A

Major Project Report

On

Cost Effective Data Placement in Edge Storage Systems With Erasure Code

submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

by

Boini Pavan

(20EG105305)

CH Manipreeth

(20EG105309)

V V S S Chandra

(20EG105351)

Under The Guidance of

Mrs. ASMA TAHSEEN

Assistant Professor,

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG UNIVERSITY

VENKATAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S 500088

(2023-24)

ANURAG UNIVERSITY | SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**Cost Effective Data Placement in Edge Storage Systems With Erasure Code**" that is being submitted by **Boini Pavan, CH Manipreeth, V V S S Chandra** bearing the Hall Ticket number **20EG105305, 20EG105309, 20EG105351** in partial fulfilment of the requirements for the award of degree of the Bachelor of Technology in Computer Science and Engineering in Anurag University is a record of bonafide work carried out by them under my guidance and supervision from December 2023 to April 2024.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this Report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of The Supervisor	**Dean, CSE**
Mrs. Asma Tahseen	Dr. G Vishnu Murthy
Assistant Professor	M. Tech, Ph. D
Department of CSE	Professor

External Examiner

DECLARATION

We hereby declare that the project work entitled **"Cost Effective Data Placement in Edge Storage Systems With Erasure Code"** submitted to the **Anurag University** in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology(B.Tech)** in **Computer Science and Engineering** is a record of an original work done by us under the guidance of **Mrs. Asma Tahseen, Assistant Professor** and this project work has not been submitted to any other university for the award of any other degree or diploma.

Place: Anurag University, Hyderabad

Date:

Boini Pavan
(20EG105305)

CH Manipreeth
(20EG105309)

V V S S Chandra
(20EG105351)

ACKNOWLEDGEMENT

We would like to express my sincere thanks and deep sense of gratitude to project supervisor **Mrs Asma Tahseen, Assistant Professor, Department of CSE** for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like express our special thanks to **Dr. V. Vijaya Kumar, Dean School of Engineering, Anurag University,** for his encouragement and timely support in our B. Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy, Dean, Dept. of CSE, Anurag University.** We also express our deep sense of gratitude to **Dr. V V S S S Balaram, Academic co-ordinator**, **Dr. Shyam Prasad,** Assistant Professor. Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

Boini Pavan

(20EG105305)

CH Manipreeth

(20EG105309)

V V S S Chandra

(20EG105351)

ABSTRACT

Title: “COST EFFECTIVE DATA PLACEMENT IN EDGE STORAGE SYSTEMS WITH ERASURE CODE”

Edge computing, as a new computing paradigm, brings cloud computing’s computing and storage capacities to network edge for providing low latency services for users. The networked edge servers in a specific area constitute edge storage systems (ESSs), where popular data can be stored to serve the users in the area. Most existing studies of ESSs focus on the storage of data replicas in the system to ensure low data retrieval latency for users. However, replica-based edge storage strategies can easily incur high storage costs. Especially those that do not require real-time access at the edge, e.g., system upgrade files, popular app installation files, videos in online games. It may not even be possible due to the constrained storage resources on edge servers. In this project, we make the first attempt to investigate the use of erasure codes in cost-effective data storage at the edge. The focus is to find the optimal strategy for placing coded data blocks on the edge servers in an ESS, aiming to minimize the storage cost while serving all the users in the system. We model Erasure Coding based Edge Data Placement (EC-EDP). Each data blocks which is stored in server node are encrypted with different algorithms and generate two keys (AES, DES) which will reduce data attacks.

