fragmentation, Allocation and Replication in Distributed Database Systems

Sashi Tarun
Research Scholar, School of CSE
Lovely Professional University
Phagwara, India
sashi.25347@lpu.co.in

Ranbir Singh Batth
School of CSE
Lovely Professional University
Phagwara, India
ranbir.21123@lpu.co.in

Sukhpreet Kaur
Department of CSE
Chandigarh Engineering College
Chandigarh, India
sukhpreet.4479@cgc.edu.in

Abstract— Continuous efforts on the improvement of existing techniques of data distribution methods were shared by researchers' time-to-time. The purpose behind is to work for an effective distributed working environment to shape out the distributed system to handle vast data, continuous network expansion, and its numerous users. To satisfy database queries in distributed system users generally initiate, execute and complete his request on his node or from adjoining nodes. Here, focus is on maximize the local data processing and minimize the communication cost between sites. Optimality of results can be achieved by proper partitioning of database into disjoint sets called fragments, placement of fragment into network sites by allocation, and replication to maintain data availability and control fault tolerance. So, there is a need to calibrate large distributed system to satisfy daily data pattern of users. This research paper highlights all the antecedent techniques, algorithms, methods, design framework etc. used for the designing of data fragments, allocation and replication in distributed system.

Keywords— mobile host, fixed host, fragment, allocation, replication, communication cost, optimal, pattern

I. INTRODUCTION

Fragmentation of data indicates partition of database into number of small independent parts called fragments. Accessing of data from fragments introduce partial data access and an environment of working with table views. It is a step towards selection of data-items using fine grained rather than coarse grained approach.

After the fragmentation, allocation of data fragments into defined geographical dispersed environment is managed either in centralized, fragmentation, full replication and partial replication strategy.

Replication in distributed database system, is to maintain data transparency by maintaining duplicate copies at each sites depends on users access or based on their mobility working behavior. It ensures data availability, fault tolerance, and reliability at individual sites ground. Eager approach of replication is required to achieve above stated assurances.

The sole purpose of different data distribution methods is to achieve overall distributed performance by:

- Dividing the workforce load into fragments and maintain easy data availability to them without wait or delay.
- Modular approach to ensure fast execution of sub-queries.
- Allowing further network expansion without complexity.

- Controlling usage of storage space.
 - Ensure easy data sites maintenance.
- Earlier centralized system had some flaws that lead to move towards de-centralized system:
- **Load Balance and Performance:** In centralized system, user's query follows a concurrent environment, in which users simultaneously access the database and load on the centralized system increases. It slow down the performance of the database, and response to several users at the same time for the same data-items seems difficult and instant response to database queries are normally get affected.
 - **Complexity and Expansion:** Data in the centralized system is increasing at an alarming rate due to large workforce. As a result, more storage capacity is required to maintain data and become difficult to understand the logical structure. Further expansion of database in this environment is all about invitation to accidental data loss.
 - **Data Maintenance and its Availability:** Maintenance of large data in centralized database indicates interruption in service to its users and affects the availability of the data for sometime. To overcome this flaw distributed database opens the door of easy data maintenance and its availability. Data availability can be from any of the network node in decentralized system.
 - **Fault tolerance:** There is a lack of fault tolerance feature in centralized system and it affects the data availability. But this become easy by distributed approach as it enables to work on fault tolerance by replication process. It maintains the data availability because failure of any node does not affect the overall performance.

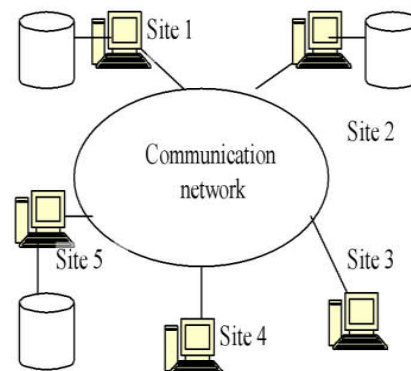


Fig.1. Distributed Database System

The key of this proposed paper is to highlights the need of data distribution and introduce its predecessor methodology adopted by researchers. Therefore, this paper focus on :

- Problem Area of Distributed Database System.
- The need of distribution of data.
- To depicts different problems and hindrance in the distribution of data.
- Taxonomy of distributed database system.
- To highlight all antecedent techniques used for mobile users in distributed system.

A. Problem Area of Distributed Database System

Adoption of DDBSs introduces different problems in the distributed environment. These problem areas are now subject of discussion and continuously efforts are given by the researchers to work for optimal approaches, so that distributed performance can be enhanced. Some of the problem areas of distributed database system in distribution of data domain are:

- *Data Coverage Issues*

Data-items contained in the fragments are not efficient to cover up database queries from outside. Normally, availability of data is not feasible sometimes from where the query initiated. It is because of bad design issues of distributed database fragmentation.

- *High Replication Costing*

Cost of replication is very high for maintaining identical data copies at each sites contain the data fragments. Data access operations are regularly performed by the database users for changing records or alterations in the database. In concurrent environment, data changes should be immediately reflected on other sides to maintain data consistency at every site.

- *Performance Degradation*

Availability of partial data at each node affects the data access performance. Access operations required more time to fetch the complete information from other nodes in the network. It increase delay in query response time problems.

B. Need of Data Distribution

To give better shape to large amount of data (accumulated from different data sources) designer has to choose an effective data distribution approach. It works to achieve the following objectives:

- *Data Transparency*

Data accessibility or its availability to mobile users are from any network sites and make them feel the overall system as a single coherent system. It permits for the execution of high level queries in the distributed database without looking into where the data is coming from.

- *Manage Workload*

Data distribution is helpful by following ways:

- a) To manage mobile workforce by optimizing processing power to different task routing towards their destination.
- b) Allow to access application interfaces from multiple locations as the suitability of work.
- c) Service continuation by routing the request to other path to a healthy node contains requested information.
- d) Allow to work in a transparent system by doing operations such as maintenance and upgrading of peer nodes

without interrupting data availability to other users at node stations.

- *Failure Transparency*

One of the drawbacks of centralized database is that it does not compromise with any failure. But, distributed database survive in case of failure of any of the network node. Load of failure node is equally taken by the nearby nodes to avoid any loss of data and processing delay.

- *Autonomy*

All sites in the distributed environment are working in an independent way. Every site is having own data and maintains his own information system. This autonomy features is highly supported in business for establishing different divisions, department to facilitate a strong system.

- *Location Transparency*

This is all about ease of access to data by users at different locations. Users from anywhere, anytime in the network get access their data easily. Communication system enables this transparency to access required data remotely.

- *Modular Structural Approach*

It helps to achieve following objectives without interruption of others:

- a) Easy modification in case of upgrading on the existing nodes,
- b) Addition to increasing the number of nodes for the existing network expansion and
- c) Removal of nodes/functionality in case of not required in near future.

- *Restricted Data Access*

Data distribution in the distributed environment allow user to view certain part of the data or information not as a whole. Here in this system, users intend to access only required information from the request node without the interest of other data.

C. Data Distribution Problems

Different problems occur during the distribution of data in distributed database system:

- *Behaviorial Practice*

A good data distribution schemes is always as per the interest of the users because users himself work with the data as a producer and consumer. Mobile users continuously change their database queries to satisfy their business requirements. Here, existing distribution schemes are not able to cope up from changing access pattern behavior. As a result data fragments become worthless and not able to cover database queries demanded or expected by the database users.

- *Network Expansion*

To satisfy every user globally there is a need to expand the size of the existing network by adding nodes at the appropriate locations. It is the communication system that facilitates the network nodes to coordinate with each other during transactions in the large network. But it has serious concern with the existing data distribution schemes. Are new nodes in the expanded network affect communication cost, time, and degrade performance. Sometimes, it leads to refragmentation and reallocation issues. Due to expansion, links between the nodes can be added or removed to define new communication path.

- *Lack in Architectural Design*

A good architectural design is helpful to build strong system and works for a long period of time. A design for the mobile world is a challenging task because design reflects the interconnection blueprints. A bad architectural design of the network affects the data distribution schemes. It raises anomalies during the transaction hours by showing incomplete, irrelevant information, failure of queries as a result of problem in data extraction. It results redesign of whole architectural design with a view to implement affective data distribution approaches.

- *Software Application Compatibility*

Every mobile unit depends on interfaces programs to perform their required database operations in distributed environment. Due to network expansion existing application programs are not compatible and able to extract data completely from each node. On the other side, to increase the capability by adjusting new data-items on the existing nodes seems unreachable to software applications. Moreover data synchronous issues with data nodes when transaction occurs by users using application programs. So, to make effective compatibility between application programs and data distribution schemes advance action is needed.

- *Integrity Check*

Integrity Control becomes more difficult at fragments level. As a result correctness at fragments level is not controlled easily. When large global schema is divided into independent pieces called fragments then they are treated as separate entities. In such cases integrity during data operation is not verified easily by the checker and result of this data redistribution problem may occur in the near future.

- *Testability Issues*

To smoothly run the system by entertaining all supporting users in distributed system, it is important that all data sites will be available for use most of the time without fault, failure and error. But testing in distributed system at node (site) level is not possible due to the existence of multiple data-items from different databases schemas. There is no relationship between data inside the fragments and allocated at distributed sites. Each site contains data information from multiple databases independent to each other.

D. *Data Distribution Hindrance:*

Some of the hindrances in the distribution of data in distributed system faced are:

- *Network Latency*

Delay during query response, replica propagation delay, and communication delay are common network hindrances who discard the effectiveness of data distribution approaches.

- *Resources Availability:*

Resources like power during regular usage, insufficient bandwidth to transfer and communication with others is required to smooth line the distributed database operations. Deficiency in these resources leads to implementation hurdle.

- *Disconnection*

Frequent disconnection of network during distributed operations is a hindrance to achieve performance. To

perform operations trusted service connection with the network is required.

II. RELATED WORK

Work on distributed Resource Description Framework (RDF) is performed to manage the growing massive RDF. To utilize this large volume RDF is partition into small parts called fragments and further approach the same for allocation in the distributed database environment. Here, focus is given to reduce the communication cost during the query processing tasks. It also ensures to maintain data integrity and approximation ratio due to frequent access patterns from outside. Here RDF graph is divided using three fragmentation strategies namely horizontal, vertical and mixed fragmentation based on frequent access patterns. It is also focus on balancing and allocation of fragments into different sites [12].

A systematic review on data distribution strategies i.e. Data fragmentation, Allocation and Replication is performed by the author. In this paper, it is indicated that different problems are faced by the designers during using and designing of these strategies. First, Data fragmentation is having problems of Join Optimization, when query trying to combine more than one fragments from more than one geographical site to fetch the required data. It reduces the response time. Secondly, includes data allocation problem about finding optimal technique helpful to allocate fragments to different sites.

A heuristic approach for fragmentation is proposed to reduce transmission cost (TC) of queries in distributed environment. Here, at initial stage fragmentation is based on cost-effective model in context of relational model and at later stage based on DDBS design. Different replication based allocation scenario were proposed i.e. mixed replication-based data allocation scenario (MAS), full-replication-based data allocation scenario (FAS), and non-replication data allocation scenario (NAS)[7].

A modified Bond Energy Algorithm (BEA) is proposed and it is a hierarchical process to make fragments vertically and allocate the fragments into geographical sites across the network. This algorithm use affinity of attributes and is helpful to generate cluster of attributes, to calculate cluster allocation cost and also decide about their appropriate sites for allocation. Here attributes accessed collectively by the same query are placed into one fragment [5].

A study was to review and compare the existing algorithms in design perspective with a view to identify their strength and weakness. This is just to present an affective design for the distribution of data fragments on the distributed environment [6].

A non-redundant dynamic fragment allocation technique is proposed and is based on the changing access pattern at different sites with a view to improve the performance. Here fragments reallocation is depend on access made on each fragment data volumes based on defined time constraint and threshold value. This proposed technique change the reallocation strategy by modifying the read and write data volume factor and introduced threshold time volume and Distance Constraints Algorithm. Write data volume is considered for the reallocation process when more than one sites approach for the fragments. This ensures the overall improvement of distributed system performance [11].

A hybrid optimized model using information on the type and frequency of queries for fragmentation of data horizontally and vertically and is based on supervised machine learning approach to produce non overlapping fragments. These fragments are maintained by archiving process rather than deletion operation on them. These fragments are used to facilitate searching operations based on index so that database tables are partition horizontally and vertically [8].

Two algorithms Modify Create Read Update Delete (MCRUD) and Matrix based Fragmentation (MMF) for efficient partitioning of large databases without query statistics. It shows that earlier approaches of partitioning were based on type and frequency of the queries called observed or experimental data. Here it is also indicated that earlier partitioning approach were not suitable because at the initial stage of the design of distributed database query statistics are not available. In his paper, an optimal fragmentation technique is proposed to partition global relations of a distributed database when there is no data access statistics and no query execution frequencies are available. When data access statistics and query execution frequencies are not available at the initial stage then MMF is responsible to partition relation in the distributed database. MCRUD is responsible to take fragmentation decision without using empirical data [14].

Work on different replication strategies in MANET, mobile database, distributed database, and cellular network etc is highlighted. It discuss about replication protocols as ROWA, ROWA-A and Quorum Based Protocol. All are replica control protocol, ROWA is responsible to fetch the read request values to the nearest site from the occurrence of request location and replicate the changes to all the sites. An alternative approach is in the form of ROWA-Available and is same as ROWA in the case of read operations but replicate the changes only to all available replica copies and do not bother about any replication failure. ROWA-A is responsible for maintaining the availability of data but do not compromised with the correctness of data. In case of failure users are working with stale value of data. It shows incorrect or out-of-date copy of replica. Quorum based replica is to update the subset of replicas rather than replicate the changes as a whole and helpful to maintain consistency of data [2].

An integrated approach is proposed for DDBMS namely data fragmentation, network sites clustering and allocation of fragments. This work is responsible to improvise problems in the form of; fragmentation, redundancy in data allocation and redistribution problem due to complexity, to maintain data availability and consistency issues [14].

It is also highlighted, to maintain inconsistency issues faced by the mobile users during the access of database in his mobility from any of the activity center. Here a 5-cube structure with nearest-neighbors propagation distribution protocol is proposed to make useful distributed database system for the mobile users. It ensures consistent data to all mobile users/sites by dynamically replicates the changes to all its adjoining sites from the transactional node [16].

A new dynamic deallocation approach for a given fragment as Update Matrix (UM) and Distance Cost Matrix (DM) is proposed. It works on the basis of changing data

access patterns in replicated and non-replicated distributed database system. It was assumed that fragments are allocated on network site is based on applied frequency value of the database data items. Reallocation of data fragments on the remote sites is planned based on communication and update cost value. Each fragment is having update cost value. Fragment having maximum update cost value is considered for reallocation and chosen candidate site to store fragments to minimize the communication cost. UM is defined as the value getting after issuing of update query at a particular site for the manipulated fragment. In this approach when same query is applied at more than one site, then queries can be treated different to each other and having different frequency value [4].

An algorithm called Simulates Annealing with Genetic Algorithm (SAGA) is used for optimal allocation of fragment in distributed environment. Here, allocation of data is depends on access patterns for fragments and focused on reducing the allocation cost during movement of data fragment from one site to another [3].

A problem in large scale mobile distributed database system is introduced by replication architecture for distributing replica and updates propagation protocol to propagate recent changes occurred in different objects in distributed system to achieve the data consistency. In his paper presented a new binary hybrid approach consists of pessimistic and optimistic replication strategy, helpful to work with large number of replicas and reduce the data inconsistency rate to support mobile users. This replication is helpful to achieve update propagation delay reduction and less communication cost. Here replication architecture is proposed consist of master, zone and cell level and Wheel-based updates propagation protocol is used to maintain replicas in mobile distributed database [1].

A decentralized approach for dynamic table fragmentation and allocation in distributed database systems is proposed. It is based on observation and monitoring of the sites access patterns to tables which reforms fragmentation, replication, and reallocation based on recent access history aiming at maximizing the number of local accesses compared to accesses from remote sites [9].

A new technique called Attribute Level Precedence (ALP) to partition global schema/database relations at initial and later stage in case of non-availability of data access statistics and query execution frequencies. ALP technique is capable to take advance decision for fragmentation at the initial stage (i.e. knowledge gathered during requirement analysis phase) without empirical data statistics. ALP is a table responsible to fragment a relation horizontally based on the importance of an attribute in a network site [15].

III. TAXONOMY OF DISTRIBUTED DATABASE SYSTEMS

Dealing with data to know about different facts or figures is possible, if data is classified, arranged and distributed into different sites using optimal classification. Classification is a straight path to know about different categories and helpful to frame out distributed database system as shown in Fig. 2.

A. Classification of Distributed Database System

In control dimension, DDBS is of two types:

1) *Control based on locality*: In control dimension, DDBS is of two types:

a) *Centralized Control DDBS*: In this, only a particular sites is responsible to plan, monitor and control the distributed data transactions occurs in different sites and ensure data consistency at each sites.

b) *Distributed Control DDBS*: In this, different sites equal responsible to control and cooperate the processing of the transactions to ensure data consistency.

2) *Degree of Integration*: In this dimension, distributed control DDBS is of two types:

a) *Strongly Integrated or DDBSs*: Here information about the executing transactions is with each site to ensure global consistency on other sites.

b) *Loosely Integrated or MDBSs*: In this, each site are not willing to share control information with any other sites.

3) *Degree of Heterogenity*: Here sites can be homogenous and heterogeneous in both strongly and loosely integrated in the DDBSs.

IV. DATA DISTRIBUTION DESIGNING TECHNIQUES

Based on studies, a checklist is created to identify different distribution designing techniques and get to know about work introduced by the researchers' time-to-time for achieving the optimality of fragmentation, allocation and replication of data in distributed system.

A. Distributed Designing Techniques in Frequency

Some of the contributions in Distributed Database System are depicted below in TABLE 1 starts from 2019 to 1979 onwards with their frequency counts.

TABLE 1 DIFFERENT DISTRIBUTED TECHNIQUES WITH THEIR FREQUENCY AND PERCENTAGE

S.No.	References	Research Categories	Fragmentation	Allocation	Replication
		Framework/Algorithms/Methods/ Techniques/Review/Approach/ Model/Strategy/Architecture/ Solution/Survey/Protocol			
1	Peng et al. (2019)	Framework	Yes	Yes	-
2	Ezechiel et al. (2019)	Review	Yes	Yes	Yes
3	Abdalla et al. (2019)	Approach	Yes	-	-
4	Rahimi et al. (2018)	Algorithm	Yes	Yes	-
5	Fuaad et al. (2018)	Review	-	Yes	-
6	Lwin et al. (2018)	Technique	-	Yes	-
7	Orijji et al. (2018)	Model	Yes	-	-
8	S. I. Khan (2016)	Algorithms	Yes	-	-
9	D. Rane et al. (2016)	Strategy	-	-	Yes
10	Al-Sayyed et al. (2014)	Approach	Yes	Yes	-
11	S. Tarun (2012)	Strategy	-	-	Yes
12	Abdalla (2012)	Approach	-	Yes	-
13	Abdalla (2011)	Algorithm	-	Yes	-
14	A. Ahmad (2011)	Architecture	-	-	Yes
15	Hauglid et al. (2010)	Approach	-	Yes	Yes
16	S. I. Khan et al. (2010)	Technique	Yes	-	-
17	K. Matiasko et al. (2008)	Algorithms	-	Yes	-
18	T. Ulus et al. (2007)	Algorithm	-	Yes	-
19	Mondal et al. (2006)	Algorithm	-	-	Yes
20	I. Hababeh (2005)	Method	-	Yes	-
21	Grebla et al. (2004)	Solution	-	Yes	-
22	N. H. Daudpota (1998)	Model	-	Yes	-
23	X. Liu et al. (1995)	Survey	-	-	Yes
24	Navathe et al. (1995)	Algorithm	Yes	-	-
25	O. Wolfson et al. (1992)	Algorithm	-	-	Yes
26	Cheung et al. (1990)	Protocol	-	-	Yes
27	D. Agarwal et al. (1990)	Protocol	-	-	Yes
Frequency			9	14	10
Percentage (%)			32	50	37

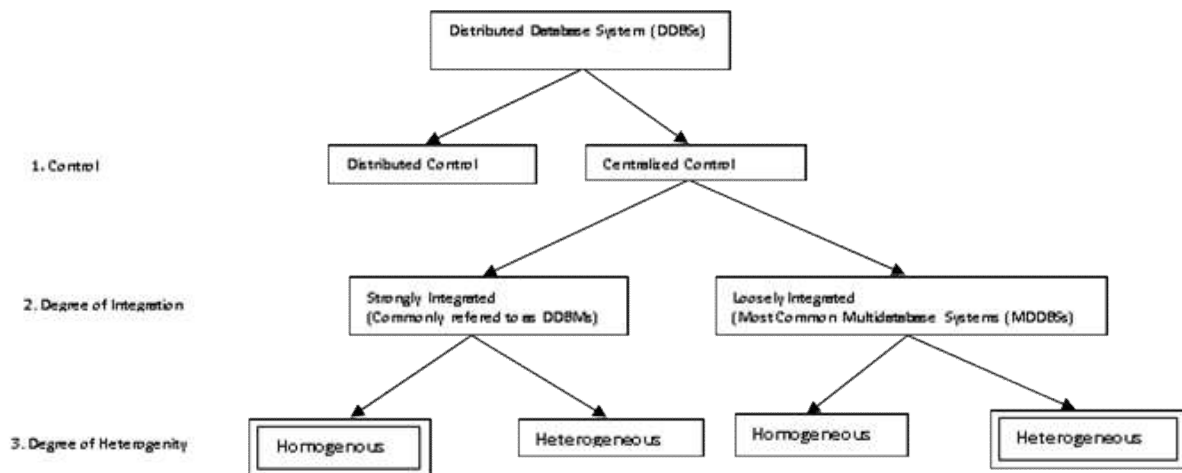


Fig.2 Taxonomy distributed database system

B. Analysis of Design Techniques

Different research activities carried out in fragmentation, allocation and replication of data in distributed system. For better understanding the data distribution Table 1, is having all details showing about the achievements yearly wise. It helps to categorizes different methodologies based on where the decisions on distribution methods are carried out.

Above table targeted about how to promote performance by introducing methodologies through their research time-to-time. Continuous research on distribution methods shows possibility of flaws and gaps in the earlier approaches precluded. It is an effort for finding new techniques to promote performance in DDBSs. Here, percentage and frequency indicates the usability of data distribution techniques and its ratio respectively. This act support information about the overview of distributed design using different techniques.

Based on the review of Table 1, it is concluded that fragmentation is having less (32%) research work initiatives to shape distributed system for better availability and performance. As fragmentation is consider as first blueprint in which other distribution techniques are depending on. Here research on fragmentation, allocation and replication is not carried out simultaneously by the researchers. All are carried out separately causes some of the flaws and gaps are still remains.

So, few researchers are having work on fragmentation issues followed by allocation to reach up to an optimal solution. Both are having impact on the design of DDBSs.

V. CONCLUSION AND FUTURE WORK

In this article, it is to address about the needs, problems, hindrance and predecessor methodologies involved in the design of distributed database system. With the continuous expansion of network and increasing data requirements, there is a need to work for affective optimal solutions and workout

for resolving various problems and hindrances in the form of data inconsistency, query performance, communication error, communication cost, availability and reliability of data etc. Different techniques are depicted in Table 1, to show information about researcher efforts in the field of fragmentation, allocation and replication of data.