Keyword - Encryption, AES, DES, Nodes, Edge Computing. Erasure Coding

TABLE OF CONTENTS

LIST OF FIGURES

Abbreviations Used

ESS: Edge Storage System

EC: Erasure Coding

AES: Advanced Encryption Standard

DES: Data Encryption Standard

1.INTRODUCTION

Edge computing is a paradigm that brings computation and data storage closer to the location where it is needed, rather than relying on a centralized data processing facility. This approach aims to reduce latency and bandwidth usage by processing data locally, at or near the source of data generation, rather than sending it to a distant data centre. At its core, edge computing leverages a network of decentralized nodes or servers situated at the "edge" of the network, closer to end-users or data sources. These edge devices can range from IoT devices, such as sensors and cameras, to edge servers deployed in close proximity to users or in remote locations like factories, warehouses, or vehicles. One of the key benefits of edge computing is its ability to process data in real-time or near real-time, enabling faster response times and improved performance for applications that require low latency, such as autonomous vehicles, augmented reality (AR), and industrial automation. By processing data locally, edge computing also reduces the amount of data that needs to be transmitted over the network, alleviating bandwidth constraints and lowering operational costs. Moreover, edge computing enhances data privacy and security by keeping sensitive information closer to its source and reducing the risk of data breaches during transit to centralized data centres. This distributed architecture also increases resilience and fault tolerance since edge nodes can continue to operate independently even if connectivity to the central cloud infrastructure is lost. However, deploying and managing edge computing infrastructure comes with its challenges, including ensuring interoperability between diverse edge devices, maintaining security across a distributed network, and managing scalability and resource constraints on edge devices with limited computing power and storage capacity. Despite these challenges, the adoption of edge computing is rapidly growing across various industries, driven by the proliferation of IoT devices, the need for real-time data processing, and the rise of applications requiring low latency and high bandwidth. As edge computing continues to evolve, it is expected to play a crucial role in shaping the future of computing infrastructure and enabling innovative use cases

across diverse domains.

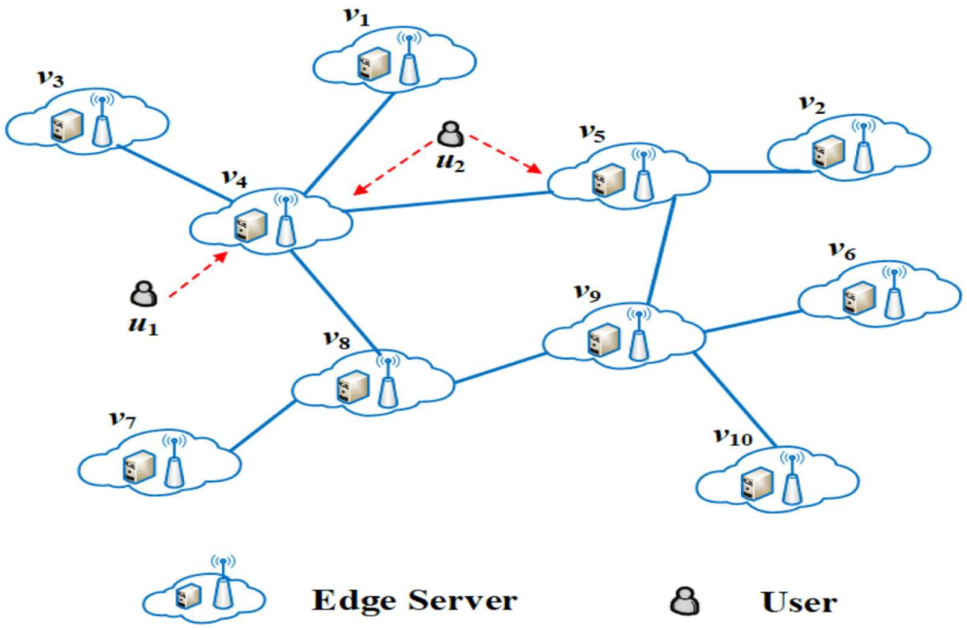

Fig 1.1 Example of Edge Storage System

Erasure coding is a method used in computer science and data storage to provide redundancy and fault tolerance by distributing data across a network or storage system in a way that allows the original data to be reconstructed even if some of the data is lost or corrupted. It achieves this by creating additional pieces of data, known as parity or redundant data, and distributing them alongside the original data.

Unlike traditional data replication, where multiple copies of the same data are stored across different locations, erasure coding divides the data into smaller fragments and generates parity fragments based on mathematical algorithms. These parity fragments contain redundant information calculated from the original data, enabling the recovery of lost or corrupted fragments without the need for complete replication.

Erasure coding offers several advantages over traditional replication methods. It reduces storage overhead by requiring fewer redundant copies of the data, making it more efficient in terms of storage space utilization. Additionally, erasure coding provides better fault tolerance and resilience to data loss, as it can tolerate multiple failures without losing the ability to reconstruct the original data.

Overall, erasure coding is a powerful technique for ensuring data integrity and availability in distributed storage systems, cloud computing environments, and communication networks, where data loss or corruption can occur due to various factors such as hardware failures, network errors, or malicious attacks.

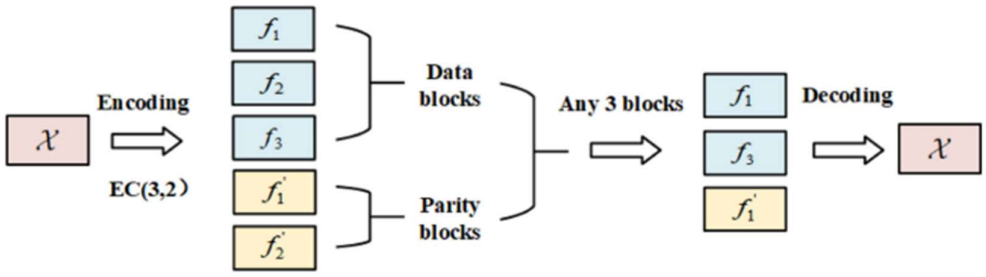

Fig 1.2 Example of Erasure Coding (3,2)

1.1. Motivation

Motivation for utilizing edge computing in the distribution of large upgrade packages, such as Microsoft's Windows-10 updates, stems from the need to optimize network resource consumption and reduce distribution costs. Microsoft has adopted peer-to-peer distribution alongside client-server models since Windows-10 to achieve these goals. By leveraging edge servers situated at the network's periphery, companies like Microsoft can minimize the expense associated with transferring data from centralized storage facilities to end-users. For instance, Amazon Web Services charges a substantial fee of up to US$0.11 per gigabyte for data transfer out of its S3 storage facilities to the internet.

In the context of edge computing, an Edge Storage System (ESS) composed of interconnected edge servers serves users within a specific area, such as New York's Central Business District (CBD). A straightforward approach to distributing large upgrade packages involves storing replicas of the data on each of the edge servers within the ESS. This allows users in the CBD to access nearby edge servers for downloading the upgrade package, thereby reducing latency.

However, this replication-based solution has notable drawbacks. Firstly, maintaining multiple replicas of the same data across numerous edge servers incurs significant storage costs for companies like Microsoft. Secondly, it fails to capitalize on the potential collaboration among edge servers to optimize data transmission. For example, in a network topology where data can be transmitted via two hops between edge servers, the direct exchange of data between servers can expedite delivery to users.

Therefore, there is a need for more efficient Edge Content Distribution Policies (EC-EDP) that leverage the collaborative capabilities of edge servers while minimizing storage costs. By optimizing data transmission pathways and enabling edge servers to share and exchange data among themselves, companies can enhance the efficiency of content distribution in edge computing environments. Such strategies can lead to improved network performance, reduced latency, and lower distribution costs for large-scale data delivery scenarios, benefiting both service providers and end-users alike.

1.2 Methodology

Achieving cost-effective data placement in edge storage systems utilizing erasure coding requires a comprehensive methodology integrating various essential components. Firstly, a thorough understanding of erasure coding's intricacies is vital. Erasure coding offers fault tolerance and data redundancy while minimizing storage overhead by generating parity fragments. This understanding lays the foundation for subsequent decisions. Next, it's crucial to grasp the unique attributes of the edge environment, including available bandwidth, storage capacity, and latency requirements. This comprehension informs the development of tailored strategies for efficient data placement. Cost factors such as hardware, maintenance, and data transfer expenses must be accurately modelled to provide a quantitative basis for optimization. Defining optimization objectives, whether focused on minimizing storage costs, enhancing reliability, or reducing latency, helps shape data placement strategies.

Various strategies can be employed, including geographical-based placement, load balancing, predictive placement, or hybrid approaches, each aimed at striking a balance between cost and performance. Algorithmic techniques such as integer programming or heuristic algorithms can be utilized to facilitate efficient data placement, considering factors like data access patterns and resource constraints. Simulation and evaluation

play a crucial role in validating the effectiveness of the methodology. Fine-tuning based on simulation results ensures optimization before deployment. Deployment of the optimized methodology is followed by continuous monitoring to ensure ongoing cost-effectiveness and performance alignment with objectives. This iterative process allows for adjustments and improvements over time. In essence, a holistic approach that considers erasure coding principles, edge environment characteristics, cost factors, optimization objectives, tailored strategies, algorithmic techniques, simulation, evaluation, and continuous monitoring ensures that data placement in edge storage systems with erasure code is optimized for cost-effectiveness without compromising reliability or performance.