Future work is to work for techniques, models responsible to perform effective data fragmentation, allocation and replication to handle distributed workforce with optimal results in comparison with existing methodologies.

REFERENCES

- [1] A. Ahmad, "A Novel Replication Strategy for Large Scale Mobile Distributed Database Systems", *Journal of Engineering Science and Technology*, Vol. 6, No. 3, pp. 268-299, 2011.
- [2] D. Rane, M.P Dhore, "Overview of Data Replication Strategies in Various Mobile Environment", *IOSR Journal of Computer Engineering*, pp. 01-06, 2016
- [3] H. I. Abdalla, "An Efficient Approach for Data Placement in Distributed Systems", *Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering*, pp. 297-301, 2011
- [4] H. I. Abdalla, "A New Data Re-Allocation Model for Distributed Database System", *International Journal of Database Theory and Application*, Vol. 5, No. 2, pp. 45-60, 2012.
- [5] H. Rahimi, F. Parand, "Hierarchical Simultaneous Vertical Fragmentation and Allocation using Modified Bond Energy Algorithm in distributed databases", *Applied Computing and Informatics* (14), pp.127-133, 2018
- [6] H. A. Fuaad, A. A. Ibrahim, A. Majed and A. Asem, "A Survey on Distributed Database Fragmentation, Allocation and Replication Algorithms" , *British Journal of Applied Science and Technology*, 27(2): pp.1-12, 2018
- [7] H. Abdalla, A. M. Artoli, "Towards an Efficient Data Fragmentation, Allocation, and Clustering Approach in a Distributed Environment", *Information*, Vol 10, No.112, 2019
- [8] I. B. Orijji, I.C. Ejiofor, "A Hybrid Model for Data Fragmentation in Distributed System", *International Research Journal of Computer Science*, Issue 04, Volume 5, pp. 186-192, 2018
- [9] J. O. Hauglid, N. H. Ryeng, K. Norvag "DYFRAM: Dynamic Fragmentation and Replica Management in Distributed Database Systems", *Distributed Parallel Databases*, pp 157–185, 2010.
- [10] K. K. Ezechiel, S. K. R. Agarwal, "A Systematic Review on Distributed Databases Systems and their Techniques", *Journal of*

- Theoretical and Applied Information Technology, Vol. 96, No. 1, 236-266, 2019
- [11] N. K. Z. Lwin, T. M. Naing, "Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System", ICIBMS, Bangkok, Thailand, pp.300-305, 2018
- [12] Peng Peng, Lei Zou, Lei Chen, Dongyan Zhao, "Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload", IEEE Transactions on Knowledge and Data Engineering, Vol. 31, No. 4, pp. 670-685, April 2019
- [13] Rizik M. H. Al-Sayyed, D. Suleiman, M. A. A. Itriq, I. Hababeh, "A New Approach for Fragmentation and Allocation to Improve the Distributed Database Management System Performance", Journal of Software Engineering and Applications, Vol. 7, pp. 891-905, 2014.
- [14] S. Tarun, "A Reputation Replica Propagation Strategy for Mobile Users in Mobile Distributed Database System", International Journal of Grid and Distributed Computing, Vol. 5, No. 4, 2012.
- [15] S. I. Khan, "Efficient Partitioning of Large Databases without Query Statistics", Database System Journal, pp. 34-53, 2016.
- [16] S. I. Khan and A. S. L. Hoque, "A New Technique for Database Fragmentation in Distributed System", International Journal of Computer Application, Vol. 5, No. 9, pp. 20-24, 2010.

Distributed Systems Design Issues, Challenges and Weaknesses: A Review

Sashi Tarun^{a,1}, Ranbir Singh Bath^b, Sukhpreet Kaur^c

^aResearch Scholar, School of CSE, Lovely Professional University, Punjab

^bSchool of Computer Science & Engineering, Lovely Professional University, Punjab

^cDeptt. of CSE, Chandigarh Engineering College, Chandigarh

Abstract. It has now become simple to build or operate on large distributed systems due to the continuous advancement of connectivity technologies, network software, high computing components, and storage space. Good system designs pave a lot for end-users and allow them to operate with high data reliability, availability, and accuracy in a diverse environment. But now, in the technical architecture of a distributed structure, several small but important bottlenecks influence workforce performance. For better scope, it is necessary to put more stress on some of the vulnerabilities that stop distributed systems from detracting their efficiency. This article stresses some of the issues, challenges, and weaknesses of the distributed system and brought into knowledge the need for system tuning to build a robust system.

Keywords. distributed, bottlenecks, reliability, availability, consistency, workforce, efficiency

1. Introduction

Through these features distributed structures are derived:

- It is a combination of nodes or sites working autonomously and having their local memory [1-2].
- All nodes are communicating with each other by different message-passing schemes [3-4].
- All nodes are geographically dispersed in nature [5].



Figure 1. Distributed System

Maintaining of data and its retrieval depends on the planned architectural design of distributed systems. It indicates how data is organized in a diverse environment and makes it easy to retrieval and usefulness to others. A good design work towards users' satisfaction level and propose how data is being distributed among all the sites to achieve data availability, reliability, and consistency. For getting data, a query processor has to depend on more than one node and result to delay in query processing and degrade system performance. Earlier, users were stationary, but with technological advancement, everyone in their network can access their resources from anywhere, anytime. Good system design satisfies end-users by responding to their diverse queries, providing distributed analysis services, and helping in building effective application interfaces for a diverse environment. In a centralized system design data or information, extraction is from one location. Reduction in overall performance, disrupt data communication, hold long-wait, high-latency, massive storage to maintain data, and high maintenance cost is some problems encountered in central storage. Alternatively, decentralized systems divide data into different sites in a connected network. In this connected arrangement all nodes can access data from each site equally without scheduling or any access priority.

A distributed system is the step towards designing a system to support a diverse application environment and involves fragmentation [6], allocation [7], and replication processes to build it effectively [19]. Fragmentation partition the data depending on users' access patterns. Allocation is for the placement of partitions into geographical sites. Replication is to achieve consistency and availability of data in each site by propagating changes over distributed sites.

Over time several designs were proposed to overcome existing problems in the distributed systems. Proposed techniques were added to the existing design to strengthen and overcome the prevailing challenges. These architectures focus on improving issues such as scalability, data processing flexibility, and effectiveness of communication, regulate efficiency, robustness, and achieve resource sharing. Existing architectural designs of the distributed environment are discussed below in table 1.

Table 1. Existing Distributed Architectural Design

S.No.	Distributed Architectural Design	Responsibilities	Limitations
1	Distributed Agent-Based Architecture [8]	This architecture is having self-organizing features and is responsible to automate different grid operations and perform changes on its topology or configuration itself. To achieve self-organizing behavior Multi-Agent System (MAS) architecture was proposed. These agents are autonomous, loosely connected in nature, and interact with each other for solving any problem, which is beyond the capacity of the problem solver.	Centralized distribution systems are not capable of avoiding earlier issues such as increased communication saturation, component up-gradation, and their mapping, slower decision rate, etc.
2	Distributed Adaption Systems Architecture [9]	This system was helpful to improve the efficiency, robustness, and scalability in heterogeneous environments. In this system, the adaptation engine watches the execution and activates the active adaptation system in case of detection of any differences.	Adaptive distributed applications are not addressed in earlier research.

3	Distributed Access Control Architecture [10]	This architecture addresses all concerns related to security challenges in cloud computing. This architecture is implemented for multitenant and virtualized environments so that sharing of physical resources can be controlled.	This research is to tackle untrusted tenants who further result in unauthorized information flow and risk of side-channel attacks.
4	Symmetric Distributed Server Architecture [11]	This architecture believes in using workstation servers in place of large mainframes. This approach is to manage thousands of devices/nodes in a large geographically dispersed environment. Here message bus is responsible to establish communication between the devices and integrated them at a lower cost.	Earlier approaches were not effective to use the network as a whole. It is not effective for communication in wide geographical distributed systems and expensive due to the dependency of the large mainframe.
5	ReDy Architecture [12]	This architecture is flexible to work in large-scale applications. In a distributed system, this solution guarantees scalability, fault tolerance, and dynamicity to achieve performance in the system.	Due to dependency on the integration of applications in the distributed system complexity is increasing day by day. To achieve expandability, efficacy, trustworthiness, flexibility, and effectiveness it is necessary to tightly integrate all processing, communication, and control technologies.
6	Lambda Architecture [13]	Lambda Architecture is an approach to tackle the limitation of the existing CAP theorem. According to the CAP theorem data in the distributed environment satisfy only two states either consistency with availability, or consistency with partition tolerance, or availability with consistency, or availability with fault tolerance, or fault tolerance with consistency, not as a whole in together.	Existing CAP theorem work for only two states out of three states.
7	Microservices Architecture [14]	This architecture is a way to decompose any large system into modules architectural patterns. Each module communicates using an interface called Application Programming Interface. This architecture is a solution to achieve scalability. In comparison to monolithic, this architecture does not affect the functions of large applications due to the failure of a single module. Programs codes are smaller and easy to deploy.	This was limited only to defeat the problems of conventional monolithic structural design.

The proposed paper explores all prevailing challenges, weaknesses, and issues involves in the designing of a distributed system. An upcoming section highlights obstacles in tuning distributed design, challenges, weaknesses, and issues involved in the distributed design process. In the end, suggestions to overcome the prevailed challenges are highlights showing different design aspects of distributed heterogeneous system.

2. Motivation

To work in a widely distributed environment it is necessary to deal with various factors equally responsible to strengthen the system design. They are resource sharing, computation speedup, throughput, performance, reliability, and communication, etc. Despite the diversity, some prevailing flaws need to be worked upon and seem to be an obstacle in the designing of a distributed system. To make distributed design more efficient and effective it is necessary to work on and to highlight all key factors responsible to shape a geographically dispersed setup.

3. Related Work

Efforts were given by the researcher to build a distributed system more effectively so that issues, challenges, and weaknesses up to some extent can be resolved. To achieve performance, reliability, availability, efficiency, robustness, and security in a widely distributed environment researchers depend on different architectural designs as discussed in [15] but due to network spreading, and increasing data behavior distributed systems become complex. It is essential to design a distributed network of self-balancing intelligent nature to avoid distributed environment problems. An unsupervised machine learning algorithm following a similarity-based-threshold segments scheme was implemented to fragment twitter textual data into a distributed environment to handle data of different size in rows [16]. For building multimedia applications in a distributed system issues of networking and data management were resolved in [17]. To do this, there is a need to strengthen the storage of data in the networking environment because the process gets affected during the integration of information among sites in [18]. Work on Distributed Real-Time (DRT) applications was conducted and it is found that it has become difficult to implement the same on the distributed system (DS) due to the existing design structure. As an outcome, a structural blueprint of distributed real-time applications was proposed in [20]. An algorithm as Weighted Least Connections (WLC) was proposed to perform load balancing to improve the performance in a distributed system-based web server in [21]. Distributed system software issues, challenges, and problems are also highlighted by [22]. It was also highlighted that there is a growing real-world interest in the exercise of industrial IoT techniques with distributed systems to achieve high performance in the existing design [23]. Work on the importance of design patterns to the developers for resolving recurring problems was also highlighted in [24].

4. Distributed Design Tuning Obstacles

Researchers put forth a lot of effort to enhance and regulate the performance of the existing distributed system. Despite of this, there are obstacles in system design that affect the tuning progression. It has been very problematic to build an efficient distributed architecture and to improve the performance part in such situations as depicted below:

- a. It has become difficult in a distributed processing environment to achieve an exact working view of the system due to the involvement of multiple components in the execution of tasks in a distributed environment.

- b. To achieve distributed tuning, log details and metrics play a key role to strengthen the working environment. As resource consumption is not limited to individual components but involves multiple sharing of components among nodes. As a result, integration of log details or viewing different metrics collectively in a single place belonging to different nodes is a serious concern.
- c. It has become difficult to know which component needs to be monitored and when to increase during load to achieve performance in a distributed system.
- d. Cascading failures are hard to solve. Component Failure propagates to other connected components and reduces traceability and handling of problems (fault, failure or errors) becomes complicated.

5. Issues in the Design of Distributed System

An issue indicates about lacks in the progression of process involved. So, Overhaul on the existing design strategies will facilitate us to enhance the performance of a distributed environment include:

- a. **Incomplete network information:** Unawareness about network infrastructure is a big hurdle for designing a distributed system. So, complete knowledge of the architectural design is required to design the robust system.
- b. **Scalability/Expandability:** In a distributed environment this feature enables us to enlarge the existing design architecture by adding other network nodes to increase the network size. It not only for increasing the network size but also accommodates data items into existing network sites/nodes wherever applicable.
- c. **Load balancing:** This is to distribute the data equally among sites to increase the throughput. In this, the load balancer equally looks into the scheduling part so that requests are responded to time without any delay. Load on sites depends on the number of user's login for regular transactions and sometimes, new sites are added to the system and the state of the distributed system rapidly changes.
- d. **Data availability and its consistency:** Placement of data items into different sites should be in such a way that maximum responses can be satisfied. In case of change on any data items, the same changes need to propagate into other sites to maintain consistency. In a concurrent environment query satisfaction level can be handled by effective scheduling strategy or by providing relevant data from the available nodes at a low cost.
- e. **Maintenance:** Design architecture should be in maintainable capacity. So that changes in the logical structure and further addition of new features to increase its functionality can be satisfied.
- f. **Concurrency:** Mobile users during their mobility simultaneously apply data requests to data sites. In such an environment propagating changes dynamically and handling concurrent requests from every corner is a big challenge.
- g. **Data Deduplication:** Having duplicate copies of data in the geographical sites not only creates confusion or ambiguity situations but also increases existing storage space. It is equally responsible to degrade the performance of the systems because it will take more time to parse each data record in relation. As a result, queries results are not showing any relevancy due to data duplicity.

6. Challenges in Distributed System Design

Challenges are the trends and directions to the system designer's, the researchers to continuously work towards the improvement of existing challenges that affect distributed system design are:

- a. **Data Incompleteness:** Non-availability of whole data at nearby sites from the requested site in response to any query. It slows the query response time and throughput. This challenge came into existence when data availability is not on the requested site.
- b. **Bugs Issues:** Distributed bugs create hurdles in the system performance of the system by decreasing the response time of any request. These bugs spread themselves like an epidemic within the network of computers and slow down data flow in the channel. They make the whole network busy and affect network efficiency.
- c. **Implementation Cost:** Designing and implementation a distributed system are very expensive and not bearable to small organizations thinking to avail or hire their services.
- d. **Network Untrustworthiness:** It disrupts the communication progress between two or more ends during the fetching of data. This unreliability of the network indicates:
 - i. The request may be queued and has to wait for their turn.
 - ii. Chances of request loss in the network.
 - iii. Response node may have any fault, failure situation
 - iv. Network overloading results delayed in response
 - v. Network partition problems divide the whole network into two fragments/partitions. As a result, the response from the other side is not in a position to reach the source, where the request generates.
- e. **Latency:** In a widely distributed network, deterioration in data during the transferring of data or information from one place to another indicates a latency issue. As enterprises thought about migration of their valuable services into distributed environment structure but network latency issue not able to lead this operation because migration activity involves backup and restore operations and found not flexible in disaster recovery in an emergency due to its speed.
- f. **Data Synchronization:** Changes took place in geographical sites needed to propagate all network sites to maintain the correctness, availability, and reliability of data. But, due to variation in the hardware configuration i.e. clock pulse minor calibration differences are found, as a result, the system might refuse to sync with others. It is impossible to take care of one global notion of the same clock time. This results in the inconsistency of data at each node. On the other side, network delay and network gateway/firewall equally responsible to affect the data synchronization process.
- g. **Fault Tolerance and overloading:** To establish reliability in a large geographical disperse system fault tolerance methods play an important role. It is maintained by applying a dynamic replica propagation strategy, resilience in hardware resources so that corrective operations can be performed, check pointing on redundant data to recover the system during any fault, failure, or error by restoring the system to

the previous checkpoint status. Overloading shows an imbalance in the functioning and degrades the performance of the system. There is a need for a good fault-tolerant system, which carefully examines the reason for failures and responds for the same.

- h. **Distributed tracing:** We are rapidly going towards an environment that is increasingly dynamic and dispersed. This means that distributed tracing is extremely important. And distributed monitoring is something that makes managing dynamic structures convenient for you.

7. Major Weaknesses of Distributed Design

In distributed environment forecast about the lifetime of any system is unpredictable. Success depends upon wrapping up every component of the distributed system in the present and future scenarios. But despite imparting the best measures still, there is a need to work out some weaknesses. Some of them are discussed below:

- a. **Lack of vision:** Time-Based Replication scheme introduces many such mistakes that further resulted in inefficiency, insecurity, and costly to maintain data in a distributed environment. But with modern mathematics approaches such as the Paxos algorithm resolve such hurdles up to some extent.
- b. **Lack of Effective Software:** As a distributed environment is an existence of different machines working for a sophisticated environment where systems are communicating or interacting simultaneously for their task completion. It is difficult to deliver refined programs that enable us to react in different situations like route-finding during congestion, the network goes down, and security issues in a distributed environment. As data access is for all users in the system so safeguarding of data from collusive attack prevails there.
- c. **Lack of Self-Management Self-Control System:** Such a system is responsible to configure, heal, and optimize machine and network services to make them functional in disrupting situations. Configuring is all about the removal or adding of any existing resource or new components. Healing is for automatically finding, identifying, restricting, and regain due to the event of any fault, failure, or error. Optimization refers to the tuning of existing processes and proactively reacts in different conditions.
- d. **Lack of Legal Jurisdiction:** Users' freedom to use different resources intended for and performs transactional activities to maintain their data. Business services accessed from outside required some government bodies who intervene in legal disputes related to data security, privacy, its ownership, rights on intellectual property, and auditing in case of any complicity.
- e. **Global status of Knowledge:** Every node is having its memory for maintaining personal as well as other data or information. In such a case, it is not easy to keep track of the worldwide status of the complete distributed environment. Every process execution involves in the distributed environment prove a coherent vision of the system but in real it shows a limited observation of the system.

8. Overcoming Distributed Design Challenges

After analysis it is concluded that designers and researcher's has to participate with their sincere efforts to overcome design challenges to construct a flexible dispersed

structural design. This can be possible by the inclusion of the following suggestions and are discussed below:

- a. Distributed architectural design is having an access control mechanism to more strengthen their security part.
- b. Distributed architecture should be designed in such a way that communication and integration between connected components can be easily managed at a low cost.
- c. Distributed systems should have scalability features. Modularization is one way to achieve scalability. It allows dividing large systems into independent parts and enabling easy deployment of services without loss or failure.
- d. Distributed systems design should have the capability to control error rates and other communication issues by using an automated approach. This is all about introducing self-organizing, adaptive features in the distributed design architecture to detect any difference to improve the working environment.
- e. Distributed design should have less dependency on join queries to access dispersed data. By this data, retrieval can be achieved without delay.
- f. Distributed design architecture is mostly affected due to high operating and energy consumption costs. Different design projects are practically failed due to the inefficiency of handling overall costs. So, design architecture should be based on the least-cost strategy and capable to work efficiently.

Conclusion

This paper concludes all alarming issues, challenges, and weaknesses prevailed in the distributed design systems. Efforts in the field of expandability, network applications, robustness, data management, security, and reliability are the chief agenda these days to build the robust design. Due to changes in the behavior of network users, data size, and technological development trends it has now become easy to overcome bottleneck issues and burdens involved in the progression of operation or event. So, to build an optimal system it is required to inculcate above design factors to overcome such occurrence in near future. As future work, there is a need to introduce an adaptive approach for the reduction of total cost to strengthen the allocation process in distributed environment and cover most of the above bottlenecks.

References

- [1] Andrews, Gregory R. Foundations of Multithreaded. Parallel and Distributed Programming. Addison–Wesley. 2000. p. 8–9 291. ISBN 978-0-201-35752-3.
- [2] Ghosh, Sukumar. Distributed Systems – An Algorithmic Approach. Chapman & Hall/CRC. 2007. p. 3. ISBN 978-1-58488-564-1.
- [3] Andrews, Gregory R.. Foundations of Multithreaded. Parallel and Distributed Programming. Addison–Wesley. 2000. p. 291. ISBN 978-0-201-35752-3
- [4] Ghosh, Sukumar. Distributed Systems – An Algorithmic Approach. Chapman & Hall/CRC. 2007. p. 3. ISBN 978-1-58488-564-1
- [5] Lynch, Nancy A.. Distributed Algorithms. Morgan Kaufmann. 1996. ISBN 978-1-55860-348-6.
- [6] S. Tarun, R. S. Bath and S. Kaur. A Review on Fragmentation, Allocation and Replication in Distributed Database Systems. 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE). Dubai. United Arab Emirates. 2019. p. 538-544. doi: 10.1109/ICCIKE47802.2019.9004233
- [7] Ranichandra, C., Tripathy, B.K. Architecture for distributed query processing using the RDF data in a cloud environment. Evol. Intel. . 2019. <https://doi.org/10.1007/s12065-019-00315-5>

- [8] K. Saxena, A. R. Abhyankar, Distributed architecture for self-organizing smart distribution systems, in IET Smart Grid. 2018. vol. 1. no. 4. p. 113-122. 12. <https://doi.org/10.1049/iet-stg.2018.0029>.
- [9] Zouari M., Segarra MT., André F., Thépaut A. An Architectural Model for Building Distributed Adaptation Systems. In: Brazier F.M.T., Nieuwenhuis K., Pavlin G., Warnier M., Badica C. (eds) Intelligent Distributed Computing V. Studies in Computational Intelligence. 2011. vol 382. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-642-24013-3_15
- [10] Yuan P, Liu P, Wu B, Jin H, Zhang W, Liu L. TripleBit: a fast and compact system for large-scale RDF data. Proc VLDB Endow. 2013. 6(7). p.517–528.
- [11] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref and A. Ghafoor. A Distributed Access Control Architecture for Cloud Computing. IEEE Software. 2012. March-April. vol. 29. no. 2. p. 36-44. <https://doi.org/10.1109/MS.2011.153>.
- [12] Slimani Y., Najjar F., Mami N.. An Adaptive Cost Model for Distributed Query Optimization on the Grid. In: Meersman R., Tari Z., Corsaro A. (eds) On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. OTM 2004. Lecture Notes in Computer Science. 2004. vol. 3292. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-540-30470-8_26
- [13] Trung A. D. Big Data: Lambda Architecture in a nutshell. gitconnected. 2020. Oct 9. <https://levelup.gitconnected.com/big-data-lambda-architecture-in-a-nutshell-fd5e04b12acc>
- [14] Joe N. What are microservices?. cloudacademy. 2019. April 10. <https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/>
- [15] Weiss C, Karras P, Bernstein A. Hexastore: sextuple indexing for semantic web data management. Proc VLDB Endow. 2008. 1(1). p.1008–1019.
- [16] Sashi Tarun, Ranbir Singh Bath, Sukhpreet Kaur. A Novel Fragmentation Scheme for Textual Data Using Similarity-Based Threshold Segmentation Method in Distributed Network Environment. International Journal of Computer Networks and Applications (IJCNA). 2020. 7(6). p. 231 – 242.DOI: 10.22247/ijcna/2020/205322.
- [17] Das S, Agrawal D, El Abbadi A. G-store: a scalable data store for transactional multi key access in the cloud. In Proceedings of the 1st ACM symposium on cloud computing. ACM. 2010. p.163–174.
- [18] Lu X.. Symmetric Distributed Server Architecture for Network Management System. In: Zhou X., Xu M., Jähnichen S., Cao J. (eds) Advanced Parallel Processing Technologies. APPT 2003. Lecture Notes in Computer Science. 2003. vol. 2834. Springer. Berlin. Heidelberg. https://doi.org/10.1007/978-3-540-39425-9_50
- [19] S. Tarun. Reputation Replica Propagation Strategy for Mobile Users in Mobile Distributed Database System. International Journal of Grid and Distributed Computing. 2012. vol. 5. no. 4. p. 55-64.
- [20] A. Valadares, C. V. Lopes. A Framework for Designing and Evaluating Distributed Real-Time Applications. 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real-Time Applications. Toulouse. 2014. p. 67-76. <https://doi.org/10.1109/DS-RT.2014.17>.
- [21] Y. Jiao, W. Wang. Design and Implementation of Load Balancing of Distributed-system-based Web Server. 2010 Third International Symposium on Electronic Commerce and Security. Guangzhou. 2010. p. 337-342. <https://doi.org/10.1109/ISECS.2010.81>.
- [22] K. S. Mishra, A. K. Tripathi. Some issues challenges and problems of distributed software system. Int. J. Comput. Sci. Inf. Technol.. 2014. vol. 5. no. 4. p. 4922-4925.
- [23] K. Iwanicki. A Distributed Systems Perspective on Industrial IoT. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). Vienna. 2018. p. 1164-1170. <https://doi.org/10.1109/ICDCS.2018.00116>.
- [24] H. Jiang, D. Liu, X. Chen, H. Liu, and H. Mei. How Are Design Patterns Concerned by Developers?. 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Montreal. QC. Canada. 2019. p. 232-233. <https://doi.org/10.1109/ICSE-Companion.2019.00090>.