1.3 Problem Definition

The problem of cost-effective data placement in edge storage systems with erasure code arises from the need to balance multiple objectives, including minimizing storage costs while ensuring data reliability, availability, and low latency in edge computing environments. With the proliferation of Internet of Things (IoT) devices and the exponential growth of data generated at the network edge, traditional data replication approaches become cost-prohibitive due to the large storage overhead. Erasure coding offers a more storage-efficient solution by distributing redundant data across multiple nodes while maintaining fault tolerance. However, optimizing data placement in edge storage systems with erasure code presents several challenges. These include dynamically varying network conditions, limited storage and computational resources at edge nodes, intermittent connectivity, and heterogeneous edge environments. Moreover, the cost-effectiveness of data placement strategies must be carefully evaluated considering factors such as hardware costs, maintenance expenses, data transfer costs, and energy consumption. Thus, the problem involves designing methodologies and algorithms that intelligently place data in edge storage systems to minimize costs while meeting performance requirements and ensuring data reliability in the face of node failures and network disruptions. Addressing these challenges is crucial for enabling efficient and scalable edge computing infrastructures capable of supporting diverse applications and workloads.

1.4 Objective of The Project

The objective of the project for Cost Effective Data Placement in Edge Storage Systems With Erasure Code is multifaceted and comprehensive. Firstly, it aims to address the escalating storage demands and cost challenges associated with edge computing environments by leveraging erasure coding techniques. Through extensive research and analysis, the project endeavours to understand the intricacies of erasure coding and its applicability in edge scenarios, considering factors such as limited storage resources, intermittent connectivity, and varying network conditions. Secondly, the project aims to develop novel algorithms and methodologies tailored to optimize data placement strategies in edge storage systems. These strategies will prioritize minimizing storage costs while ensuring data reliability, availability, and low latency, thereby enhancing the efficiency and scalability of edge computing infrastructures. Thirdly, the project seeks to validate the effectiveness of the proposed methodologies through rigorous simulation and experimentation in both synthetic and real-world edge environments. By evaluating performance metrics such as storage efficiency, data availability, and cost-effectiveness, the project aims to provide empirical evidence of the benefits of employing erasure coding for data placement in edge storage systems. Ultimately, the objective is to deliver practical guidelines and solutions that empower organizations to deploy cost-effective edge computing applications while maintaining high levels of data integrity and accessibility in diverse edge scenarios.

2.LITERATURE SURVEY

2.1 Existing Systems

Author(s)	Strategies	Advantages	Disadvantages
Alberto ceselli, Marco premoli ,Stefano secci	Strategically determine where to install cloudlet facilities based on factors like user density, service demand, and network conditions.	Optimal cloudlet placement and access point-cloudlet assignment can lead to improved user satisfaction by reducing latency and improving service reliability.	Implementing and managing a mobility-aware design adds complexity to the overall system.
Xiaoyu Xia, Feifei chen, Qiang he	Utilize content caching mechanisms at edge servers to store frequently accessed data locally	Users experience faster response times, improving the overall quality of service.	Content caching at edge servers may introduce additional resource overhead, requiring careful management to avoid excessive storage usage.
R. Luo, H. Jin, Q. He, S. Wu, Z. Zeng, and X. Xia,	Implement a collaborative approach where edge servers work together to identify and eliminate redundant data.	Optimization under latency constraints ensures that deduplication processes do not compromise user experience.	Efficient algorithms and computational optimization are required to manage the deduplication processes without impacting performance.

J. Xie, C. Qian, D. Guo, X. Li, S. Shi, and H. Chen,	Simplify the routing process by limiting it to a single overlay hop, contributing to faster data access.	GRED's design promotes efficient load balancing among edge servers, preventing server overloads	Implementing a DHT system with virtual-space and SDN integration may introduce complexity in the development and deployment phases.