A Novel Fragmentation Scheme for Textual Data Using Similarity-Based Threshold Segmentation Method in Distributed Network Environment

Sashi Tarun

School of Computer Science and Engineering, Lovely Professional University, Phagwara, India.
sashitarun79@gmail.com

Ranbir Singh Batth

School of Computer Science and Engineering, Lovely Professional University, Phagwara, India.
ranbir.21123@lpu.co.in

Sukhpreet Kaur

Department of Computer Science and Engineering, Chandigarh Engineering College, Mohali, India
sukhpreet.4479@cgc.edu.in

Published online: 25 December 2020

Abstract – Data distribution is one of the most essential architectures of any serving network. Data storage and its retrieval depend a lot on how the data is organized in the distributed environment. With the fast development of technology, the requirements of users have also changed. A user who was stationary earlier has become mobile now and requires access to the data from anywhere in the world. An unorganized data structure will result in output delay in the network and may further result in user migration from one service provider to another service provider. Data fragmentation is one of the most essential parts when it comes to data storage. Organized data always gives convenience to others to use it conveniently. Due to the vast collection of data extraction of information in a fast manner is very complicated. So, to achieve performance in a distributed system an optimal strategy is required to overcome previous lapses and serves the maximum number of users in a wide geographical network. This research paper proposes a novel relative based fragmentation method that analyses the attributes of the data in relative architecture and is helpful to achieve query performance with better speed and accuracy. To assess the current proposed work a comparison has been drawn between k-means dependent cosine similarity measurement and hybridization of cosine and soft-cosine partition methods for data partitioning. Mentioned results in the article shows that the proposed similarity-based threshold segmentation method outperforms the existing in terms of partitioning strategy, precision, and recall parameters to achieve performance.

Index Terms – Fragmentation, K-Means, Similarity, Data Partitioning, Threshold, Segmentation, Precision, Recall.

1. INTRODUCTION

A distributed system refers to the use of independent computers engaged to share various resources in the

connected networks. A good distributed design is having the capability to cover each data-items requirement raised by the users. Users looking for desire data or information, using different query angles. Some of them focus on data-items belonging to a single table or a combination of more than one. From different angles, user queries are classified as fine and coarse-grained, single database or multi-database, and follow a collective and selective approach to reach required data [1]. If knowledge grows at a rapid rate day by day, design architecture should be scalable to handle vast amounts of information in the future.

The mechanism of fragmentation, allocation, and deduplication of data is associated with improving data in a distributed network. Proper utilization of available storage space is achieved by dividing the large global data into small independent parts called fragments or segments [2]. These independent segments help to reduce the load in a widely distributed environment as compared to accessing data from a single large data schema. With distributed systems, issues such as scalability, data availability, security, searching speed, and inconsistency of data can be easily managed. It has become difficult to work with a single large data system and to entertain millions of users simultaneously with high data accuracy.

The growth of cloud computing, VANET, OPPNETs is the product of parallel technology, software technology, and network infrastructure innovations [3-4]. This is a new form of a computer model that provides users with the data, applications, and various IT resources through the network as a service without delay in the exchange of information [5]. Cloud computing can be considered as a kind of infrastructure

RESEARCH ARTICLE

management tool, resource management through virtualization technology, which consists of a large capacity resource pool. Cloud users will send requests through the network and then receive the service. In which dynamically deploy, modify, and reconfigure the resource pool, and cancel the operation, etc are included [6].

The availability of numerous services over the Internet is cloud computing. These assets include data processing software and apps for records, servers, computers, networks, and tablets. You can store files in cloud-based storage in an external database rather than storing them on your hard drive or local storage unit. Tools and data resources can be used as long as an electronic device has access to the Internet. Cloud storage is referred to as such because it is possible to remotely recover the stored information in the data or a virtual space. On remote servers, cloud service providers allow users to store files and programmes and then access all data online. This means that the user does not have to be in a certain position to use it in order to be able to function remotely.

Cloud storage is a popular choice for individuals and businesses for many reasons, including cost savings, increased efficiency, quality and efficiency, reliability and security [7].

The key aim of this study is to implement a new fragmentation architecture suitable for scalable question addressing in the distributed network world in terms of textual data. Earlier data fragmentation was based on empirical data and far-reaching to get the desired result. This approach introduces a novel relative-based fragmentation architecture where there is no ground reality and similarity calculations are carried out on the textual data to reach the conclusive result using vector calculations. This paper uses a mixture of similarities of cosine, soft cosine, and hybrid similarity as a differentiation between the entities of data for partitioning.

Existing design not proved to be effective on diverse data trends include textual data context. Earlier techniques focus on finding similarities between more than one documents but this technique is responsible to find the similarity in the relation itself by comparing each row to one another and apply similarity calculation on vector values. So, there is a need to depend on the required strategy suitable for a diverse environment with adaptive nature.

As follows, the paper is structured. The classification of data is in the form of a category based on related attributes is addressed in Section Segmentation. Section Similarity Measures helps to discover the relationship between the rows in relation or two data-items using Cosine, Soft Cosine similarity, and hybrid similarity. In the proposed methodology section steps are evaluated one by one i.e. selection of dataset, stop word removal, word-to-vector conversion, implementation of cosine, soft-cosine, and hybrid similarity,

determination of initial centroid of the dataset used, Euclid distance calculation, finding centroid positions for each cluster, creating fragmentation, and validating the fragments using machine learning and neural network are included. The outcomes and discussion of the research work section is included to illustrate the feasibility of the work proposed. Comparative analysis is done at the end of the section to equate the new model with the current scheme.

1.1. Segmentation

Segmentation is the method of collecting data with similar properties or separating cloud data into smaller, coherent, and interconnected areas. Text segmentation is a process by which a document is split into smaller parts, typically called segments. It is used extensively in word-processing. These segments are classified as word, phrase, subject, sentence, or any unit of information, depending on the task of the text analysis. The method of removing coherent blocks of text is Text segmentation. The section is called the boundary section or passage.

There are several explanations of why a split document could be useful for the analysis of the text. One explanation for this is that it is smaller and more coherent than whole documents. Segmentation of text is a big problem when it comes to obtaining information. It aims to divide a text into homogeneous segments i.e. segments with the following characteristics: (a) each segment has a particular topic and (b) adjacent segments deal with different topics. These segments can be traced as appropriate to a query from a broad base of unformatted or loose text [8].

1.2. Similarity Measures

When there is available ground truth for the clustering then the similarity value will be evaluated through the ground truth of the cluster or region but when there is no ground truth of the region or cluster then the ground truth becomes radical and hence similarity measures are calculated through vector calculation. Discourse is the measurement of the equivalence of two pieces of evidence. In the sense of data mining, an agreement is commonly defined as a gap along with the dimensions describing the objects' properties. The degree of similarity will be high if this distance is small; if a distance is large, the degree of similarity will below. The similarity measure is used in many ways, including plagiarism, asking for a similar question previously asked about Quora, collaborative filtering in recommendation systems, etc. A similar measure may be described as the distance between various points of data. While the similarity is a quantity that represents the strength of the relationship between two data items, the difference is between the two data items measuring divergence. Three similarity measures are used in conjunction in this study, namely Cosine, Soft Cosine similarity, and hybrid similarity [9].



RESEARCH ARTICLE

The resemblance is determined in the 0 to 1 [0, 1] scale.

- Relationship equal to 1 such that if X = Y
- Similarity is equal to 0, unless X is equal to Y

Where, X, Y are two different vector lists.

Similarity is arbitrary and relies extensively on the domain and its use. Two fruits, for instance, are similar because of color or height, or taste.

1.3. Cosine Similarity

Similarity is in general, a measure of similarity; that is, how similar things are compared to similar things. One was with the use of vectors, an equation for the computer. A vector is literally a quantity which has both size and direction. A vector is considered a 1-dimensional sequence in Computer Science. The resemblance of cosine is a method used to calculate the angle of cosine between them. The point product of the two vectors is required for finding the angle between the two vectors as shown in Figure 1. Measurement of cosine equality assumes the uniform point sum of the two objects. By determining the cosine relation, we will effectively attempt to find the cosine of the angle between the two lines. The 0° cosine is 1, and it is less than 1 for every other variable. Therefore, it is an orientation and not a magnitude judgment: two vectors with the same direction have a cosine similarity of 1, two vectors with a 90° similarity of 0, and two vectors with the same direction have a similarity of -1, irrespective of their magnitude [10].

$$\text{Cos } \theta = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

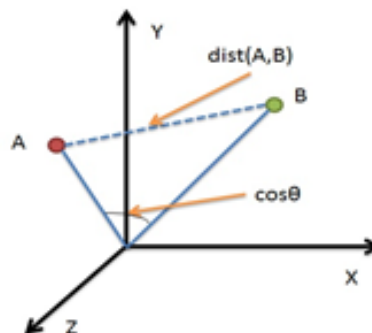


Figure 1 Similarity of Cosine

1.4. Soft Cosine Similarity

A soft cosine agreement tests attribute to the agreement. The conventional criterion of Cosine conformity determined similarity based on features determined by the model of vector space (VSM), which are completely different from each other. On the other hand, it can be a great advantage of soft cosine similarity if one needs to use a criterion of agreement that can help with the grouping or classification of documents [11].

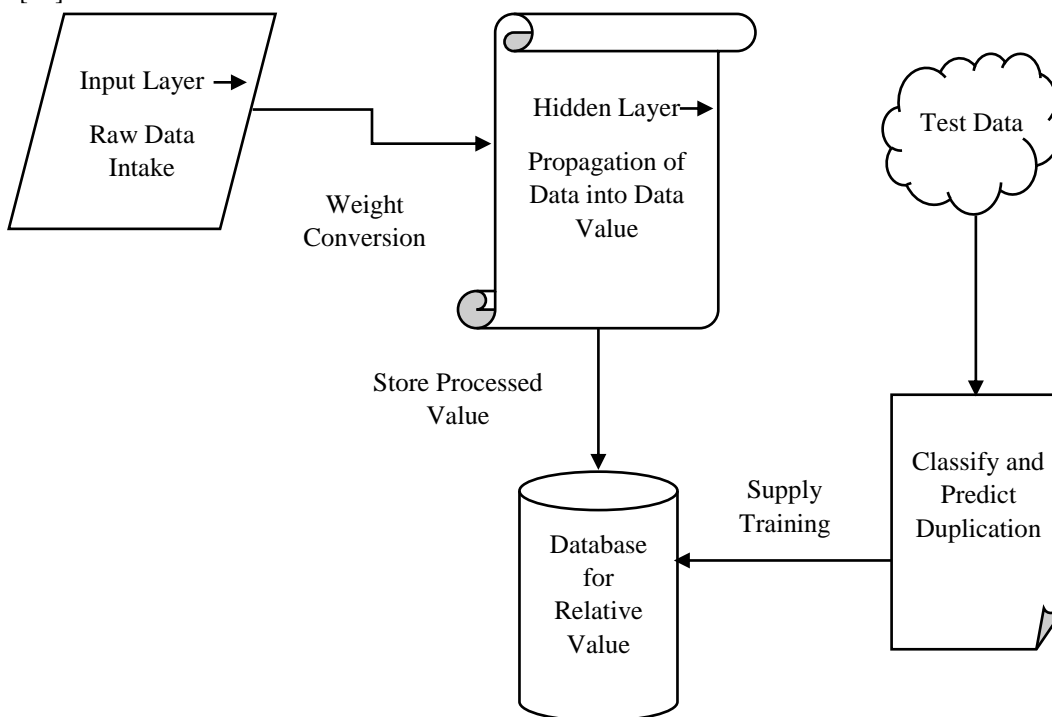


Figure 2 Learning Computer Architecture

RESEARCH ARTICLE

1.5. Hybrid Similarity

In this similarity measures, the features of cosine similarity and soft cosine similarity are combined.

1.6. Machine Learning

Machine learning (ML) is an artificial intelligence (AI) technique that without specific programming provides devices with the ability to learn and strengthen automatically. Machine learning focuses on the development of computer systems capable of viewing and learning knowledge themselves. Based on the explanations we present, the learning process begins with insights or data, such as examples, direct knowledge or input, looking for data patterns, and making informed future decisions. The key goal is to allow computers learn automatically and adjust behavior appropriately without human intervention or support [12]. ML categorization is always undertaken as given below;

- Machine learning algorithm supervised
- Unsupervised Algorithm for machine learning
- Semi-supervised algorithm for deep learning
- Machine learning algorithm for strengthening [13]

The learning computer architecture (shown in Figure 2) works completely on the relative information model. It learns from what is provided to it. It is separated into the input layer, the hidden layer, and the output layer into three parts. The input layer takes the data as raw data and transforms it against the specified goal label into a more understandable form. The hidden layer propagates the meaning of the data and generates the cross-validation training platform.

2. RELATED WORK

Several contributions were given by the researchers to build the distributed system robust. It was noticed that the researchers did not stress the efficacy of the proposed data partitioning work and that it was important to strengthen it. This issue arises due to methods of unsupervised data partitioning. Some researchers have used clustering methods, but the accuracy is lower because of the presence of unsupervised algorithms, and it is important to use the theory of similarity for clustering. K-methods are considered to be a less popular clustering algorithm since clustering is too difficult and costly to use than k-mean. The suggested algorithm uses a standard deviation that decreases the overall time for formulating the cluster by a simple k-mean. The proposed solution splits the gap with the standard deviation of the square root. This modified k-mean does even better than k-mean and k-methods. Less time is needed to formulate the clusters. But neither k-means nor k-methods also work on very large-scale outcomes. The authors have not used any kind of similarity measurement technique for distance

calculation between different types of data that results in poor clustering performance, and it must be integrated with k-means using optimization approaches [14]. Researchers suggest a deep neural network such as CODEnnn (Code-Description Embedding Neural Network). CODE does not fit the textual resemblance, but includes code fragments and high-dimensional vector field examples in natural language, as well as associated vectors in the code fragment and the accompanying definition. Code fragments associated with natural language questions can be obtained by the associated vector representation in accordance with their vectors. In the queries that must be handled, the task could even be semantically identified with the keywords. The researchers did not include source code management structures in this study to help symbolize, and the deep neural network is used and limited for the basic benefit of information engineering issues [15]. For image co-segmentation tasks, a new clustering algorithm called salience-guided limited cosine similarity clustering method (SGC3) has been proposed, where a one-step clustering technique extracts the usual foreground. In the method, the unsupervised significant prior is used to direct the clustering mechanism's auxiliary partition-level information. To ensure the robustness of the noise and outliers in a given previous one the similarity between the instance level and the partition level is used for joint estimation. Eventually, the optimization of associated K-means aims to successfully solve the objective function. Experimental outcomes from two widely used data sets show that the proposed solution has achieved successful performance against the most mature distribution methods [16]. A systematic experimental analysis of twenty-four benchmark functions in a test suite. ABC (Artificial colony of bees) is a very common and effective tool for optimization. ABC still does however have a lack of convergence. In order to further increase ABC convergence velocity, a new form of ABC (CosABC) is proposed to pick better neighbors based on cosine similarity. Under the direction of chosen neighbors, a new solution search equation was applied to reduce the constraint of ABC undirected search. There is a further contrast with some of the most sophisticated algorithms to check Cos-ABC supremacy. The related results of the comparison show that Cos-ABC is efficient and competitive [17]. To find similar knowledge for a user whose original question cannot be addressed precisely, clustering-based fragmentation is suggested. Approximation algorithms and lookup tables are used to give a better shape to the distributed system for supporting flexible query answering [18]. Work for optimized fragmentation approach on each attribute was conducted to know about their retrieval and update frequencies in each site. And proposed a synchronized horizontal fragmentation approach to reduce data locality issues and total cost. In this work, if query (Q) is initiated from multiple (M) locations, this query will be interpreted as a separate query for each position with a different radio

RESEARCH ARTICLE

frequency [19]. To achieve fragmentation an algorithm based on agglomerative hierarchical clustering was introduced to reduce the number of iterations. It mainly focused on the minimization of transmission cost [20]. It is proposed to render fragments vertically with an Updated Bond Energy Algorithm (BEA). This algorithm utilizes attribute affinity and seeks to create clusters of attributes and attributes that are individually evaluated by the same query [21]. A hybrid fragmentation approach for deductive database systems is proposed as HFA for horizontal fragmentation and, RCA and DVF for vertical fragmentation. This is a two-phase process deductive database is fragmented using variable bindings and dependency relationships represented by dependency graph [22]. For the efficient partitioning of large datasets without query statistics, MCRUD and MMF algorithms have been suggested. It is suggested here that earlier partitioning methods were not acceptable because there were no usable statistics at the initial stage of the implementation of distributed database query statistics. [23]. Work on frequent access patterns (FAP) is given to reflect the behavior workload to ensure the data integrity and ratio of approximation. It presented a data structure that was based on trees by utilizing the depth-first search (DFS) coding for maintaining them as well as to manage newly entered queries [24]. The fragmentation mechanism has recently been shown to have a detrimental effect on the performance of negatively exported processes. Finally, by merging Process Mining (PM), Social Network Analysis (SNA) and Text Mining, the fragmentation process and improved knowledge sharing among port Community System (PCS) actors have been improved so that process efficiency can be achieved [25]. Study on data fragmentation in the public and private sectors is being carried out in order to hold information in a structural archive in order to attain data security [26].

It is concluded that, continuous efforts were given to improve partitioning strategies for distributed network environment. Earlier, partitioning of data was not using query statistics but later on the basis of the behavior of users' frequent access pattern has used to make partitions. Due to lack in picking neighbors for the clustering and as a result query get affected during response and suffered delay. So for effective partitioning, information sharing between network nodes, maintaining efficiency, controlling delay and balancing data load a new scheme is required work for diverse environment.

3. PROPOSED METHODOLOGY

The steps are as follows:

- i) Begin by uploading unlabeled Twitter data.
- ii) Apply filtering of data by removing English Stop Words.
- iii) Apply word to vector operations.

- iv) Apply similarity calculations on the set of row list.
- v) Finding of initial centroid of data.
- vi) Calculation of Euclid distance of each tweet.
- vii) Finding the number of centroids.
- viii) Creating fragments using K-Means.
- ix) Validate the fragments using ML Training and classify using a neural network.

To achieve this, we have followed further processes in sequence. The step description of each process is provided in the next sections. Each step is individually implemented in Matlab R2016a.

3.1. Dataset Used

The dataset used for the proposed work is from the sample of data and is accessible from [27] on 27.12.20.

The above-defined dataset consists of tweets in unlabelled form, which needs to be segmented, and for segmentation. In this dataset "text" column having 1048576 instances of data out of that 93 instances are used for further implementation of the relative fragmentation experiment.

3.2. Stop Word Removal

The foremost step is to filter English stop words. The list of those stop words can be accessed from stop list [28] on 27.12.20.

The process of removing English stop words (as shown in Figure 3) from the tweets helps to decrease the dimension of the available data.

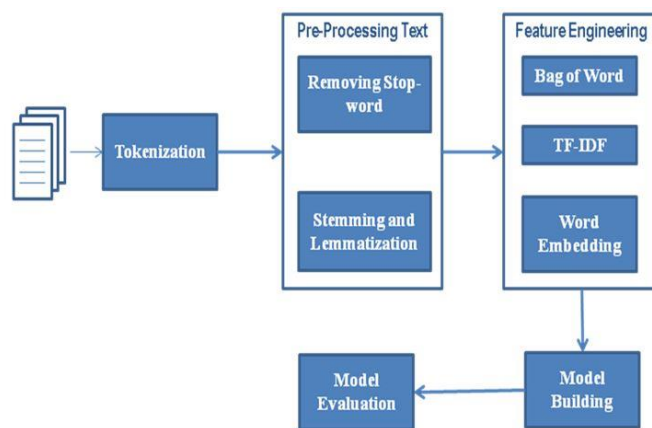


Figure 3 Stop Words Removal Process

Prepositions, articles, nouns, seem to be the most common words in text documents, etc. These words do not provide meaningful information about the text. Stop list words were omitted from the text because certain words in information retrieval (IR) software are not called keywords. For E.g. By

RESEARCH ARTICLE

maintaining an English stop word dictionary, English stop words are deleted from each text file in the data set [29].

For the removal of stop words, the code snippet is given below in Figure 4.

```

for i=1:r
    currentdata=rawdata{i,1};

    words=getwords(currentdata);
    allwords(i)=words;
    filteredwords=filterdata(words, stopwords);
    allfiltered(i)=filteredwords;
    for wd=1:numel(filteredwords)
        w2v(i, wd)=sum(double(filteredwords(wd)));
    end
end

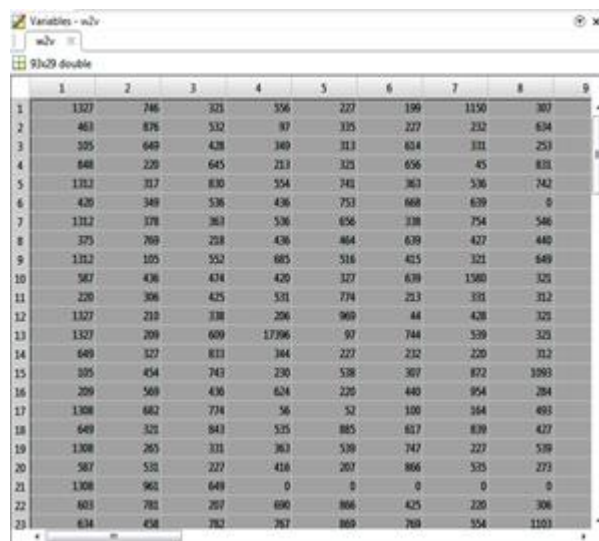
```

Figure 4 Stop Word Removals and W2V Conversion

3.3. Word to Vector

Similarity is usually a test of similarity, i.e. how similar or similar objects are compared. By using vectors, one method of calculating similarity is. A vector is essentially a number that has both direction and magnitude. A 1-dimensional sequence is considered a vector in Computer Science. A way to build a compact space of word vectors is Word2vec. It takes as input a broad text corpus (tweets after stop words have been removed) and assigns each word in the tweet a vector. First a dictionary is generated and then the vector representation of the terms is computed. Contextual proximity based vector representation: the words in the text adjacent to the same words (and thus have a similar meaning) in the vector representation have indexes of high similarity. The values of the vector after approximation (as seen in Figure 5) are represented in the dataset for each row of words in column 5. Therefore, the next step is to evaluate the similarity index of the data using three similarity measures named Cosine, Soft Cosine, and hybrid similarity index. A detailed description of these three measures is provided in the upcoming sections.

Skip-gram and Continuous Bag of Words are the two primary models in word2vec. In the Skip-gram model, terms are predicted from a word in their context, and the most possible word is chosen on the basis of the context in the CBOW model. To get the output of the probability distribution of each term, the output layer uses a softmax feature or a combination of it. Input and output words are given in one-hot encoding in these models, such that a single row is chosen *W* when multiplied by the matrix *W* connecting the input and hidden layers. Dimension *N* is the algorithm hyper parameter and the qualified *W*-output matrix, since its lines contain vector representations of terms. [30].



	1	2	3	4	5	6	7	8	9
1	1327	746	321	556	227	199	1150	307	
2	461	876	532	87	335	227	232	634	
3	505	649	438	349	313	654	331	253	
4	848	228	645	213	325	656	45	831	
5	1312	317	830	554	741	363	536	742	
6	426	349	536	436	753	668	639	0	
7	1312	378	363	536	656	338	754	546	
8	375	769	258	436	464	639	427	440	
9	1312	105	552	685	516	415	321	649	
10	567	436	474	420	327	639	1580	321	
11	220	306	425	531	774	213	331	312	
12	1327	210	338	206	969	44	428	321	
13	1327	209	609	1796	97	744	539	321	
14	649	327	811	344	227	232	220	312	
15	105	454	743	230	538	367	872	1093	
16	209	569	436	624	226	440	954	284	
17	1308	682	774	56	52	100	364	493	
18	649	321	943	525	865	617	839	427	
19	1308	265	311	363	539	747	227	539	
20	567	531	227	418	267	866	535	273	
21	1308	961	649	0	0	0	0	0	
22	603	781	267	690	866	425	220	306	
23	634	438	782	387	869	769	554	1103	

Figure 5 Vector Representation of the Word

To speed up the training of Skip-gram and CBOW models, modifications are used softmax, such as hierarchical softmax and negative sampling, which allow calculating the probability distribution faster than in linear time from the size of dictionary [31-32].

3.4. Cosine/ Soft Cosine / Hybrid Similarity Index

A document in the vector model is considered as an unordered set of terms. Terms to retrieve information are the words that make up the text to obtained essential or useful information. Here Cosine similarity is applied to calculate the similarity index for the uploaded document to the rest of the text in the set. For example, if we have considered 100 rows in the tweet, then the similarity (either cosine or soft cosine) is determined by comparing the word vector to the rest of the 99 rows. In this way for row 2, row3, row4.....row 100, have to be determined.

Input: Word to Vector data

Output: simvalue=Calculatecossim(v1, v2)

1. Calculatecossim(v1, v2) = [];
2. nume = 0; //numerator
3. deno=0;//denominator
4. deno1=0;
5. deno2=0;
6. for I = 1 → v1.length
7. nume=nume+v1(I)*v2(I);
8. End for
9. for J = 1 → v1.length

RESEARCH ARTICLE

10. $deno1=deno1+v1(J)^2;$
11. $deno2=deno2+v2(J)^2;$
12. End for
13. $deno=sqrt(deno1)*sqrt(deno2);$
14. $simvalue=nume/deno;$
15. Return: $simvalue$ as output
16. End function;

Algorithm 1 Cosine Similarity between Vectors

Algorithm 1 showing the functioning of cosine similarity, it is represented by calculating the cosine angles between two vectors $v1$ and $v2$. To do this, in relation each row is compared with other rows using a vector list and use numerator and denominator as a variable. It is calculated by multiplying each vector with one another row-wise sequentially and stores their result in $nume$ variable. And $deno$ is calculated by multiplying by the square root of $deno1$ and $deno2$. $deno1$ and $deno2$ is the square of $v1$ and $v2$ respectively. In the end, $simvalue$ is calculated by the division of $nume$ and $deno$.

Input: Word to Vector data

Output: $sc=Calculatesoftcosine(v1, v2)$

1. $Calculatesoftcosine(v1,v2)=[]$
2. $sc=0;$
3. $num=0;$
4. for $I = 1 \rightarrow v1.length$
5. for $J = 1 \rightarrow v2.length$
6. $num=num+ v1(I)*v2(J);$
7. End for
8. End for
9. $avalue=0;$
10. $bvalue=0;$
11. for $I = 1 \rightarrow v1.length$
12. for $J = 1 \rightarrow v1.length$
13. $avalue=avalue+v1(I)*v1(J);$
14. $bvalue=bvalue+v2(I)*v2(J);$
15. End for
16. End for
17. $avalue=sqrt(avalue);$
18. $bvalue=sqrt(bvalue);$

19. $deno=avalue*bvalue;$
20. $sc=num/deno;$
21. $sc=sc/(max(v1)/max(v2));$
22. End function

Algorithm 2 Soft Cosine Similarity

To achieve accuracy in the result of cosine similarity an improved algorithm as soft cosine similarity is proposed. Algorithm 2 is used to calculate soft cosine by a division of numerator and denominator. The numerator is the multiplication of $v1$ and $v2$ for each row with other available rows in the relations. The denominator on the other side is the square-root of $v1$ and $v2$ for each row and column.

Input: calculated results of cosine similarity and soft cosine similarity

Output: $allhybrid=hybridsim()$

1. $[r,c]=size(w2v);$
2. $allcossim=[];$
3. $allsoftcossim=[];$
4. $allhybrid=[];$
5. for $i=1 \rightarrow r$
6. $v1=w2v(i,:);$
7. $simvalue=0;$
8. $softvalue=0;$
9. $counter=0;$
10. for $j=i+1 \rightarrow r$
11. $v2=w2v(j,:);$
12. $simvalue=simvalue+Calculatecossim(v1,v2);$
13. $softvalue=softvalue+Calculatesoftcosine(v1,v2);$
14. $counter=counter+1;$
15. End for
16. $allcossim(i)=simvalue/counter;$
17. $allsoftcossim(i)=softvalue/counter;$
18. $allhybrid(i)=allcossim(i)+allsoftcossim(i);$
19. End for
20. End function

Algorithm 3 Hybrid Similarity

Algorithm 3 calculation is based on the results of cosine and soft-cosine similarity. To calculate firstly compute the size of the vector to find out the number of rows and columns. Next

RESEARCH ARTICLE

is to fetch the vectors $v1$ and $v2$ values from each word present in the rows using $w2v()$. Calculate similarity value by calling $Calculatecossim()$ on $v1$ and $v2$ values. Calculate Soft Cosine similarity value by calling $Calculatesoftcosine()$ on $v1$ and $v2$ values. And at last, calculate the hybrid similarity by adding the average of cosine and soft cosine similarity for each row.

In Figure 6 columns 1, 2, 3 represent cosine, soft-cosine, and hybrid similarity calculation outcomes respectively. After applying Cosine, Soft Cosine, and hybrid similarity measures, the similarity index graphical representation is as shown in Figure 7.

dtp	1	2	3
1	0.71144	1.9851	2.6965
2	0.58426	2.5277	3.112
3	0.54093	4.3554	4.8963
4	0.5818	2.1891	2.7709
5	0.73933	2.0626	2.8019
6	0.47753	4.0862	4.5637
7	0.69349	2.093	2.7865
8	0.68527	2.5232	3.2085
9	0.7215	2.121	2.8425
10	0.62197	1.7704	2.3924
11	0.63983	3.2016	3.8414
12	0.58562	2.1394	2.7251
13	0.3668	0.15274	0.51954
14	0.63537	3.2175	3.8529
15	0.7287	3.1002	3.8289
16	0.68648	2.8575	3.544
17	0.46367	2.0984	2.5621
18	0.64726	2.5198	3.167
19	0.6095	2.1303	2.7398
20	0.29064	0.15057	0.44121

Figure 6 Outcomes after Similarity Measures

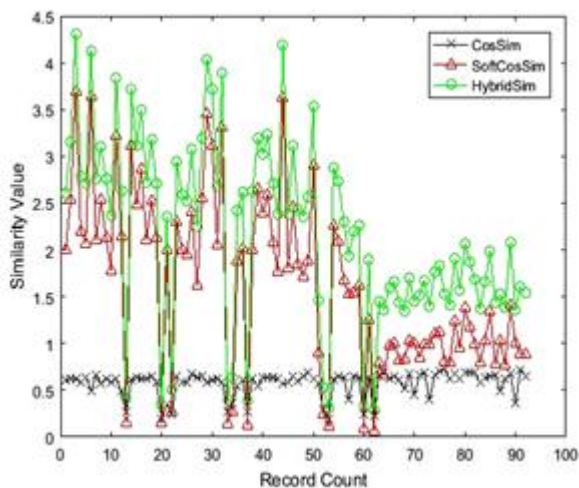


Figure 7 Similarity Index Graphical Representation

The resultant output for all tweets in the row is evaluated to obtain an average similarity index is then measured. The

formula used for cosine, soft cosine, and hybrid similarity index is represented by equations mentioned below:

$$\text{Cosine Similarity} = \sum_{i=1}^n \text{cosinesimilarity} / n \quad (1.1)$$

$$\text{Soft Cosine Similarity} = \sum_{i=1}^n \text{Softcosinesimilarity} / n \quad (1.2)$$

$$\text{Hybrid Similarity} = \text{Cosine} + \text{Soft Cosine} \quad (1.3)$$

$$\text{Set} = [\text{Cos}_i, \text{soft}_i, \text{hybrid}_i] \quad (1.4)$$

3.5. To Find Initial Centroid (IC) of Data

The next step is to determine, the Initial Centroid (IC) of the tweet data, which is obtained as the average of each similarity measure obtained from each similarity index (Cosine, soft cosine, and hybrid) individually. This value is taken as IC for the tweet. The formula used to determine the IC is given by equation (1.5).

$$\text{Initial Centroid (IC)} = \left[\sum_{i=1}^k \frac{\text{All Cosine}}{K}, \sum_{i=1}^k \frac{\text{All softCosine}}{K}, \sum_{i=1}^k \frac{\text{All hybrid}}{K} \right] \quad (1.5)$$

Where, k is the total number of tweets in a given document.

As shown in Figure 8, the centroid of each cluster is represented by the "X" sign. The available data are grouped into three clusters named cluster1, cluster2, and cluster3, each represented by different colors the blue, red, and orange colors respectively.

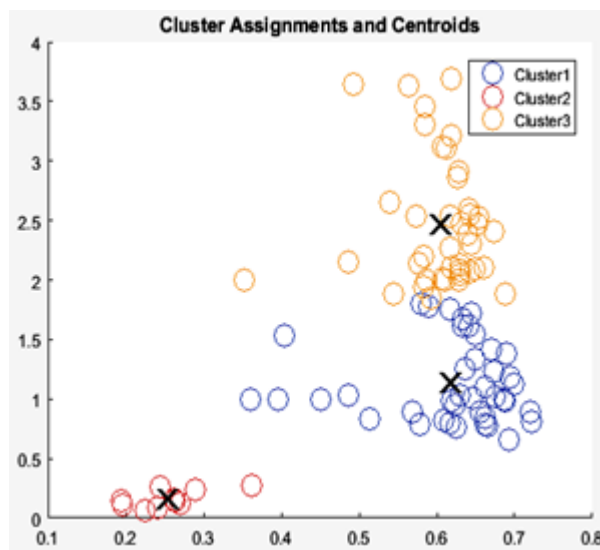


Figure 8 Cluster Assignments for Centroid

3.6. To Find Euclidian Distance of Each Tweet

Euclidean distance is the geometric distance in multidimensional space. The Euclidean distance between points T1 and T2 in n-dimensional space is calculated using the following formula (1.6). The formula used to calculate the

RESEARCH ARTICLE

Euclidian distance of each tweet from the set (st), which is calculated using equation (1.4) for cosine data, soft cosine, and hybrid data is represented by equations shown below:

For all St in sets calculate,

$$D1 = ECU1(IC, ST) \tag{1.6}$$

$$D2 = Squared ECU1 (IC, ST) \tag{1.7}$$

$$D3 = (D1 + D2)/2 \tag{1.8}$$

Note that the Euclidean distance (and its square) is calculated from the Tweet data obtained from the previous step.

After this, to determine the number of fragments, the total number of centroids in the given data has been calculated by measuring the average of D1, D2, and D3. i.e.

D=Total number of centroids

$$Calculate D = \frac{D1+D2+D3}{3} \tag{1.9}$$

An example of distance measured from st that is d1 is represented by Figure 9.

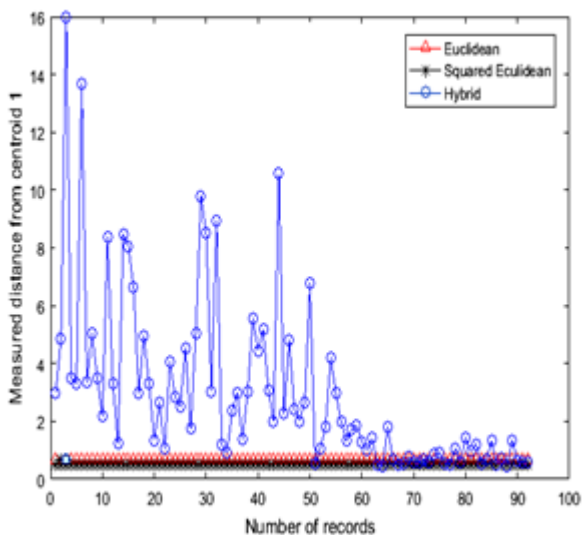


Figure 9 Measured Distances from Centroid 1

3.7. Fragmentation

Data fragmentation refers to dividing the data into segments so that the storage becomes easy. To determine the number of fragments from the available data, the formula used is written by equation (1.10).

$$p = \frac{\sqrt{D \times K}}{c} \tag{1.10}$$

Where K is the total number of rows.

3.8. Neural Network

The data obtained after fragmentation is passed along with the original word 2 vector data as input to the Artificial Neural Network (ANN). The used structure of ANN is given below (see Figure 10).

The classified fragments for 100, 200, 300, 400, and 500 rows are listed in Table 1 below.

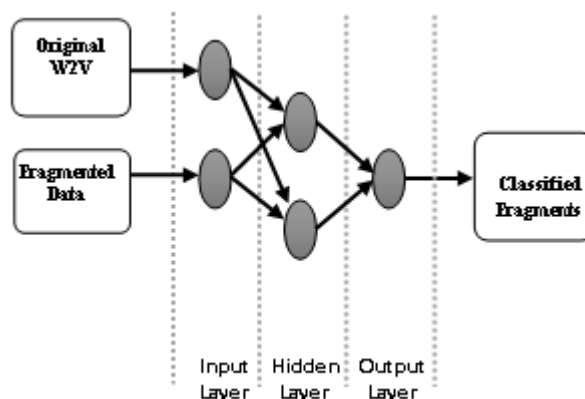


Figure 10 ANN Structure to Classify Fragmented Data

For 100 Rows	For 200 Rows	For 300 Rows	For 400 Rows	For 500 Rows
1	1	2	4	3
1	1	2	2	2
2	1	1	1	1
1	2	3	3	2
1	2	1	1	3
1	1	2	2	2
2	1	1	2	4
1	3	2	3	4
1	2	3	3	3
3	1	4	4	3
2	2	3	2	2

Table 1 Classification of Fragments for Each Row

4. RESULTS AND DISCUSSIONS

Fragmented architecture has been designed using MATLAB simulator with 4 GB RAM, 64-bit operating system, and a processor of 2.30 GHz. The performance has been analyzed in terms of the classified accuracy. The results have been evaluated individually, for 100, 200, 300, 400, and 500 rows. Experiments have been performed five times to determine the detection accuracy as depicted in Table 2. Figure 11 represents the classification accuracy of the designed fragmented structure. The simulations have been performed five times so that the exact accuracy for the uploaded data that might contain rows (100, 200, 300, 400, and 500). From the figure, it is observed that with the increase in rows, the classification accuracy increases. This is because with the

RESEARCH ARTICLE

increase in the data the ability to train ANN structure is increases that result in improved classification of the fragmented data. The average of the classification accuracy obtained for 100, 200, 300, 400, and 500 rows are 63.81, 76.28, 81.52, 83.58, and 92.078 respectively.

Iterations	100	200	300	400	500
1	62.45	75.89	82.15	84.57	91.04
2	63.57	76.28	81.27	83.57	92.57
3	62.78	76.18	79.68	84.25	93.17
4	64.28	75.12	80.15	82.37	92.14
5	65.97	77.94	84.36	83.14	91.47

Table 2 Classification Accuracy for Different Rows

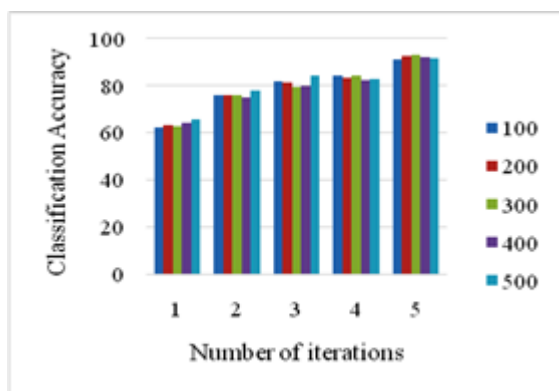


Figure 11 Classification Accuracy

To evaluate precision and recall average percentage of iterations is calculated as shown in Table 3. The fragments architecture created after the evaluation of the proposed work is further evaluated by parameters such as True Positive, True Negative, False Negative, and False Positive are shown in Table 4. Precision and Recall are shown in Table 5.

True Positive (Tp) =
 $\frac{\text{Total true selected elements}}{\text{Total sample size}}$ (1.11)

False Positive (Fp) =
 $\frac{\text{False selected elements}}{\text{Total sample size}}$ (1.12)

True Negative (Tn) =
 $\frac{\text{True left samples}}{\text{Total Sample Size}}$ (1.13)

False Negative (Fn) =
 $\frac{\text{False left sample}}{\text{Total Sample Size}}$ (1.14)

Table 5 showing the precision and recall and found that they are almost the same for every row passed and Figure 12 shows the graphical representation of the precision and recall. The Fp value reveals that the components are not put in fitting clusters. The Fp outcomes in this work are lower. Tn showing

bad samples that are left. If it is high it indicates a good search response. Depends on the value calculated for T_p, T_n, F_p, and F_n precision and recall values are calculated.

No. of Rows/Records	Average of all five iteration in % (approx)	Remaining average (%)
100	64	36
200	76	24
300	82	18
400	84	16
500	92	8

Table 3 Average Percentage of Iterations

Rows	100	200	300	400	500
Total true selected samples (T _p)	0.45	0.53	0.57	0.59	0.64
True left samples (T _n)	0.25	0.17	0.13	0.11	0.11
False left samples (F _n)	0.44	0.53	0.57	0.58	0.64
false selected sample (F _p)	0.25	0.08	0.04	0.03	0.01

Table 4 Evaluation of T_p, F_p, T_n, and F_n

Total Passed Rows	100	200	300	400	500
Precision	0.64	0.87	0.93	0.95	0.98
Recall	0.64	0.76	0.82	0.84	0.85

Table 5 Evaluation of Precision and Recall

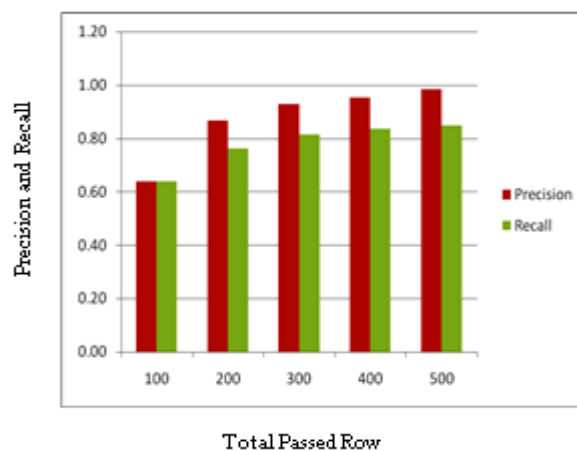


Figure 12 Precision and Recall

RESEARCH ARTICLE

5. COMPARATIVE ANALYSIS OF PROPOSED AND EXISTING WORK

The maximum attained precision is 0.98 for total passed rows of 500. The recall value for every row passed is 0.85. It is found that an enhancement in proposed work in the case of precision and recall is seen. Earlier researchers used the k-means dependent cosine similarity measurement method to determine the feature similarity between the cluster centroids and the data points to quantify the similarity between the outcome of the clustering and the side details. A clustering algorithm for data partitioning using the principle of a learning method has been proposed. The main drawback of this proposed work is that only cosine similarity based k-means have been used for partitioning large data sets [33]. An effort on the hybridization of cosine and soft-cosine is also carried out to improve precision and recall parameters during partitioning [34]. In this research, we have proposed cosine, soft cosine, and hybrid similarity as an enhanced mechanism and achieved an increase of 0.98 precision and 0.85 recall values as outcome. So, we rely on this technique for fragmenting the text data for a diverse system. Table 6 shows the comparison of calculated precision and recall. Figure 13 shows the Graphical Views of Precision and Recall

Parameters	Huaping Guo et al. [33]	Kiranjeet Kaur et al. [34]	Proposed Work
Precision	0.47	0.69	0.98
Recall	0.54	0.67	0.85

Table 6 Comparison of calculated Precision and Recall

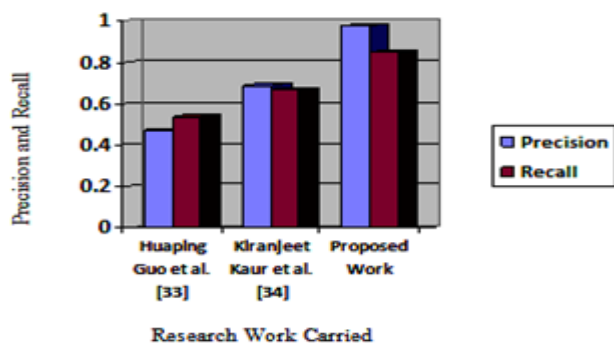


Figure 13 Graphical Views of Precision and Recall

Proposed works has many advantages over existing methodologies and are following as:

- It helps to improve the classification of fragmented data with high precision and recall and indicates maximum coverage, accuracy, and reduce overall computational time.