2.2 Proposed Systems

The proposed system for Cost Effective Data Placement in Edge Storage Systems With Erasure Code will be a comprehensive framework designed to optimize data placement while leveraging erasure coding techniques in edge environments. It will consist of several interconnected components. Firstly, a distributed network of Edge Storage Nodes will be strategically deployed across edge locations, each equipped with storage resources and erasure coding mechanisms to ensure fault tolerance and minimize storage overhead. Secondly, a Centralized Management System will serve as the orchestrator, coordinating data placement decisions across the network. This system will employ sophisticated optimization algorithms that consider factors such as network conditions, storage capacity, and access patterns to determine the most cost-effective data placement strategies. Thirdly, a Monitoring and Optimization Module will continuously monitor system performance metrics, including data availability, latency, and resource utilization, and dynamically adjust data placement strategies in response to changing conditions. Additionally, a Simulation Environment will be utilized to simulate various edge scenarios and validate the effectiveness of proposed methodologies before deployment in real-world environments. By integrating these components, the proposed system aims to provide a scalable, efficient, and cost-effective solution for managing data in edge storage systems with erasure code, facilitating the deployment of edge computing applications across diverse use cases.

3.ANALYSIS

3.1 SOFTWARE REQUIREMENTS SPECIFICATION

OS Windows, Mac

java

spring suite tool 4

3.2 PURPOSE

The project "Cost Effective Data Placement in Edge Storage Systems With Erasure Code" is driven by the need to efficiently manage data in edge computing environments. Edge computing, characterized by its distributed nature and proximity to end-users or data sources, presents challenges in terms of storage management, particularly in ensuring data reliability, availability, and low latency. By leveraging erasure coding techniques, the project aims to address these challenges while minimizing storage costs. Erasure coding provides fault tolerance and data redundancy while reducing storage overhead compared to traditional replication methods. Thus, the project seeks to exploit the benefits of erasure coding to optimize data placement in edge storage systems.The primary objective is to develop a systematic methodology and algorithmic framework tailored to the unique characteristics of edge environments. This involves understanding factors such as available bandwidth, storage capacity, and latency requirements, which vary significantly in edge computing setups compared to centralized data centers.

Through this methodology, the project aims to optimize data placement strategies that balance cost-effectiveness with performance. These strategies may include geographical-based placement, load balancing, predictive placement, or hybrid approaches, customized to achieve specific optimization objectives such as minimizing storage costs while ensuring data reliability and low latency.Ultimately, the project's goal is to enable the cost-effective deployment of edge computing applications across various industries. By enhancing scalability, performance, and resource utilization in edge storage systems, the project aims to facilitate the widespread adoption of edge computing, unlocking its potential to support innovative applications and services at the network edge.

3.3 SCOPE

The scope of the project "Cost Effective Data Placement in Edge Storage Systems With Erasure Code" encompasses several critical aspects aimed at efficiently managing data in edge computing environments while leveraging erasure coding techniques.The project entails developing algorithms and methodologies for optimizing data placement strategies tailored to the unique characteristics of edge environments. This involves considering factors such as network conditions, storage capacity, and performance requirements specific to edge computing setups. For example, data placement strategies may need to account for limited bandwidth, varying latency, and heterogeneous edge device capabilities .The algorithms and methodologies developed in the project aim to optimize data placement decisions to achieve cost-effectiveness while meeting performance objectives. This may involve determining the optimal placement of data replicas across edge nodes, considering factors like data access patterns and resource constraints.

Additionally, the project may explore dynamic data placement strategies that adapt to changing network conditions and workload demands in real-time. This could involve developing predictive algorithms that anticipate future data access patterns and adjust data placement accordingly to optimize resource utilization. Overall, the scope of the project encompasses research, algorithm development, and methodology design aimed at optimizing data placement in edge storage systems with erasure code. By addressing these key aspects, the project aims to enhance the efficiency, reliability, and cost-effectiveness of edge computing environments, enabling the seamless deployment of applications and services at the network edge.