- Earlier techniques depends on cosine based k-means, cosine and soft cosine hybridization for clustering were not effective due to lack of balance in the quality and efficiency of clustering in categorical data sets.
- The principles of existing approaches are fine for some case, but not applicable. So algorithm hybridization is the best approach. Cosine and soft cosine similarity notions are used to compute hybrid similarity in this research work to ensure improved efficiency and can easily dealing with large data sets.

6. CONCLUSION

In this paper, novel relative data fragmentation architecture is proposed to divide the large dataset into different fragments. Here, twitter data is being applied for experiment purposes and converted into vectors to use it for fragmentation purposes. Cosine, soft cosine and hybrid similarity calculation is calculated and centroid positions are discovered. K-Mean algorithm is used to calculate the distance between data points with each centroid to discover clusters. At last, validation and performance is performed by checking its accuracy using ANN. In this research, efforts are given to introduce novel similarity based data fragmentation architecture in an unsupervised learning environment. Comparison is performed and attained high precision and recall as compared to the existing proposed methods.

Future work is stress on the allocation and deduplication of data to strengthen the wide distributed network environment. So, that most of the user’s requirements can be satisfied and also work for the enhancement of various parameters such as cost, delay factors, duplication of data.

REFERENCES

- [1] Tarun S., Batth R. S. (2019). Distributed Database Design Challenges and its Countermeasures-A Study. Journal of the Gujarat Research Society 21 (6), pp. 875-886
- [2] S. Tarun, R. S. Batth and S. Kaur, "A Review on Fragmentation, Allocation and Replication in Distributed Database Systems," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2019, pp. 538-544, doi: 10.1109/ICCIKE47802.2019.9004233
- [3] R. Singh and K. S. Mann, "Improved TDMA Protocol for Channel Sensing in Vehicular Ad Hoc Network Using Time Lay," Proceedings of 2nd International Conference on Communication, Computing and Networking Lecture Notes in Networks and Systems, pp. 303-311, 2018.
- [4] A. Nayar, R. S. Batth, D. B. Ha, and G. Sussendran, G. "Opportunistic networks: Present scenario-A mirror review" International Journal of Communication Networks and Information Security," 10 (1), pp. 223-241, 2018.
- [5] G.S Shahi, R.S Batth, S. Egerton, 2020 "MRGM: An Adaptive Mechanism for Congestion Control in Smart Vehicular Network", International Journal of Communication Networks and Information Security 12 (2).
- [6] Qi, H., & Gani, A. (2012, May). Research on mobile cloud computing: Review, trend and perspectives. In 2012 Second International

RESEARCH ARTICLE

Conference on Digital Information and Communication Technology and its Applications (DICTAP), IEEE, pp. 195-202.

[7] Venters, W., & Whitley, E. A. (2012). A critical review of cloud computing: researching desires and realities. *Journal of Information Technology*, 27(3), pp. 179-197.

[8] Borkar, V., Deshmukh, K., & Sarawagi, S. (2001, May). Automatic Segmentation of text into structured records. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pp. 175-186.

[9] Santini, S., & Jain, R. (1999). Similarity measures. *IEEE Transactions on pattern analysis and machine Intelligence*, 21(9), pp. 871-883.

[10] Huang, A. (2008, April). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand ,Vol. 4, pp. 9-56.

[11] Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3), pp. 491-504.

[12] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning. *Neural and Statistical Classification*, 13(1994), pp. 1-298.

[13] Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8), pp. 966-974.

[14] Verma and A. Kumar, "Performance Enhancement of K-Means Clustering Algorithms for High Dimensional Data sets", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 4, No. 1, pp.5-9, 2014.

[15] Z.Tao, H. Liu, H. Fu and Y.Fu, "Image Co-segmentation via Saliency-Guided Constrained Clustering with Cosine Similarity", *AAAI*, pp. 4285-4291, 2017

[16] X. Gu, H. Zhang and S. Kim, "Deep code search", In *Proceedings of the 40th International Conference on Software Engineering*, ACM, pp. 933-944, 2018.

[17] W. L. Xiang, Y. Z. Li, R. C. He, M.X. Gao, M.Q. An, "A novel artificial bee colony algorithm based on the cosine similarity", *Computers & Industrial Engineering*, Vol. 115, pp.54-68, 2018.

[18] Wiese, L. (2014). Clustering-based fragmentation and data replication for flexible query answering in distributed databases. *Journal of Cloud Computing* 3, 18. <https://doi.org/10.1186/s13677-014-0018-0>

[19] Ali A. Amer, Adel A. Sewisy, Taha M.A. Elgandy. (2017). An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs). *Heliyon* 3 e00487. doi: 10.1016/j.heliyon.2017. e00487

[20] Abdalla, H., &Artoli, A. M. (2019). Towards an efficient data fragmentation, allocation, and clustering approach in a distributed environment. *Information*, 10(3), 112. <https://doi.org/10.3390/info10030112>

[21] Rahimi, H., Parand, F. A., & Riahi, D. (2018). Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases. *Applied computing and informatics*, 14(2), pp. 127-133. <https://doi.org/10.1016/j.aci.2015.03.001>

[22] Lim, S., Ng, Y. (2001). A Hybrid Fragmentation Approach for Distributed Deductive Database Systems. *Knowledge and Information Systems* 3, pp. 198–224. <https://doi.org/10.1007/PL00011666>

[23] Khan S. I., (2016). Efficient Partitioning of Large Databases without Query Statistics", *Database System Journal*, pp. 34-53.

[24] Peng, P., Zou, L., Chen, L., & Zhao, D. (2019). Adaptive distributed RDF graph fragmentation and allocation based on query workload. *IEEE Transactions on Knowledge and Data Engineering*, 31(4), pp.670-685. <https://doi.org/10.1109/TKDE.2018.2841389>

[25] Aloini, D., Benevento, E., Stefanini, A., & Zerbino, P. (2020). Process fragmentation and port performance: Merging SNA and text mining. *International Journal of Information Management*, 51, 101925. <https://doi.org/10.1016/j.ijinfomgt.2019.03.012>

[26] Memmi, G., Kapusta, K., & Qiu, H. (2015, August). Data protection: Combining fragmentation, encryption, and dispersion. In *2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)* (pp.1-9). IEEE. <https://doi.org/10.1109/SSIC.2015.7245680>

[27] Links: <https://www.kaggle.com/soaxelbrooke/first-inbound-and-response-tweets/data?select=sample.csv>

[28] Links: <https://gist.github.com/larsyencken/1440509>

[29] Lende, S. P., &Raghuwanshi, M. M. (2016, February). Question answering system on education acts using NLP techniques. In *2016 world conference on futuristic trends in research and innovation for social welfare (Startup Conclave)* (pp. 1-6). IEEE.

[30] Zeyu, X., Qiangqian, S., Yijie, W., & Chenyang, Z. (2018). Paragraph vector representation based on word to vector and CNN learning. *Computers, Materials & Continua*, 55(2), pp. 213-227.

[31] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[32] Bartunov, S., Kondrashkin, D., Osokin, A., &Vetrov, D. (2016, May). Breaking sticks and ambiguities with adaptive skip-gram. In *artificial intelligence and statistics*, pp. 130-138.

[33] H. Guo, J. Zhou and C.A. Wu (2018), "Imbalanced Learning Based on Data-Partition and SMOTE", *Information*, Vol. 9, No. 9, pp. 238.

[34] Kaur K., Laxmi V. (2019), "Hierarchical Clustering Based Improved Data Partitioning using Hybrid Similarity Measurement Approach", *International Journal of Innovative Technology and Exploring Engineering*, Volume-8 Issue-8, pp. 3008-2014.

Authors

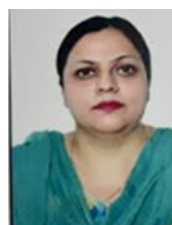


Mr. Sashi Tarun is a PhD. Research Scholar in the School of Computer Science And Engineering at Lovely Professional University, Punjab, India. He has completed M.Tech. Computer Science from Jamia Hamdard University, New Delhi. His research interests are Distributed Systems, Cloud Systems, Database System, Computer Networks, AI, and Machine Learning. He has number of papers in his credit. He has 7 years of teaching experience as

Assistant Professor.



Dr. Ranbir Singh Batth is working as an Associate Professor in the School of Computer Science and Engineering and he also serves as an International coordinator for at Lovely Professional University, Punjab, India. He has received his Ph.D. from IKG Punjab Technical University, Kapurthala, Punjab, India in 2018 and the Master degree in Computer Engineering from Punjabi University, Patiala. His research interests include Wireless Sensor Networks, Cloud Computing, Network Security, Ad Hoc Networks, IoT, Machine Learning, Deep Learning, Wireless Communications and Mobile computing. He also serves as an editorial member, guest editor, and reviewer for various reputed International journals. He has been the organizing chair, session chair and advisory member for various reputed International conferences. He is an active member of ACM and IEEE computer Society.



Dr. Sukhpreet Kaur is working as Associate Professor in CSE department at Chandigarh Engineering College, Landran, Mohali. She has in total of 15 years of vast experience in teaching and research. She has done Ph.D in CSE from I K Gujral Punjab Technical University, Jalandhar and has done her Masters in Technology in CSE from GNDEC, Ludhiana. The various research areas in which she worked includes Image Processing, Artificial Intelligence and Computer Vision.

A Scheme for Data Deduplication Using Advance Machine Learning Architecture in Distributed Systems

Sashi Tarun¹, ,
Research Scholar, School of CSE,
Lovely Professional University,
Phagwara, India
Sashitarun79@gmail.com

Ranbir Singh Bath²
School of CSE,
Lovely Professional University,
Phagwara, India
ranbir.21123@lpu.co.in

Sukhpreet Kaur³
Department of CSE,
Chandigarh Engineering College,
Chandigarh, India
Sukhpreet.4479@cgc.edu.in

Abstract-- In a distributed architecture, data as a resource has its own value, but continuous integration of large amounts of data across several locations without cross-verification to preserve a single instance data pattern appears impossible. As a result, systems have encountered hurdles that have a direct influence on the efficiency and performance of distributed workforces. Users need high-quality data or information in order to continue working as improved data services in order to find future trends. However, duplicate data entries in storage repositories are considered a major flaw or stumbling block in the data analysis and query operations processes. As a result, businesses have invested significant resources in detecting duplicate data throughout the duplicate entry detection process. We've introduced a cutting-edge machine learning framework for detecting duplicate data on both current and new data entries. Textual data inputs or queries are imported into memory, preprocessed, and transformed to a vector space model using this technique. To arrange data in groups with equal capacity, a clustering K-means approach is used. To save time and money during the detection phase, similarity computations were done cluster-by-cluster rather than on a huge dataset. The suggested technique performs better than existing deduplication algorithms, with an optimal accuracy of 99.7%. If the result-test and gt-test outcomes are determined to be same during comparison, the accuracy performance parameter of the deduplication process is greater.

Keywords-- machine learning, vector space model, k-means, computational cost, similarity, deduplication, performance

I. INTRODUCTION

The distributed environment meets business needs by allowing individuals to regularly store, transfer, and secure their data with reducing operational costs. In the domain of dispersed data, researchers have worked on [1-3]. Data quality problems have been discovered as a result of companies' efforts to expand storage capacity at dispersed sites. When greater focus on data consumption includes digital social media contents, transactional data, archived data, and

regular data backups, all of which have emerged in the form of data problems, workers are responsible for creating data-related concerns. It decreases distributed system efficiency by increasing data storage capacity (due to redundancy). It degrades data quality by slowing query replies, raising computing costs, wasting surplus bandwidth during data transport, and slowing query responses. Such roadblocks reduce the efficiency of distributed workforces and put data analysts and data engines in danger of not being able to extract data from dispersed enormous data volumes in order to fulfil their tasks. Prior to distributing data to scattered sites, it is necessary to identify duplicate data as shown in Fig. 1.

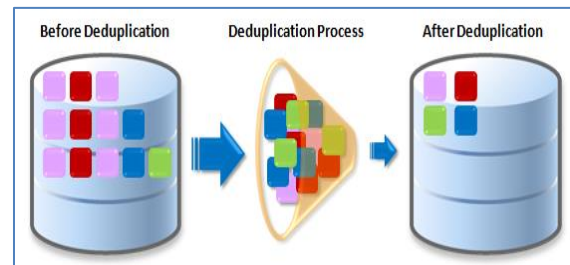


Fig. 1. Deduplication Process

The researchers have proposed a variety of data deduplication techniques for distributed data structures. To address existing challenges, several strategies were proposed, including attribute-based deduplication, attribute and role-based deduplication, ANN, clustering algorithms, hash indexing, MapReduce and HDFS, bucket-based deduplication, Finger print clustering, Sampling method, Token generation techniques, and clock level deduplication. Data uniqueness helps us to make our data error-free, consistent, and non-ambiguous. Data redundancy is produced by combining data from several sources in an overlapping manner, which creates an incorrect scenario for people who believe in performance characteristics such as data reliability, accuracy, and

consistency. As a result, frequent data requests must be checked first before being allowed to write to existing data storage. It only allows non-conflict data that would otherwise be considered duplicate data.

It was discovered that businesses spend a significant amount of money safeguarding their data by doing periodic data backups and keeping it on remote servers. It creates data difficulties including increased data duplication ratios, delays in discovering and retrieving data, confusing data issues, and other hidden expenses. Data deduplication is a method of identifying duplicate data in storage units. The proposed approach detects duplicate data copies to save time and money while uploading and transferring data across a dispersed network.

For the deduplication process, an advanced machine learning architecture is proposed in this work. The “user’s sentimental analysis data” and “tweets” datasets, each with 1048576 data instances, are deployed for the experiment. Columns 6 and 4 are utilized as *rawdata* or inputs in the deduplication process, with 2000x20 and 2001x30 records/instances, respectively, which are then used for text pre-processing or refinement by removing stop words to get *filteredwords*. Each word in the *filterdata* is transformed into a vector space model and assigned a value based on the sum of the ASCII values for each character. Data clustering is a technique for categorizing data into groups in order to save computing costs and effectively discover duplicate data. The work on performance parameters has been covered in order to show the scheme’s efficacy, and the results generated are based on the result test and gt test list values and represented in boolean terms.

The remainder of the paper is organized as follows: The second section highlights current work on deduplication methods. The proposed technique was explained in Section 3. The explanation of simulation and calculative findings obtained using the proposed advanced machine learning architecture is covered in Section 4. The fifth section depicts a comparison of proposed and existing approaches. The paper work is concluded at Section 6.

II. RELATED WORK AND STUDIES

To detect substantial redundancy, content-defined chunking (CDC) has been employed in data deduplication. They optimized the deduplication system by tweaking relevant parameters in CDC to determine chunk cut-points and provided an efficient fingerprint using a novel hash function in the

proposed study. They developed a new low-cost hashing function and a novel byte’s frequency-based chunking (BFBC) approach in [1]. To tackle data duplication issues, works on new techniques are on continuous development track. Earlier work had contributed a lot to achieve performance in distributed model. A service named KGTMaaAS Key Generation and Token Maintenance was designed to manage cloud storage and minimize duplicated data. It can detect duplicate data at both the block and file level in [2]. A framework for role and attribute-based de-duplication was proposed in the cloud that supports diverse content types. De-duplication has been done on a role-by-role basis. That is, distinct material should be kept for each role type, or de-duplication should be performed. It will handle a variety of content types, including text, photos, and video. Users with the same function may have several versions of the same content that they share or trade. These can be text, photos, or any other sort of data, hence de-duplication is necessary at the role level rather than at the system level to decrease memory and storage requirements in [3]. To work on the contents of large data sets, a novel de-duplication methodology has been developed. Different dictionary indexing approaches will be utilized to de-duplicate the field’s contents that have bounded variability. In addition, for fields with lengthy strings, a set of computationally low-cost hash algorithms will be utilized to speed up deduplication in [4]. It was found that to detect data duplicity string to string comparison results is less effective identification architecture. For the removal of redundant data stored on the disk fingerprint index detection and prefetching techniques. It was based on reinforcement learning and helped to remove 2% to 10% duplicity in comparison to sparse indexing method in [5]. Data deduplication work was proposed to reduce the duplicate data in scale-out distributed storage systems. To achieve this more scalable and efficient deduplication process mechanism as SEP-D for locating metadata information using content-based hashing to reduce I/O operations and CRUSH for the application of placement strategies was presented in [7]. The sub-file-level chunking deduplication system is an example of system deduplication since it divides the incoming data stream into several data “chunks” and uses comparing methods to find duplicates. De-duplication systems delete duplicate chunks, allowing only one copy of these chunks to be stored or sent in order to achieve the storage space or network bandwidth savings goal. It’s worth noting that de-duplication systems have a long runtime and require a lot of CPU resources to function in [8]. The genetic evolution-based data deduplication methodology has a greater

deduplication ratio than existing systems, allowing it to detect duplicated data even quicker. The divisor D values in prior methodologies are static; however, in global optimization using a genetic algorithm, we dynamically choose the optimum divisor D value to uncover maximal redundancy. Bucket indexing based on genetic evolution deduplication is 16 times quicker than Rabin CDC, 5 times quicker than AE, and 1.6 times quicker than Fast CDC in [10]. In Bigdata storage, large amount of data is stored in digital form without using a structured approach. So, to identify duplicate data and create unstructured data into structured data a new technique was presented to increase the efficiency of Bigdata storage. Based on hash values and MD5 methods a bucket-based deduplication system with a fixed-size chunking level is proposed. To execute this strategy, the researchers employed the Destor open-source application and HDFS (Hadoop Distributed File System) in [11]. To achieve high data deduplication rates and minimal overhead communication while maintaining load balancing the AR-Dedup cluster deduplication system was proposed. In their programme, they adopt an application-aware method to deduplication. AR-Dedup makes use of cluster deduplication routing in [12]. To discover duplicate data, an artificial neural network deduplication methodology was presented. The suggested system takes as input a set of data created by various similarity measurements. The ANN is used to process the model parameters that have been computed from similarity functions i.e. dice coefficient, damerau-lavenshtein distance, and tversky index. The ANN has two stages, one for training and the other for testing data used in [13]. To achieve maximum disc storage savings, it spreads data blocks over numerous storage nodes and performs rapid compression following data deduplication. Droplet improves disc IO by combining write requests for tiny data blocks into big chunks. Droplet has demonstrated great deduplication performance in tests. Droplet is a high-throughput and scalability distributed deduplication storage solution in [16]. An in-line data deduplication distributed architecture follows an intelligent storage balancing strategy was proposed to use storage nodes for storing data with a view to enhance deduplication efficiency. This architecture was capable to perform deduplication with high throughput and ratio and found affective with comparison to perform deduplication on a single system. A technique was proposed called sampled hashing to strengthen the scalability of the distributed architecture in [17].

III. PROPOSED METHODOLOGY

String based data mining is one of the most traditional approach in the history of computer

architecture. Data matching through string comparison is one of the most ambiguous architectures, as a small variation in the string architecture will lead to a mismatch in the string phase. In addition to this, de-duplication is a relative process as there is no ground truth available for the processing. The relative analysis is best attainable when vector-based learning module is applied. To handle deduplication an advance machine learning architecture is proposed to identify/detect duplicate data as shown in Fig. 2.

Step-by-step implementation of proposed methodology is performed in Matlab 2016a. It follows following steps:

- Start uploading of stored data/queries.
- Apply text pre-processing by removal of stop words from the dataset.
- Translate textual data into vectors.
- Finding indexes (indicate no. of possible clusters) and centroids positions (centre point of each cluster) using k-means for categorizing existing or new data into clusters.
- Generation of training and testing indexes.
- Calculation of similarity index cluster-wise.
- Apply ground truth (gt) test to check duplicate data in testing data. Assign label 1 for duplicate data entry otherwise put label 0.
- Calculate similarity test on testing data.
- The results of similarity tests are compared to the similarity of each cluster according to set threshold values in order to discover duplicate figures.
- Calculation of sensitivity, specificity, accuracy, and fmeasure performance parameters to evaluate proposed deduplication scheme efficacy.

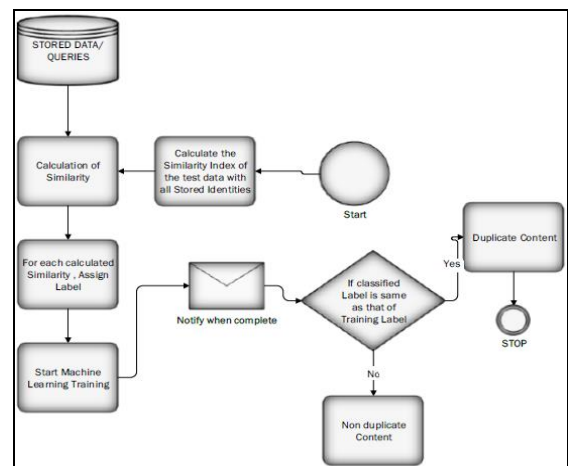


Fig. 2. Proposed Methodology work flow

A. Text Pre-Processing

It refers to process involves removal of common words from the dataset in textual context using stop word lists which contains articles, prepositions, and nouns etc. It is necessary to reduce dimension of available dataset text-records by eliminating stop words. The reason for this is that it provides no valuable information. These stop words can be accessible from the stop list in [19]. To apply preprocessing, sixth column from dataset in [18] and fourth column from dataset in [20] are used as data-inputs (*rawdata*) as specified below in code lines:

```

1. Datacol = 6;
2. for i=1:r
3.   rawdata{i,1}=data{i,datacol};
4. end

```

Data-input queries (*rawdata*) are selected row-wise from the storage at every iteration and set as *currentdata* which is further interpreted by *getwords()* method. *getwords()* method is used to obtain individual words from *currentdata* to stored them in the array-list of allwords. Such words are compared with stop word lists and *filteredwords* list is generated. Filteredwords containing suppressed words obtained after preprocessing step.

1) Pseudocode for Text Pre-Processing

Input: *rawdata*

Output: *filteredwords*

```

1. Load rawdata as input   % Select text column row-wise
2. [r,c]= size(rawdata);
3. for i=1:r
4.   currentdata=rawdata{i,1};
5.   words=getwords(currentdata);
6.   allwords{i}=words;
7.   filteredwords=filterdata(words,stopwords);
8. end

```

B. Word to Vector (W2V) Translation

Similarity is commonly measured by comparing how similar or similar two objects are. An approach of calculating similarity is to use vectors. A vector is essentially a number with both magnitude and direction. It takes a large text corpus as input (tweets with stop words deleted) and allocates a vector to each word in the tweet. Such vectors are represented in rows and columns depending on size of attributes or filteredwords, which finally used for clustering the data. Each attribute or filteredwords are replaced by a value, which is the sum-up of ASCII value of each character in a word.

1) Pseudocode for W2V Translation

Input: *filteredwords*

Output: *w2v*

```

1. for wd=1: numel(filteredwords)
2.   w2v(i,wd)=sum(double(filteredwords{wd}));
3. End

```

C. Categorizing Data into Clusters

For deduplication, textual data samples are converted into vector data model which further applied for creating clusters. These datasets are having data instances used for training machine learning model. Datasets are partitioned into three clusters having their own data instances as shown below in Table I.

TABLE I. DISTRIBUTION OF DATASET CLUSTER WISE

Dataset	Dataset-Size Used	Clusters Indexes	Data-Instances	
			Rows	Cols.
Sentiment Analysis with Tweets [18]	2000x20	1	724	20
		2	701	20
		3	575	20
Tweets-about-the-top-companies-from-2015-to-2020 [20]	2001x30	1	35	30
		2	1935	30
		3	31	30

Clustering is an approach helps to find groups on unlabeled data in machine learning. It helps to examine and categorize new data patterns into their associated groups. It maintains intra-cluster similarity and helps to identify duplicate data contents at cluster level during deduplication process. Group-data analysis enhances similarity measure results rather than applying on the large dataset values. Data analysis interpretation in groups also improves the computational cost.

In the process of making data clusters, dataset is categorized into indexes and centroids to begin with. K-means follows nearest neighbor strategy to allocate data in cluster category and estimate centroid position for each cluster based on vector data model. These centroid positions helps to categorize vector data during cluster identification process. So, deduplication process directly applied at cluster level rather than applying comparison on dataset values.

D. Generation of Training and Testing Indexes

To evaluate machine learning model dataset is split into training and testing data. A ratio of 80% and 20% is used for training and testing data. To generate training index (*tindex*) and testing index (*ti*) random generated integer is selected in each iteration till not equivalent to training and testing count. Such generated indexes are maintained in *training_indexes* and *test_index* lists that is used further to extract data-rows. These indexes are further used for similarity calculations based on vector values of data rows. To estimate training_count and testing_count, equations (i) and (ii) are used as given below:

```
training_count = round(training_rate/100*rrc)      (i)
test_count = rrc-numel(training_indexes)          (ii)
```

1) Pseudocode for Training and Testing Index

Input: training_count, rrc, test_count

Output: training_indexes, test_index

```
1. for kki=1:training_count
2.   tindex=round(rrc*rand);
3.   if tindex==0
4.     tindex=1;
5.   end
6.   training_indexes(kki)=tindex; % generate training index
7. end
8. for kkm=1:test_count
9.   ti=round(rrc*rand);
10.  if ti==0
11.    ti=1;
12.  end
13.  test_index(kkm)=ti; %generating testing indexes
14. end
```

E. Cluster-Wise Similarity Calculation

To detect duplicate data records, a similarity computation is performed individually on each cluster. It takes a long time to go through each data point in the cluster and calculate the similarity index. Rather of focusing on individual indexes, we construct cluster indexes to save time. To save computing time, a random evaluation approach is used to build cluster indexes. To describe each cluster index, ten randomly applied indices are chosen individually. These validations are a set of randomly chosen data row values that are used to determine cosine similarity with the test data. During the computation, the prior result of the similarity value is kept and added to the most recent result. This method is repeated according to the characterization set-value (which represents row attributes) and validation set-value (which represents random data values). Cluster-wise similarity is used to detect duplicate data received as testing data or fresh data entries to put in data storage as a whole. As a result, all data contents that fall within the cluster similarity index's range might be considered duplicate material. Cluster wise similarity value results are shown below in Table II.

1) Pseudocode for Cluster-Wise Similarity

Input: rowvalue, test_index

Output: simvalue=calculatecossim(v1,v2)

```
1. dfa=[];
2. for i=1:3
3.   dtcluster=[];
4.   dtcluster=w2v1(ind1==i, :);
5.   smvalue=0;
6.   [rcl,cls]=size(dtcluster);
7.   total_no_of_validations=10;
8.   total_characterization=10;
9.   counter=0;
10.  for dd=1:total_characterization
11.    for kd=1:total_no_of_validations
12.      rowvalue=round(rcl*rand);
13.      if rowvalue==0
14.        rowvalue=1;
```

```
15. end
16. counter=counter+1;
17. dtver(counter,:)=dtcluster(rowvalue,:);
18. end
19. end
20. datatovaildate=w2v1(test_index,:);
21. try
22.   smvalue=smvalue+calculatecossim(dtver(1:numel(test_index,:),datatovaildate);
23. catch
24.   smvalue = smvalue + calculatecossim(datatovaildate, dtver);
25. end
26. dfa(i)=smvalue/numel(test_index); % clusters wise similarity index
27. end
```

TABLE II. CLUSTER-WISE SIMILARITY VALUES

Datasets References	Cluster Indexes	Similarity Value (Cluster-wise)
[18]	1	0.0012
	2	0.0009
	3	0.0014
[20]	1	0.2222
	2	0.2077
	3	0.2579

1) Pseudocode for Cosine Similarity

Input: test_index, row_value

Output: simvalue = simvalue + calculatecossim(v1,v2)

```
1. nume=0; %numerator
2. deno=0; %denominator
3. for i=1:numel(v1)
4.   try
5.     nume=nume+v1(i)*v2(i);
6.   catch
7.   end
8.   end
9.   deno1=0;
10.  deno2=0;
11.  for j=1:numel(v1)
12.    try
13.      deno1=deno1+v1(j)^2;
14.      deno2=deno2+v2(j)^2;
15.    catch
16.    end
17.    end
18.  deno=sqrt(deno1)*sqrt(deno2);
19.  simvalue=nume/deno;
20. end
```

F. Generation of Ground Truth Test (gt_test)

To detect duplicate data on testing data gt_test is applied. To achieve this, test index and training indexes having data-size of 1x400 and 1x1600 dimension are used. If vectors on training indexes is matched and found identical with vectors of test indexes than records are identified as duplicate, otherwise not. During comparison gt_test elements list is update to 1 (for matching found) or otherwise 0 (for matching not found). This procedure is repeated till all elements on the training data is not compared

with test data as a whole and *gt_test* list of elements is populated.

1) Pseudocode for Ground Truth Test (*gt_test*)

Input: training_indexes, test_index

Output: *gt_test*

```

1. w2v_test=[];
2. w2v_test=w2v1(test_index,:);
3. [rtest,ctest]=size(w2v_test)
4. gt_test=[];
5. gt_test=zeros(1,numel(test_index));
6. for kdi=1:numel(gt_test)
7.   ssd=[];
8.   ssd=find(training_indexes==test_index(kdi));
9.   if ~isempty(ssd)
10.    gt_test(kdi)=1;
11.  end
12. end

```

G. Similarity Test on Testing Data

In this section, similarity test is carried out on test data and test value. Test values are testing data having size of 400x20 vector values, and test data is based on randomly generated test indexes. The set value for *test_indexes* is depicted at line 5 in pseudocode in this section. Here, each test value of testing data is compared with randomly generated test data to compute cosine similarity test to reduce computational cost. Similarity test is estimated by summing up of previous computed *simtest* results fraction by *test_indexes*.

1) Pseudocode for Test-Data Similarity Test

Input: test_data, test_value

Output: *sim_test*

```

1. simtest=0;
2. [rtest,ctest]=size(w2v_test)
3. for ii=1:rtest
4.   test_value=w2v_test(ii,:);
5.   test_indexes=5; % generations of test index
6.   for kk=1:test_indexes
7.     test_index=round(rtest*rand);
8.     if test_index==0
9.       test_index=1;
10.    end
11.   test_data=w2v(test_index,:);
12.   simtest=simtest+calculatedcossim(test_data,test_value);
13. end
14. simtest=simtest/test_indexes;

```

H. Comparison of Similarity-Test Results with Clusters-Similarity to Detect Duplication

After computation of similarity test, threshold limits are defined by adding and decreasing 10% on cluster similarity results. The threshold range limit is defined by setting up threshold as defined in Pseudocode lines 4-5. Decision boundary (threshold limit) is compared with similarity test (*simtest*) to confirm about duplicate (1) or otherwise (0) in testing data. Each *test_value* (400x1) of testing data is compared with threshold limits equivalent to numbers of cluster similarity index. If similarity test

results are found close or within threshold limits, such input is categorized as duplicate data and updating on *result_set* list with duplicate (1) for each comparison done. Sometimes, data is not permitted to house in existing storage because duplicate data entries may be restricted or monitored.

1) Pseudocode to Compare Cluster Similarity and Similarity Test Results

Input: dfa, simtest, duplicate

Output: *result_test*

```

1. result_test=[];
2. duplicate=0;
3. for kd=1:numel(dfa)
4.   th1=dfa(kd)+dfa(kd)*.10;
5.   th2=dfa(kd)-dfa(kd)*.10;
6.   if simtest<th1 && simtest>th2
7.     duplicate=1;
8.   else
9.     end
10.  end
11. result_test(ii)=duplicate;
12. end

```

I. Calculation of Performance Parameters

To check the performance of machine learning model works on sensitivity, specificity, fmeasure, and accuracy parameters are carried out. Accuracy percentage rate increases when outcome of *result_test* and ground truth test (*gt_test*) are nearby same to each other. Here, frequencies of 1 and 0 are computed separately for each *result_test* and *gt_test* elements. Calculation of *sensitivity* (for frequency 1) and *Specificity* (for frequency 0) percentage is carried as per coding lines 24 and 25 respectively.

1) Pseudocode to compute performance parameters

Input: *result_test*, *gt_test*

Output: function [sensitivity, Specificity, fmeasure] = parameters (output, actual)

```

1. ar=actual;
2. actual=[];
3. actual=ar;
4. fone=0; % one in test result
5. ft=0; % one in test gt
6. for i=1:numel(actual)
7.   if actual(i)==1
8.     ft=ft+1;
9.   if output(i)==1
10.    fone=fone+1;
11.  end
12. end
13. end
14. fzero=0; % zero in test result
15. fz=0; % zero in test gt
16. for i=1:numel(actual)
17.   if actual(i)==0
18.     fz=fz+1;
19.   if output(i)==0
20.    fzero=fzero+1;
21.  end
22. end
23. end
24. sensitivity=(fone/ft)*100;

```

```

25. Specificity=(fzero/fz)*100;
26. acflag=zeros(1,numel(output));
27. for kk=1:numel(output)
28. if output(kk)==actual(kk)
29. acflag(kk)=1;
30. end
31. end
32. match=0;
33. for k=1:numel(output)
34. if output(k)==actual(k)
35. match=match+1;
36. end
37. end
38. tempv=sensitivity;
39. sensitivity=Specificity;
40. Specificity=tempv;
41. accuracy=((((numel(find(acflag==1))/numel(acflag)))
+((numel(find(acflag==0))/numel(acflag))))/2)*100;
42. accuracy=(match/numel(output))*100;
43. fmeasure=2*sensitivity*Specificity/(sensitivity+Specificity);
44. fmeasure=fmeasure/100;
45. end

```

Four aspects are examined for the assessment of proposed deduplication approach includes sensitivity, specificity, accuracy and fmeasure evaluated at code lines 24-44 in above concerned pseudocode. The sensitivity and specificity value are evaluated based on pertinent information at *fone*, *fzero*. *fone* is characterized by prediction of duplicate records as duplicate and *fzero* which incorrectly imply a duplicate record. Accuracy is defined as the proportion of correct test data predictions. The number of right predictions may be determined easily by the number of total forecasts. Fmeasure shows the exactness of a model on a dataset.

IV. RESULTS AND ANALYSIS

Different assessment criteria are used to analyze the performance of proposed deduplication approach. The methods are implemented in MATLAB 2016a, Intel core 2 Processor, Clock Speed of 2.1MHz, and 4GB of RAM.

A. Performance Evaluation

This section presents the performance study of the proposed deduplication method based on dataset in [18],[20]. The performance is evaluated on both the datasets and found that proposed machine learning architecture is efficient in achieving accuracy performance parameter. Specificity, sensitivity, accuracy, and fmeasure are the evaluation parameters employed as represented in Fig. 3 for both datasets. Measurements on parameters are carried out to estimate performance specification and are discussed in Table 3. An average of 99.7 accuracy result is found in proposed advanced machine learning architecture.

TABLE III. PERFORMANCE PARAMETERS

Dataset	Sensitivity	Specificity	Fmeasure	Accuracy
Sentiment Analysis with Tweets [18]	1	0.004545	0.00904	100
Tweets about the top companies from 2015-to-2020 [20]	0.99408	0	0	99.4083
Average Accuracy				99.7

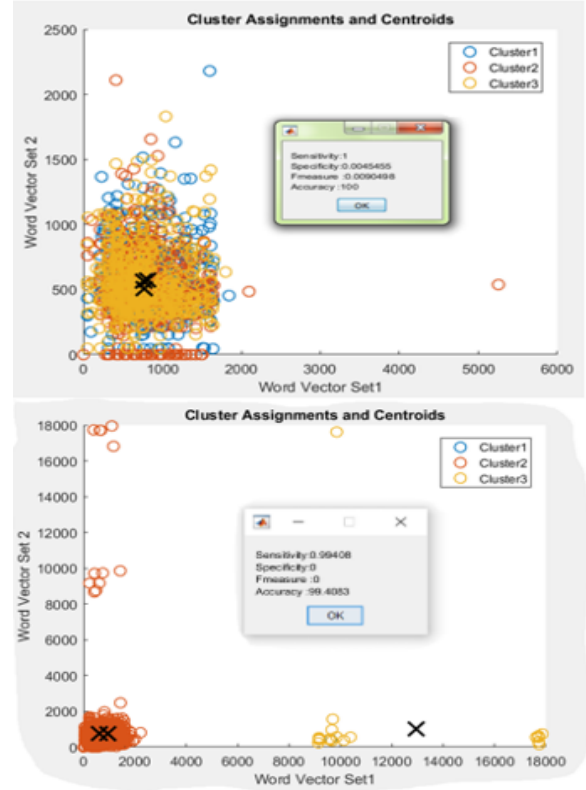


Fig. 3. Graphical Representation of clusters and centroids with performance parameters

V. COMPARATIVE STUDY ON PROPOSED AND EXISTING TECHNIQUES

Different techniques were purposed by researchers to maintain accuracy in the deduplication process. In this proposed scheme an advanced machine learning architecture is used and a comparative analysis is conducted with existing SVM, NN, Fuzzy approaches and found that our scheme is having high accuracy rate of 100%. The performance analysis of deduplication process is carried out on accuracy parameter as shown in table 4 and bar-graph in Fig. 4. to represent metrics results. The comparison is done on proposed work and existing methodology worked on accuracy parameter.

TABLE IV. COMPARITIVE RESULTS OF PROPOSED WITH EXISTING DEDUPLICATION SCHEMES

Proposed and Existing Data Deduplication Schemes	Accuracy %age
Advance Machine Learning Architecture (Proposed Model)	99.7
FEM (Fuzzy Expectation Maximization) Clustering [9]	97.98
Enhanced Fuzzy Ontology Based Record Deduplication [6]	91.5
Support Vector Model [14]	88
Artificial Neural Network Model [15]	79.8

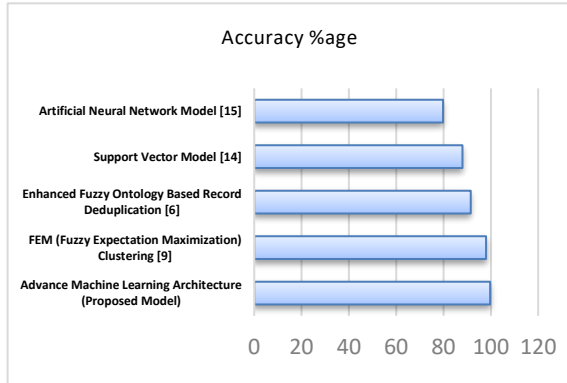


Fig. 4. Graphical Bar Representation of Proposed Scheme Accuracy with Existing Deduplication Schemes

VI. CONCLUSION

This study suggests an advanced machine learning architecture to detect duplicate data on distributed storage and also monitor new data inputs as well. It stresses on the need of an affective deduplication technique that helps to detect duplicate data entries. We proposed an advance algorithm based on machine learning model helps to detect duplicate data entries and work for the enhancement of workforce efficiency and system performance. It follows a relative approach to detect duplicate data in the textual context using unsupervised learning approach. The proposed methodology has achieved an accuracy of 100% in comparison to existing SVM, NN, Fuzzy models accuracy results.

REFERENCES

- [1] Ahmed Sardar M. Saeed and Loay E. George, "Data Deduplication System Based on Content-Defined Chunking Using Bytes Pair Frequency Occurrence", *Symmetry*, 2020, 12(11), 1841. <https://doi.org/10.3390/sym12111841>
- [2] A Vijayakumar and A Nisha Jebaseeli, "Pioneer approach of data deduplication to remove redundant data from cloud storage", *International Journal of Advanced Research in Engineering and Technology (IJARET)*, Volume 11, Issue 10, October 2020, pp. 535-544, DOI: 10.34218/IJARET.11.10.2020.057
- [3] Dinesh Mishra, Dr. Sanjeev Patwa, "Attribute and Role Based Deduplication", *International Journal of Advanced Science and Technology*, vol. 29(7), pp. 14597 – 14606, 2020.
- [4] Duaa S. Naji and Loay E. George, "A Technique for Big Data Deduplication based on Content Attributes and Dictionary Indexing", *IOP Conference Series: Materials Science and Engineering*, Volume 928, 2nd International Scientific Conference of Al-Ayen University (ISCAU-2020) 15-16 July 2020, Thi-Qar, Iraq
- [5] S. Ruba, A.M. Kalpana, "Machine Learning Techniques Used To Store Efficient Cloud Data Through Chunking And Data Deduplication Process", *Proteus Journal*, 2020, pp. 60-75, Vol.11, Issue 12. <https://doi.org/10.37896/PJ11.12/043>
- [6] R. Parimala Devi, "Enhanced Fuzzy Ontology Based Record Deduplication", *Aut Aut Research Journal*, Vol. 11(9), pp. 499-515 [DOI:10.0001865.Aut.Aut.2020.V11I9.463782.00746](https://doi.org/10.0001865.Aut.Aut.2020.V11I9.463782.00746)
- [7] Guangpin X, Bo T, et al., "LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication", *35th Symposium on Mass Storage Systems and Technologies (MSST)*, *IEEE*, pp. 299-310, 2019.
- [8] H. Fingler, M. Ra and R. Panta, "Scalable, Efficient, and Policy-Aware Deduplication for Primary Distributed Storage Systems," 2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2019, pp. 180-187, doi: 10.1109/SBAC-PAD.2019.00038.
- [9] P. Selvi, D. Shanmuga Priyaa, "An Enhanced Unsupervised Fuzzy Expectation Maximization Clustering for Deduplication of Records in Big data", *International Journal of Recent Technology and Engineering (IJRTE)*, Volume-8 Issue-3, pp. 988-993, 2019. DOI:10.35940/ijrte.C1269.1083S219
- [10] H. A. S. Jasim and A. A. Fahad, "New techniques to enhance data deduplication using content based-TTDD chunking algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 5, pp. 116–121, 2018, doi: 10.14569/IJACSA.2018.090515.
- [11] N. Kumar, S. Antwal, G. Samarthyam and S. C. Jain, "Genetic optimized data deduplication for distributed big data storage systems," 2017 4th International Conference on Signal Processing, Computing and Control (ISPPCC), 2017, pp. 7-15, doi: 10.1109/ISPPCC.2017.8269581.
- [12] N. Kumar, R. Rawat, and S. C. Jain, "Bucket based data deduplication technique for big data storage system," *2016 5th Int. Conf. Reliab. Infocom Technol. Optim. ICRITO 2016 Trends Futur. Dir.*, pp. 267–271, 2016, doi: 10.1109/ICRITO.2016.7784963.
- [13] Y. xuan Xing, N. Xiao, F. Liu, Z. Sun, and W. hui He, "AR-dedupe: An efficient deduplication approach for cluster deduplication system," *J. Shanghai Jiaotong Univ.*, vol. 20, no. 1, pp. 76–81, 2015, doi: 10.1007/s12204-015-1591-1.
- [14] M. Padmanaban and T. Bhuvaneshwari, "A Technique for Data Deduplication using Q-Gram Concept with Support Vector Machine", *International Journal of Computer Applications*, Vol. 61(12), pp. 1-9, 2013
- [15] M. Padmanaban and T. Bhuvaneshwari, "An Approach Based on Artificial Neural Network for Data Deduplication", *International Journal of Computer Science and Information Technologies*, Vol. 3(4), pp. 4637-4644, 2012.
- [16] Y. Zhang, Y. Wu and G. Yang, "Droplet: A Distributed Solution of Data Deduplication," *2012 ACM/IEEE 13th International Conference on Grid Computing*, pp. 114-121, 2012, doi: 10.1109/Grid.2012.21.
- [17] S. S. Sengar and M. Mishra, "E-DAID: An Efficient Distributed Architecture for In-Line Data De-duplication," 2012 International Conference on Communication Systems and Network Technologies, 2012, pp. 438-442, doi: 10.1109/CSNT.2012.101.
- [18] URL: <https://www.kaggle.com/kazanova/sentiment140>
- [19] URL: <https://gist.github.com/larsyencken/1440509>
- [20] URL: <https://www.kaggle.com/omermetin/tweets-about-the-top-companies-from-2015-to-2020?select=Tweet.csv>

An optimized cost-based data allocation model for heterogeneous distributed computing systems

Sashi Tarun¹, Mithilesh Kumar Dubey¹, Ranbir Singh Batth¹, Sukhpreet Kaur²

¹School of Computer Science Engineering, Lovely Professional University, Phagwara, India

²Department of Computer Science Engineering, Chandigarh Engineering College, Chandigarh, India

Article Info

Article history:

Received Jun 25, 2021

Revised Jun 11, 2022

Accepted Jul 7, 2022

Keywords:

Communication cost

Computation cost

Data allocation

Execution time

Network cost

Total cost

ABSTRACT

Continuous attempts have been made to improve the flexibility and effectiveness of distributed computing systems. Extensive effort in the fields of connectivity technologies, network programs, high processing components, and storage helps to improvise results. However, concerns such as slowness in response, long execution time, and long completion time have been identified as stumbling blocks that hinder performance and require additional attention. These defects increased the total system cost and made the data allocation procedure for a geographically dispersed setup difficult. The load-based architectural model has been strengthened to improve data allocation performance. To do this, an abstract job model is employed, and a data query file containing input data is processed on a directed acyclic graph. The jobs are executed on the processing engine with the lowest execution cost, and the system's total cost is calculated. The total cost is computed by summing the costs of communication, computation, and network. The total cost of the system will be reduced using a Swarm intelligence algorithm. In heterogeneous distributed computing systems, the suggested approach attempts to reduce the system's total cost and improve data distribution. According to simulation results, the technique efficiently lowers total system cost and optimizes partitioned data allocation.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sashi Tarun

School of Computer Science Engineering, Lovely Professional University

Phagwara, India

Email: sashitarun79@gmail.com

1. INTRODUCTION

The success of a distributed system is determined by how data is fragmented and distributed across several geographical locations [1], [2]. One of the most difficult aspects of distributed architecture is overcoming the growing workforce load and cutting down data allocation costs. Load implies a longer query completion time, which has an impact on operational costs and increases the overall system execution cost. To keep the overall system execution cost low, data fragments must be broken down into little sub-tasks and planned to be processed in parallel. The objective of this type of heuristic division is to reduce the time it takes to complete all jobs.

The number of processing engines and tasks used in the execution is determined by the query file and the number of processing engines. Each engine has its own computing cost for calculating task execution costs. The execution cost is calculated by adding the lowest execution cost to the communication and network costs incurred during the processing engine's execution.

To parallel execute all jobs, a directed acyclic graph (DAG) with vertices and edges as shown in Figure 1 was utilized. Each job is represented as a vertex or node in a DAG, and the edges between them

reflect the communication cost and connection between the tasks. The cost of communication is incurred when data is transferred between nodes. DAG is responsible for three sorts of costs: communication, network, and computation. Each expense factored into the total cost of system execution. Only the calculation cost with the shortest execution time is considered for total cost computation.

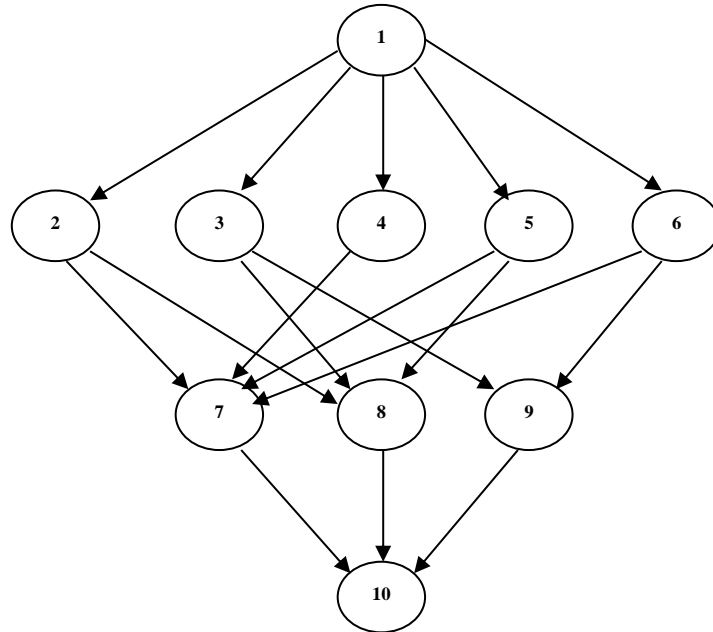


Figure 1. Directed acyclic graph

We consider a distributed system with heterogeneous processors and system tasks in this paper. $P=P_1, P_2, \dots, P_n$ of heterogeneous processors connected by communications links, and $T=t_1, t_2, \dots, t_m$ of system tasks that collectively express a purpose, are represented by a distributed system. The execution cost matrix (ECM), which is an asymmetrical matrix of order $m*m$, represents the cost of execution of tasks on different processors, while the network cost matrix (NCM), which is an asymmetrical matrix of order $m*m$, represents the cost of communication between multiple tasks, as shown in Tables 1 and 2. By using both, the suggested artificial bee colony (ABC) algorithm may compute the best overall cost.

Table 1. Execution cost matrix

Query/Tasks	Processors		
	P1	P2	P3
1	0.81	0.16	0.66
2	0.91	0.97	0.04
3	0.13	0.96	0.85
4	0.91	0.49	0.93
5	0.63	0.80	0.68
6	0.10	0.14	0.76
7	0.28	0.42	0.74
8	0.55	0.92	0.39
9	0.96	0.79	0.66
10	0.96	0.96	0.17

Table 2. Network cost matrix

Query/Tasks	Processors		
	P1	P2	P3
1	0.71	0.44	0.28
2	0.03	0.38	0.68
3	0.28	0.77	0.66
4	0.05	0.80	0.16
5	0.10	0.19	0.12
6	0.82	0.49	0.50
7	0.69	0.45	0.96
8	0.32	0.65	0.34
9	0.95	0.71	0.59
10	0.03	0.75	0.22

Delay in response, high execution time, and high completion time are the issues with data allocation in a distributed system [3]. It progressively boosts system costs and impacts workforce progress. The use of several processors increased network and communication costs during task execution, which had an impact on the overall system cost. Earlier techniques used by the researchers excluded network costs from overall system cost calculations. In this situation, the previously produced study results appear to be erroneous and useless in different experiments in a distributed environment. The proposed study focuses on a swarm intelligence-based artificial bee colony method useful for addressing and resolving current difficulties and

rapidly working for handling previous flaws [4]–[9] discusses previous issues. The proposed method is based on the learning and adaptive behavior of bees, which might be beneficial in resolving performance difficulties. It controls bee-degradation loss that arises as a result of the high expense of collecting the bees from the location. To accomplish optimization, such costs are subtracted from the overall system cost to balance loss. It aids in the betterment of data allocation in a distributed computing system.

The following is a list of the topics covered in this paper. The second section goes through important research and findings. The third section introduces the technique we suggest. The simulation results of our suggested ABC method are shown in section 4. Section 5 compares and contrasts the proposed algorithm with existing algorithms. Finally, section 6 presents the paper's conclusion.

2. RELATED WORK

An energy-efficient dynamic loading and resource scheduling method includes reducing energy usage and decreasing application times. The method also successfully decreases energy efficiency by modifying the central processing unit (CPU) clock frequency of smart mobile devices to the optimum in local computing and adjusting the communication energy of wireless channels in cloud computing [10]. For the placement of virtual machines in cloud computing an energy-efficient order exchange and migration ant colony system (OEMACS) algorithm was created. The intended virtual machine placement was achieved with the fewest number of active machines and by turning off idle nodes. According to experimental investigations, OEMACS aimed to minimize the number of active servers, increase resource use, balance diverse resources, and reduce power consumption [11]. To offer energy and service-sensitive performance in the placement and consolidation of virtual machines, a multi-target colony optimization technique was presented. The results demonstrate that this technique outperforms the other ways in terms of energy consumption, limiting CPU waste, lowering energy communications costs caused by traffic sharing across virtual machines, and reducing the number of virtual migrations to system and service level agreement (SLA) violations [12]. To minimize all of cloud data center power consumption a platform for virtual machine placement was introduced. The adaptability and scalability of the platform proposed resulted in exceptional success in virtual machine deployment and relocation processes [13].

An evaluation of current reliability and energy management strategies and their effect on cloud computing was discussed. There were debates on the classification of resource loss, failure tolerance mechanisms, and mechanisms for energy conservation in cloud systems. Different problems and study gaps have been established in the balance between energy reliability and quality [14]. A strong immune clonal optimization method based on the dynamic load balance approach and immune clonal selection theory in green cloud computing has solved the problem of high energy consumption and reduced cloud utilization. In terms of solution efficiency and processing costs, the experimental findings show that the method outperforms clonal selection techniques [15]. The need for energy management is demonstrated when addressing the dual position of cloud computing as a significant contributor to rising energy use and reducing energy waste. The research provided an in-depth analysis of current energy management methods in cloud computing. It also supplies taxonomies for the assessment of current work in the area of science [16]. Consolidation of tasks as an efficient way to maximize resource usage and minimize energy use was addressed. The research focused on two energy-conscious energy consolidation heuristics intended to optimize the use of resources and take both active and idle energy use directly into account. The heuristics suggested assigning each job to the resource, which minimizes the energy needs explicitly or indirectly, without degradation in performance [17]. An energy-efficient cloud computing architectural structure and concepts were suggested. The study identified open analysis problems, the provision of infrastructure as well as algorithms to handle cloud computing environments effectively. The conclusions indicate that the suggested model of cloud storage makes substantial cost savings and has a high capacity in complex workload environments for energy efficiency improvements [18].

An algorithm for allocating the total energy consumption was introduced. The effects of simulations were also viewed on a state-of-the-art platform. The suggested solution results in tangible energy conservation, showing energy efficiency dominance in comparison with well-known and widely accepted allocation methods [19]. The research was conducted on maximizing physical and virtual machines' capacity and energy consumption in a cloud computing system. Findings offered a good understanding of how power and energy usage were affected by various workloads. The tools and structure presented can be used for research and improving energy efficiency in any cloud environment and of any scale [20]. A problem of energy optimization has been modeled whereas the task dependence, transfer of data, and some constraints such as response time, and cost have been considered and solved by genetic algorithms. A series of simulation trials have been carried out to assess the algorithm efficiency and the findings suggest that the proposal is more effective than the benchmark method [21]. To decrease energy usage in cloud data centers, an optimal paradigm for work schedules has been proposed. The proposed solution was designed as an

integer programming problem to reduce the energy consumption of a cloud-based data center by organizing activities for a small number of servers and adhering to task response time constraints. As a realistic program, the authors have developed the most effective initial task-programming method for the server to decrease energy expenses. A data center planning system with diverse tasks is modeled and simulated. The study findings reveal that the recommended work scheduling strategy reduces server power consumption by more than 70 times on average when compared to a random job scheduling system [22]. A power-aware scheduling approach for a heterogeneous cloud network was suggested to solve the issue of high energy consumption. The results show that the average power consumption in this system is 23.9-6.6% lower than in modern technology [23]. An abstract model was proposed that uses piecewise linear functions to handle data analytics workload in a distributed cluster architecture. This is responsible to reduce the makespan time to handle cost issues [24]. A hybrid heuristic genetic algorithm and the steepest descent methods were used to achieve optimal task allocation with the reduction in hardware policies to reduce system cost [25]. A latency-aware max-min algorithm (LAM) has been developed for resource allocation in cloud infrastructures. The suggested method was developed to handle resource allocation challenges such as changing user requirements and on-demand access to infinite resources. It may allocate resources in a cloud-based environment to enhance infrastructure performance and increase revenue [26].

3. PROPOSED METHOD

The data allocation method is based upon the processing engine. The data allocation process consumes a higher network cost if the cost parameter is not controlled correctly. The cost of doing activities on multiple processors varies in this case. When two dependent jobs are run on the same processor, the computing cost is the same, and communication between them is regarded as zero. The planned activities are repeated on the specific processing engines to compute execution costs. To calculate the task execution cost, the lowest cost of query execution is applied to the communication cost. The following stages are carried out to carry out the planned task, as illustrated in Figure 2.

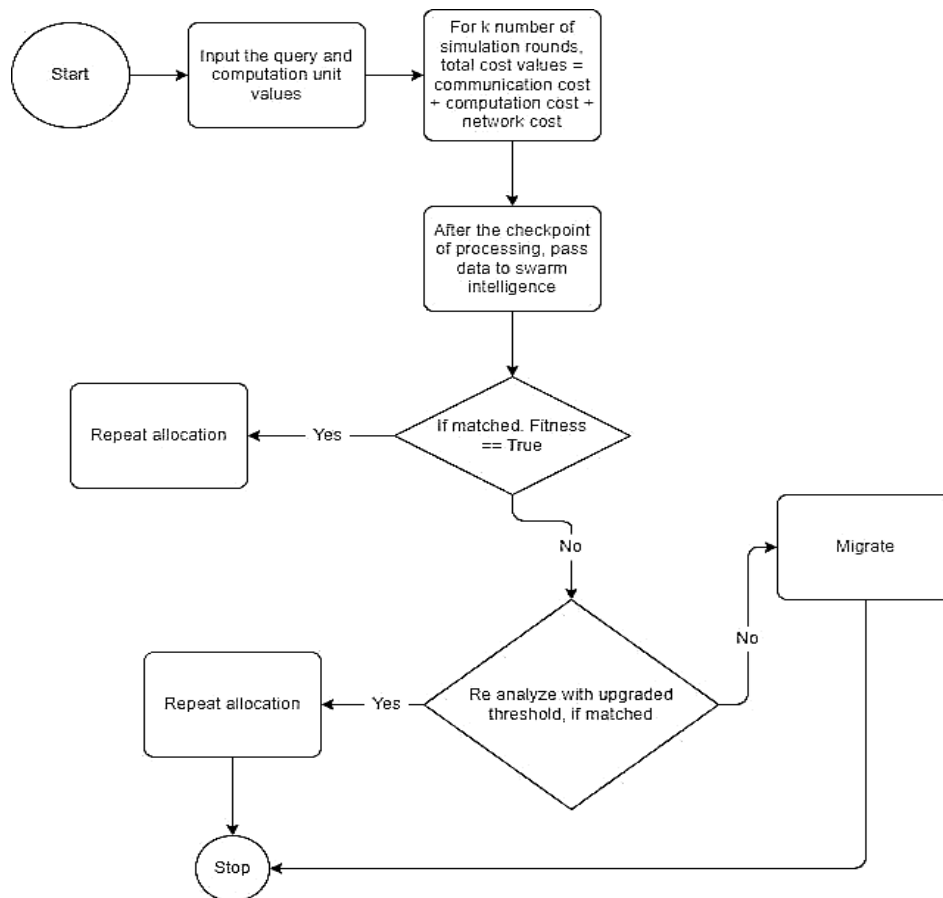


Figure 2. Proposed method flow

3.1. Load the query data file

The communication and processing costs of the tasks in a flow are utilized to compute the query cost. It is a flow diagram that shows the execution of queries and tasks at different levels. The lowest cost processing engine (PE) is chosen to determine the calculation cost. The query model flow is used to illustrate the details of all jobs, including their flow and associated communication and computing costs, as shown in Table 3.

Table 3. Query model flow (*query_data*)

Query/Tasks flow		Communication Cost	Computation Cost		
T1	T2		PE1	PE2	PE3
1	2	12	12	10	14
1	3	15	8	9	15
1	4	17	14	18	15
1	5	14	11	14	9
1	6	16	12	14	17
2	7	14	11	15	17
2	8	15	12	17	19
3	8	18	12	17	19
3	9	17	17	18	13
4	7	18	11	15	17
5	7	17	11	15	17
5	8	25	12	17	19
6	9	10	17	18	13
6	7	14	11	15	17
7	10	12	14	15	13
8	10	13	14	15	13
9	10	14	14	15	13

3.2. Total cost estimation

The total cost is defined as the total energy incurred during the execution of all tasks. At the node level, a level-wise cumulative computation is used to calculate overall costs. At each level, the load is measured by summing all communication, computation, and network access expenses for all jobs. The overall cost spent after selecting processors for task execution is represented by all three factors. All processors have their specifications and perform tasks without violating the priority limitations set by the operating system. It indicates it will not let you break the workflow sequence. Sub-tasks are executed after each of its parents has completed their execution. Instead of focusing on task prioritization at each node, preference is given to processors with lower execution costs for a task assessment. Tasks are data fragments that are placed on each node and vary in size. To compute the total cost at level 1 (root-level) following steps are executed as shown in pseudocode 1. It is responsible to compute total cost by SumUp least execution cost of PE with other parameters cost incurred at each level.

Pseudocode 1. Pseudocode to calculate execution pattern of level 1 tasks

Input: *entry_query*, *query_comp_cost*, *myminvalue*, *myminpos*

Output: *total_engine_value*

```

1 execution_pattern=[ ];
2 total_engine_value=zeros(1,3);
3 [myminvalue,myminpos]=min(query_comp_cost(1,:));
4 execution_pattern(1,1)=entry_query;
5 execution_pattern(1,2)=0;
6 execution_pattern(1,3)=myminvalue;
7 execution_pattern(1,4)=myminpos;
8 total_engine_value(1,myminpos)=total_engine_value(1,myminpos)+myminvalue;

```

Here, to calculate the execution pattern at level 2, it is required to count number of queries comes under this level, find connectivity between parent and current nodes and compute their communication cost. The communication cost become same in the case if tasks are executed on the same processor else not. To calculate the total cost incurred during the execution of individual task/queries at current level communication cost, and least execution cost of machines are involved. Pseudocode 2 is responsible to evaluate all cost parameters to compute the overall cost of level 2.

Pseudocode 2. Pseudocode to calculate execution pattern of level 2 tasks

Input: *last_engine*, *total_query_count*, *lvcount2*, *current*, *parent*, *query_engine*

Output: *comm_costtt*, *total_cost*, *total_engine_value*

```

1 last_engine=myminpos;

```



```

2 total_cost=[ ];
3 total_query_count=2;
4 for i=1:lvcount2           % return level-2 queries count
5 current=level(2,i);
6 parent=entry_query;
7 for j=1:query_engines
8 comp_current=query_comp_cost(current,j);
9 if j~=last_engine
10 comm_costt=0;
11 sd=find(query_comm_cost(:,1)==entry_query); % return total no. of queries related to 1
12 for k=1:numel(sd)
13 kp=query_comm_cost(sd(k),2);
14 if kp==current
15 comm_costt=query_comm_cost(k,3);
16 end
17 end
18 total_cost(j)=comm_costt+comp_current+total_engine_value(1,j);
19 else
20 total_cost(j)=comp_current+total_engine_value(1,j);
21 end
22 end

```

The pseudocode 3 estimate the level 3 tasks total execution cost. Here, all tasks are evaluated level-wise in sequence to compute the total cost pattern. During computation cost, communication cost and network cost values are summed-up.

Pseudocode 3. Pseudocode to calculate execution pattern of level 3 tasks

Input: *lvcounter*, *query_data*, *execution_pattern*
Output: *parentcurrent*, *parentfinishtime*, *totalcost*

```

1 for i=1:lvcounter           % Tally execution pattern of level 3
2 current=lv3jobs(i);
3 parentcurrent=[ ];
4 parentfinishtime=[ ];
5 counter=0;
6 [dp,pos]=find(query_data(:,2)==current);
7 for j=1:numel(dp)
8 parentcurrent(j)=query_data(dp(j),1);
9 currentparent=parentcurrent(j);
10 m=find(execution_pattern(:,1)==currentparent);
11 currentparentfinishtime=execution_pattern(m,3);
12 parentfinishtime(j)=currentparentfinishtime;
13 parentprocessor(j)=execution_pattern(m,4);
14 end
15 [maxval,maxpos]=max(parentfinishtime);
16 minstarttime=maxval;
17 parentp=parentprocessor(maxpos);
18 totalcost=[ ];
19 for j=1:3
20 [p,k]=find(execution_pattern(:,4)==j);
21 lasttime=execution_pattern(p(numel(p)),3);
22 if lasttime<minstarttime
23 lasttime=minstarttime;
24 end
25 totalcost(j)=lasttime;
26 end

```

The queries at level 4 are conducted once the parent tasks at level 3 have been completed. The size of the level is calculated at this level, and each task is assessed row-by-row. There is one task marked as current in this. According to *query_data*, the current task or query has three parents that are each represented as *parentcurrent*. Each *parentcurrent's* execution cost is calculated separately. As illustrated in pseudocode 4, network costs and PEs with the lowest execution costs are added to compute the overall cost during evaluation of tasks and queries at level 4. At the end, total execution cost of processing engines (PEs) is computed and the smallest execution cost of PE is picked.

Pseudocode 4. Pseudocode to calculate execution pattern of level 4 tasks

Input: *lvcounter*, *current*, *parentcurrent*, *parentfinishtime*, *query_data*, *execution_pattern*
Output: *totalcost*

```

1. for i=1:lvcounter
2. current=lv4jobs(i);
3. parentcurrent=[ ];

```

```

4. parentfinishtime=[ ];
5. counter=0;
6. [dp, pos]=find(query_data(:,2)==current);
7. for j=1:numel(dp)
8. parentcurrent(j)=query_data(dp(j),1);
9. currentparent=parentcurrent(j);
10. m=find(execution_pattern(:,1)==currentparent);
11. currentparentfinishtime=execution_pattern(m,3);
12. parentfinishtime(j)=currentparentfinishtime;
13. parentprocessor(j)=execution_pattern(m,4);
14. end
15. [maxval, maxpos]=max(parentfinishtime);
16. minstarttime=maxval;
17. parentp=parentprocessor(maxpos);
18. totalcost=[ ];
19. for j=1:3
20. [p, k]=find(execution_pattern(:,4)==j);
21. lasttime=execution_pattern(p(numel(p)),3);
22. if lasttime<minstarttime
23. lasttime=minstarttime;
24. end
25. totalcost(j)=lasttime;
26. end

```

3.2.1. Calculation of total system cost

The system's total cost is calculated by adding the least execution cost of all tasks using (3). The execution cost of the processing engine for each task is estimated using (1). The least execution cost is estimated using (2). The practical calculation of implementation for each processing engine is shown in Table 4.

Table 4. Execution cost of each processing engine

Query/tasks	1	2	3	4	5	6	7	8	9	10																				
Computation cost of tasks	9	8	7	12	10	14	8	9	15	14	18	15	11	14	9	12	14	17	11	15	17	12	17	19	17	18	13	14	15	13
Execution cost of each PEs	0	0	7	24	22	21	23	24	43	54	35	43	48	63	37	51	65	82	53	57	60	53	45	37	63	60	61	64	72	70
Least exec. cost	7		21		23		35		37		51		53		53		45		60		60		64		64		64		64	
Total system cost																														396

$$PEs \text{ Execution Cost} = \text{communication cost} + \text{computation cost} + \text{network cost} \quad (1)$$

$$\text{Least Execution Cost} = \min(\text{execution cost of each PEs}) \quad (2)$$

$$\text{Total Execution Cost} = \text{sum of least execution cost of all tasks} \quad (3)$$

3.2.2. Calculation of execution pattern

Execution cost matrix (ECM) and network cost matrix (NCM) results as shown in Tables 1 and 2 respectively are used to compute execution patterns for each processing task and indicated processing engine task wise. In this, jobs at various levels are processed in topological order, and their associated communication, network, and computing costs from the communicating node are calculated and added. This method is used for all jobs and lies on different levels (0-nth levels) depending on directed acyclic graph (DAG) size.

The results computed at each level are added to determine the overall execution cost. Information on the tasks that are carried out one after the other starting at the root level is provided by pseudocode 5. From the root level, the execution cost pattern is calculated individually for each job from 1 to 10 by noting the starting and ending consumption units and the processing engines involved in each task. The overall cost is then calculated task-by-task as shown in Table 5. Every time, the difference between the starting and ending consumption units is used to calculate the execution cost. For each task, the energy pattern from the ECM is chosen based on the corresponding processing engine ID.

Pseudocode 5. Pseudocode to calculate execution cost pattern for each task with PEs ID

Input: *execution_pattern*, *network_cost*, *currentp*, *total_query_count*, *ex_pt*

Output: *ecost*

```

1. ex_pt=[ ];
2. ex_pt{1,1}='Query No';
3. ex_pt{1,2}='Starting Consumption Unit';
4. ex_pt{1,3}='Ending Consumption Unit';
5. ex_pt{1,4}='DB Engine ID';
6. total_query_count=10;
7. for i=1:total_query_count
8. ex_pt{i+1,1}=execution_pattern(i,1);
9. ex_pt{i+1,2}=execution_pattern(i,2);
10. ex_pt{i+1,3}=execution_pattern(i,3);
11. ex_pt{i+1,4}=execution_pattern(i,4);
12. end
13. for i=1:total_query_count
14. currentdiff=execution_pattern(i,3)-execution_pattern(i,2);
15. currenttp=execution_pattern(i,4);
16. ecost=energypattern(i,currenttp);
17. ecost=ecost+networkcost(i,currenttp);
18. execution_pattern(i,5)=ecost;
19. end

```

Table 5. Execution cost pattern for each processing task

Tasks	Starting consumption unit	Ending consumption unit	Processing engines ID	Execution cost pattern (mJ)
1	0	7	3	6.5224
2	7	21	3	0.7154
3	0	23	1	0.4039
4	0	35	2	1.2806
5	28	37	3	0.7977
6	23	51	1	0.9210
7	51	53	1	0.9733
8	37	45	2	1.5620
9	51	60	2	1.5016
10	60	64	1	0.9993

3.3. Proposed artificial bee colony

A swarm intelligence algorithm is a step toward dealing with issues that cannot be handled by the traditional numerical methods. The honey bees represent a quick social collective behavior having the ability to adapt, learn, and update themselves. It inspired most researchers to apply it for the optimization of results. This algorithm is based on bee colony behavior. Here bees are of three types; employed bees (those responsible for food collection), onlooker bees (those responsible for food monitoring), and scout bees (those are in rest). ABC algorithm works here to optimize the total execution cost as shown in pseudocode 6.

Pseudocode 6. Artificial bee colony (ABC)

```

Input: Food Source
Output: [scout,beedegradation] =beefitness(employed_bee, energypattern, networkcost,
timemodel, currentprocessor, taskname)
1. scout=0;
2. beedegradation=0;
3. restprocessors=[ ];
4. rc=1;
5. for i=1:3
6. if i~=currentprocessor
7. restprocessors(rc)=i;
8. rc=rc+1;
9. end
10. end
11. for i=1: numel(restprocessors)
12. onlooker_bee_value(i)=timemodel*(energypattern(taskname, restprocessors(i)),
networkcost(taskname, restprocessors(i)));
13. end
14. selected_food_source=min(onlooker_bee_value);
15. onlooker_bee_selection=selected_food_source;
16. employed_bee=employed_bee*timemodel;
17. natural_change_onlooker=rand;
18. natural_change_employed=rand;
19. if onlooker_bee_selection*natural_change_onlooker > employed_bee*natural_change_employed
20. scout=0;

```

```

21. beedegradation=0;
22. else
23. scout=1;
24. beedegradation=((employed_bee*natural_change_employed) -
    (onlooker_bee_selection*natural_change_onlooker)) /timemodel;
25. end
26. end
    
```

3.4. Fitness function checking

Fitness function aids in validating the overall execution cost in distributed systems to accomplish data allocation at a minimal cost. It compares and verifies the outcomes, as well as cover the bee degradation part to reach to the optimal state. The code lines 19-25 of proposed ABC pseudocode 6 depicts that if the results after multiplication of *onlooker_bee_selection*natural_change_onlooker* is greater than the results of *employed_bee*natural_change_employed*, in such case there is no occurrence of *scout* and *beedegradation* loss otherwise *scout* and *beedegradation* occurs. This can be accomplished by subtracting the cost of *beedegradation* from the total system cost. This is the expense of collecting nectar from various sites, including the time spent traveling back and forth.

3.5. Re-analyze with upgraded threshold to get optimized result

Table 6 illustrates the results of the ABC algorithm's calculations. It shows the measurements between different variables involves to achieve optimal results. Finally, it calculates optimal results in mJ, which show the system's total execution cost in pseudocode 7. This can be achieved by setting threshold value in which optimal results is further subtracted by time spent by bees during the collection of nectar from flowers in different location indicated as *beedegradation*. The measurement in Table 6 shows the calculations carried out to justify the effectiveness of ABC algorithm. It shows total system cost estimated after reanalyzing the cost discovered. Here, all variables are interpreted in view to achieve the results.

Pseudocode 7. Pseudocode to re-analyze with upgraded threshold

```

Input: scout, beed, timemodal
Output: Optimal
1. if scout>0
2. optimal(i,5)=abs(optimal(i,5)-beed/timemodel);
3. end
    
```

Table 6. Fitness value and re-analyze process for optimal result

Tasks	Current Processor	Rest Processor	Onlooker_ bee value	onlooker _bee_ selection (A)	natural_ change_ onlooker (B)	employed_ bee (C)	natural_ change _ employed (D)	If A*B > C*D	Scout	beed	Optimal Result After re-analyze (total system cost in mJ)
1	3	1	10.6454	4.1745	0.7513	45.6565	0.2551	False	1	1.21	6.3487
		2	4.1745							58	
2	3	1	13.1267	13.1267	0.5060	140.2212	0.6991	False	1	6.52	9.5496
		2	18.9301							74	
3	1	2	39.6217	34.5972	0.8909	213.6683	0.9593	False	1	7.5716	8.9607
		3	34.5972								
4	2	1	33.5842	38.3812	0.5472	44.8201	0.1386	False	1	5.6881	44.6576
		3	38.3812								
5	3	1	6.5654	6.5654	0.1493	64.6164	0.2575	False	1	1.7399	6.9863
		2	8.8844								
6	1	2	17.6862	17.6862	0.8407	722.0626	0.2543	False	1	6.0264	25.5727
		3	35.1709								
7	1	2	1.7347	1.7347	0.8143	3.8933	0.2435	True	0	0	1.9467
		3	3.4058								
8	2	1	6.9118	5.8609	0.9293	99.9711	0.3500	False	1	3.6927	12.0348
		3	5.8609								
9	2	1	17.1696	11.1667	0.1966	121.6273	0.2511	False	1	3.1493	13.1642
		3	11.1667								
10	1	2	6.8567	1.5800	0.6160	15.9894	0.4733	False	1	1.6486	3.5852
		3	1.5800								

4. SIMULATION RESULTS AND DISCUSSION

The ABC method is implemented in MATLAB and executed on an Intel Core i3 processor 11 generations with a clock speed of 3.00 GHz and 4 GB of RAM. Megajoules (mJ) are units of measurement for the amount of energy consumed. In Table 7, the proposed work is compared to existing approaches. It was discovered that previously suggested approaches did not account for network costs in their calculations,

and anomalies were discovered that impacted overall system costs. To obtain the optimal least cost, all parameters such as communication cost, computation cost, network cost, as well as bee degradation are applied in the suggested work.

All activities are executed in parallel on processing engines, which improves performance and strengthens distributed task allocation. Processing units with low execution costs can complete jobs in large numbers and in a short amount of time. As indicated in Table 8, the total incurred cost on task execution is utilized to determine system cost. The suggested work is compared to other current techniques with concerns, and it is discovered that the ABC algorithm helps to obtain optimal system cost to construct a resilient distributed environment, as shown in Table 9.

Table 10 compares the before and after results of total energy consumption during task execution. It shows that total cost is reduced by the proposed ABC algorithm. The simulation results of before and after optimization are shown graphically in Figure 3.

Table 7. Comparative study of projected work with existing approaches

Existing Approaches	References	Purpose	Assumption	Drawbacks
Artificial intelligence	[4]	This used list-based heterogeneous earliest finish time (HEFT) algorithm to reduce cost by minimizing energy consumption rate.	Assume to reduce system cost	Network cost is not used during the computation of cost
Communication link sum (CLS)	[19]	To reduce the inter-processor communication to minimize the system cost for task allocation in distributed computing systems	Assume to reduce system cost	Network cost is not used during the computation of system cost. On the other side, this approach follows a static task allocation policy
Enhanced PSO	[27]	Proposed a load balancing mutation particle swarm optimization (LBMPSTO) to allocate the best resources to tasks for maintaining execution time, transmission cost, and makespan.	Assume to improve efficiency by allocating data with all resources at a low cost	network cost parameter is avoided here in the data allocation perspective
Proposed Approach		Work to reduce overall system cost using artificial bee colony approach for DAG	Assume to reduce system cost by learning, adapting, and updating behavior to achieve performance in distributed computing	does not consider fault tolerance part to adjust the load

Table 8. Optimal task allocation

Optimal allocation		Total execution cost	System cost
Tasks	Processing engines		
t ₃ , t ₆ , t ₇ , t ₁₀	PE1	191	396
t ₄ , t ₈ , t ₉	PE2	140	
t ₁ , t ₂ , t ₅	PE3	65	

Table 9. Assessment of system cost parameter with other methods

Proposed algorithm			Hamed algorithm [28]			Yadav algorithm [27]		
Tasks	Processing engines	System cost	Tasks	Processing engines	System cost	Tasks	Processing engines	System cost
t ₃ , t ₆ , t ₇ , t ₁₀	PE1	396	t ₄ , t ₇	PE1	459	t ₅ , t ₇	PE1	528
t ₄ , t ₈ , t ₉	PE2		t ₂ , t ₃ , t ₈ , t ₉	PE2		t ₂ , t ₃ , t ₈ , t ₉	PE2	
t ₁ , t ₂ , t ₅	PE3		t ₁ , t ₅ , t ₆	PE3		t ₁ , t ₄ , t ₆	PE3	

Table 10. Earlier and subsequent total cost results in mJ

Query/Tasks	Total Cost Before Optimization in mJ	Total Cost After Optimization in mJ
1	6.5224	6.3487
2	10.0158	9.5496
3	9.2899	8.9607
4	44.8201	44.6576
5	7.1796	6.9863
6	25.7880	25.5727
7	1.9467	1.9467
8	12.4964	12.0348
9	13.5141	13.1642
10	3.9973	3.5852
Average Cost of Tasks	13.56	13.28

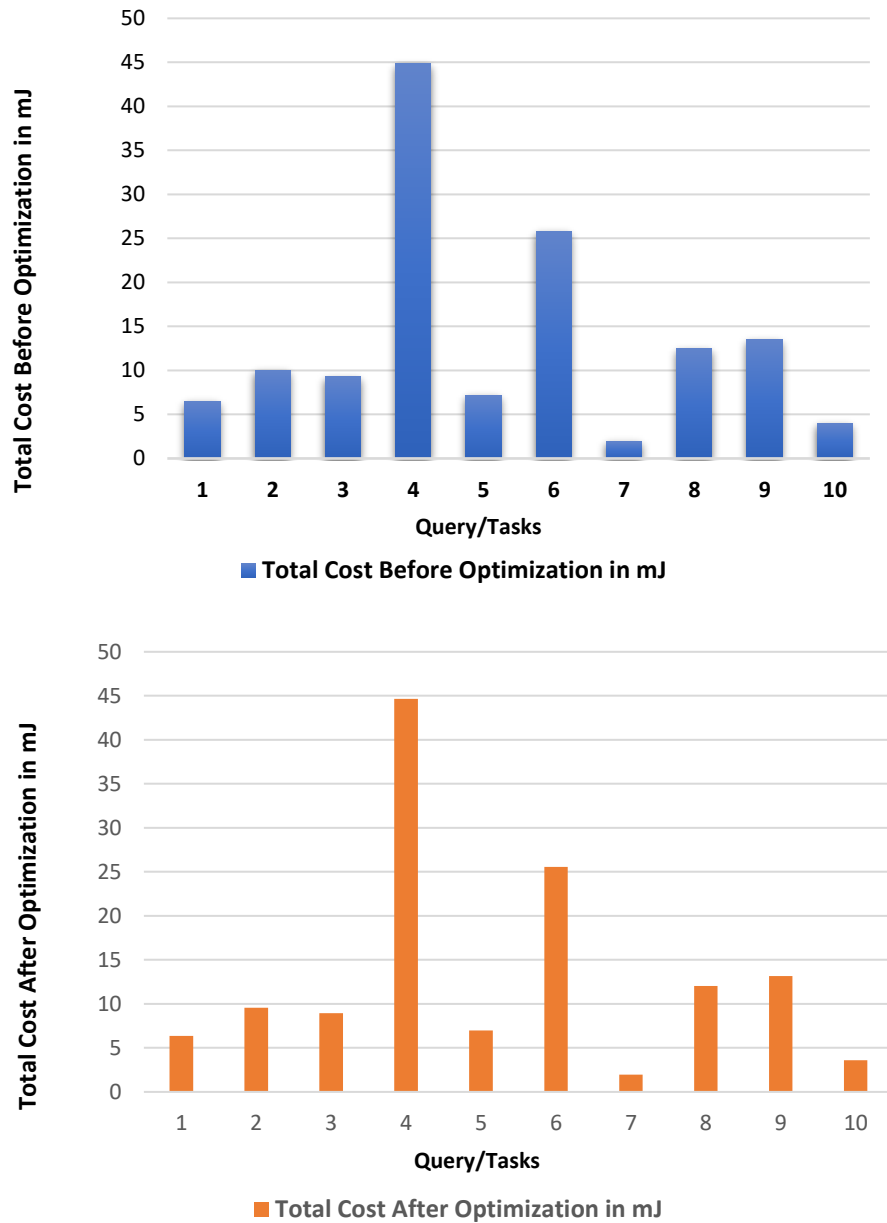


Figure 3. Energy consumption with and without optimization

In Table 11, the results are compared with a list-based task scheduling algorithm that employs artificial intelligence [4], and a task allocation model for system cost analysis that employs communication link sum (CLS) [27]. These techniques consider all indicators except network cost. In comparison to prior techniques, the proposed work saves 13.28 in overall costs, as shown in Figure 4. This method can easily allocate large data fragments and perform them fast and inexpensively. With this method, there are no extended waits, delays, or completion times, which lowers the performance of the distributed system.

Table 11. Comparison with existing techniques

Research technique used	Reduced total execution cost (%age)
Proposed work using artificial bee colony (ABC)	13.28
Existing work using AI [4]	60.6
Existing work using communication link sum (CLS) [27]	24

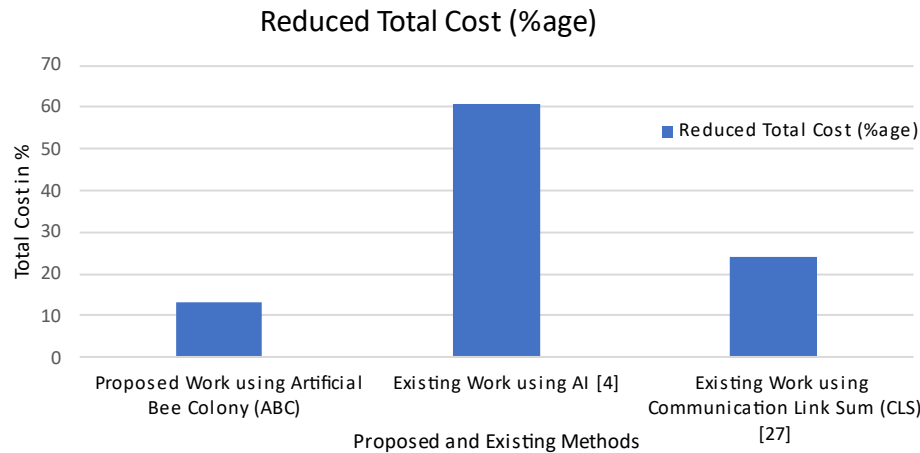


Figure 4. Comparative representation of techniques

5. CONCLUSION

We present a swarm intelligence-based artificial bee colony (ABC) method to reduce system execution costs and enhance data allocation in distributed systems in this study. It also makes it easier to trace the degradation loss of bees by subtracting equivalent cost units from the overall cost. When compared to previous approaches, the ABC algorithm was found to considerably lower total execution costs and improve system efficiency. Network expenses are not utilized to calculate system costs, according to previous studies. As a result, past results used to perform tests are inaccurate. This cost allocation model takes into account all expenses incurred during data processing in a distributed system. In the future, attempts might be made to bring new approaches to enhance data allocation by focusing more on fault tolerance in distributed computing systems.




REFERENCES

- [1] S. Tarun, R. S. Bath, and S. Kaur, "A novel fragmentation scheme for textual data using similarity-based threshold segmentation method in distributed network environment," *International Journal of Computer Networks and Applications*, vol. 7, no. 6, 231, Dec. 2020, doi: 10.22247/ijcna/2020/205322.
- [2] S. Tarun, R. S. Bath, and S. Kaur, "A review on fragmentation, allocation and replication in distributed database systems," in *International Conference on Computational Intelligence and Knowledge Economy*, Dec. 2019, pp. 538–544., doi: 10.1109/ICCIKE47802.2019.9004233.
- [3] A. Osman, A. Sagahyoon, R. Aburukba, and F. Aloul, "Optimization of energy consumption in cloud computing datacenters," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 686–698, Feb. 2021, doi: 10.11591/ijece.v11i1.pp686-698.
- [4] Akanksha, "List-based task scheduling algorithm for distributed computing system using artificial intelligence," in *Advances in Intelligent Systems and Computing*, vol. 941, Springer International Publishing, 2020, pp. 1006–1014, doi: 10.1007/978-3-030-16660-1_98.
- [5] A. Gandomi, A. Movaghar, M. Reshadi, and A. Khademzadeh, "Designing a MapReduce performance model in distributed heterogeneous platforms based on benchmarking approach," *The Journal of Supercomputing*, vol. 76, no. 9, pp. 7177–7203, Sep. 2020, doi: 10.1007/s11227-020-03162-9.
- [6] N. Lotfi, "Data allocation in distributed database systems: a novel hybrid method based on differential evolution and variable neighborhood search," *SN Applied Sciences*, vol. 1, no. 12, Dec. 2019, doi: 10.1007/s42452-019-1787-3.
- [7] R. Tariq, F. Aadil, M. F. Malik, S. Ejaz, M. U. Khan, and M. F. Khan, "Directed acyclic graph based task scheduling algorithm for heterogeneous systems," in *Advances in Intelligent Systems and Computing*, vol. 869, Springer International Publishing, 2019, pp. 936–947., doi: 10.1007/978-3-030-01057-7_69.
- [8] S. Sandokji and F. Eassa, "Communication and computation aware task scheduling framework toward exascale computing," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, pp. 119–128, 2019, doi: 10.14569/IJACSA.2019.0100718.
- [9] I. O. Hababeh and N. Bowring, "A method for fragment allocation design in the distributed database systems," *The Sixth Annual UAE*, pp. 4–12, 2005.
- [10] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, Feb. 2019, doi: 10.1109/TMC.2018.2831230.
- [11] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113–128, Feb. 2018, doi: 10.1109/TEVC.2016.2623803.
- [12] M.-H. Malekloo, N. Kara, and M. El Barachi, "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments," *Sustainable Computing: Informatics and Systems*, vol. 17, pp. 9–24, Mar. 2018, doi: 10.1016/j.suscom.2018.02.001.




- [13] S. Vakiliinia, B. Heidarpour, and M. Cheriet, "Energy efficient resource allocation in cloud computing environments," *IEEE Access*, vol. 4, pp. 8544–8557, 2016, doi: 10.1109/ACCESS.2016.2633558.
- [14] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Reliability and energy efficiency in cloud computing systems: survey and taxonomy," *Journal of Network and Computer Applications*, vol. 74, pp. 66–85, Oct. 2016, doi: 10.1016/j.jnca.2016.08.010.
- [15] Z. Long and W. Ji, "Power-efficient immune clonal optimization and dynamic load balancing for low energy consumption and high efficiency in green cloud computing," *Journal of Communications*, vol. 11, no. 6, pp. 558–563, 2016, doi: 10.12720/jcm.11.6.558-563.
- [16] T. Kaur and I. Chana, "Energy efficiency techniques in cloud computing: a survey and taxonomy," *ACM Computing Surveys*, vol. 48, no. 2, pp. 1–46, Nov. 2015, doi: 10.1145/2742488.
- [17] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, May 2012, doi: 10.1007/s11227-010-0421-3.
- [18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.
- [19] A. Scionti, K. Goga, F. Lubrano, and O. Terzo, "Towards energy efficient orchestration of cloud computing infrastructure," in *Advances in Intelligent Systems and Computing*, vol. 772, Springer International Publishing, 2019, pp. 172–183, doi: 10.1007/978-3-319-93659-8_15.
- [20] N. Khan and R. Shrestha, "Optimizing power and energy efficiency in cloud computing," in *9th International Conference on Cloud Computing and Services Science*, 2019, pp. 380–387., doi: 10.5220/0007723503800387.
- [21] C. Tang, S. Xiao, X. Wei, M. Hao, and W. Chen, "Energy efficient and deadline satisfied task scheduling in mobile cloud computing," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jan. 2018, pp. 198–205, doi: 10.1109/BigComp.2018.00037.
- [22] N. Liu, Z. Dong, and R. Rojas-Cessa, "Task scheduling and server provisioning for energy-efficient cloud-computing data centers," in *IEEE 33rd International Conference on Distributed Computing Systems Workshops*, Jul. 2013, pp. 226–231, doi: 10.1109/ICDCSW.2013.68.
- [23] H. Zhao, G. Qi, Q. Wang, J. Wang, P. Yang, and L. Qiao, "Energy-efficient task scheduling for heterogeneous cloud computing systems," in *IEEE 21st International Conference on High Performance Computing and Communications*, Aug. 2019, pp. 952–959, doi: 10.1109/HPCC/SmartCity/DSS.2019.00137.
- [24] R. Li, N. Mi, M. Riedewald, Y. Sun, and Y. Yao, "A case for abstract cost models for distributed execution of analytics operators," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10440, Springer International Publishing, 2017, pp. 149–163., doi: 10.1007/978-3-319-64283-3_11.
- [25] C.-C. Hsieh and Y.-C. Hsieh, "Reliability and cost optimization in distributed computing systems," *Computers and Operations Research*, vol. 30, no. 8, pp. 1103–1119, Jul. 2003, doi: 10.1016/S0305-0548(02)00058-8.
- [26] K. A. Shakil, M. Alam, and S. Khan, "A latency-aware max-min algorithm for resource allocation in cloud," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 671–685, Feb. 2021, doi: 10.11591/ijece.v11i1.pp671-685.
- [27] P. K. Yadav, M. P. Singh, and K. Sharma, "An optimal task allocation model for system cost analysis in heterogeneous distributed computing systems: a heuristic approach," *International Journal of Computer Applications*, vol. 28, no. 4, pp. 30–37, Aug. 2011, doi: 10.5120/3374-4664.
- [28] A. Y. Hamed, "Task allocation for maximizing reliability of distributed computing systems using genetic algorithms," *International Journal of Computer Networks and Wireless Communications (IJCNWC)*, vol. 2, no. 5, pp. 560–569, 2012.

BIOGRAPHIES OF AUTHORS







Sashi Tarun    is a Ph.D. Research Scholar in the School of Computer Science and Engineering at Lovely Professional University, Punjab, India. He has completed M.Tech. Computer Science from Jamia Hamdard University, New Delhi. His research interests are distributed systems, cloud systems, database systems, computer networks, artificial intelligence, and machine learning. He has several papers on his credit. He has 7 years of teaching experience as an Assistant Professor. He can be contacted by email: sashitarun79@gmail.com.







Mithilesh Kumar Dubey    is working as a Professor in the School of Computer Application of Lovely Professional University Jalandhar Punjab India. He has handsome experience in the software industry as well as in Research development. He has published many articles at international level in reputed journals. He can be contacted by email: mithilesh.21436@lpu.co.in



Ranbir Singh Batth     is working as an associate professor in the School of Computer Science and Engineering and also serves as a coordinator for international relations at Lovely Professional University, Punjab, India. In 2018, he received his Ph.D. in computer science and engineering from Punjab Technical University, India. His research interests include wireless sensor networks, cloud computing, network security, ad-hoc networks, machine learning, deep learning, wireless communications, and mobile computing. He is a senior member of IEEE and faculty coordinator of the ACM research chapter. He can be contacted by email: ranbir.21123@lpu.co.in.



Sukhpreet Kaur     is working as Associate Professor in the CSE department at Chandigarh Engineering College, Landran, Mohali. She has in total 15 years of vast experience in teaching and research. She has done a Ph.D. in CSE from I.K Gujral Punjab Technical University, Jalandhar, and has done her Masters in Technology in CSE from GNDEC, Ludhiana. The various research areas in which she worked include image processing, artificial intelligence, and computer vision. She can be contacted by email: sukhpreet.4479@cgc.edu.in.