3.4 OVERALL DESCRIPTION

The project for "Cost Effective Data Placement in Edge Storage Systems With Erasure Code" is a comprehensive endeavor aimed at developing a framework to optimize data storage in edge computing environments. It centers on leveraging erasure coding techniques to minimize storage costs while maintaining data reliability and achieving low latency.A systematic approach guides the project, beginning with thorough research to understand erasure coding techniques and their applicability in edge computing. This foundational understanding serves as the basis for designing and implementing algorithms and methodologies tailored to the unique characteristics of edge

environments.The project considers various factors critical to edge computing, such as network conditions, storage capacity, and performance requirements. By analyzing these factors, the project aims to develop scalable and efficient solutions for managing data placement in edge storage systems.Central to the project's objectives is the optimization of data placement strategies. This involves determining the most effective distribution of data across edge nodes to achieve cost-effectiveness while meeting performance objectives. The algorithms and methodologies developed will take into account dynamic factors such as fluctuating network conditions and varying workload demands.

Ultimately, the project aims to facilitate the cost-effective deployment of edge computing applications by providing robust solutions for data storage optimization. By minimizing storage costs through erasure coding and ensuring data reliability and low latency, the project contributes to enhancing the efficiency and scalability of edge computing environments. Through its research, design, and implementation efforts, the project aims to provide valuable insights and tools for organizations seeking to leverage edge computing technologies. By addressing the complexities of data placement in edge storage systems, the project lays the foundation for the widespread adoption and success of edge computing applications across various industries.

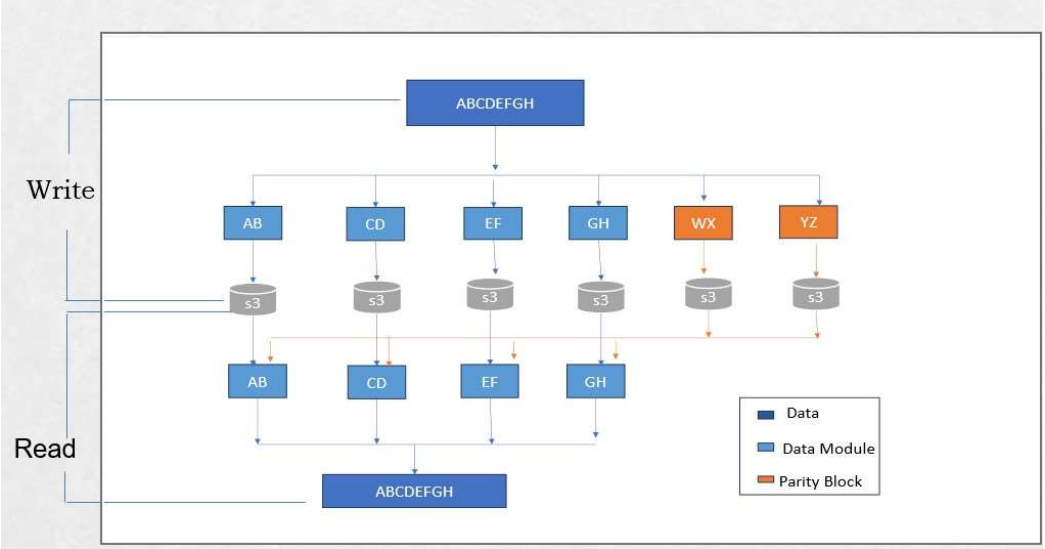

Fig 3.1 Project Flow Diagram

4. IMPLEMENTATION

4.1 Experiment Environment

Java:

Java's versatility stems from its platform independence, allowing code to run across various environments. Its object-oriented paradigm fosters modular, maintainable codebases, enabling smooth collaboration among developers. Java's robustness ensures stable, secure applications, while its extensive library ecosystem provides solutions for diverse development needs. Frameworks like Spring and libraries like Apache Commons further enhance productivity and scalability, enabling developers to efficiently build a wide range of projects, from web applications to enterprise systems. Java's widespread adoption and mature ecosystem make it invaluable for project development, empowering developers to create robust, scalable, and secure software solutions across different domains.

Springtoolsuite:

Spring Tool Suite (STS) stands out as an exceptional Integrated Development Environment (IDE) designed specifically for Spring Framework-based projects. It significantly simplifies the development process by offering a plethora of features tailored to the needs of Spring developers.At the core of STS's functionality is its support for Spring Boot, a popular framework for building stand-alone, production-grade Spring-based applications. STS seamlessly integrates with Spring Boot, providing developers with an intuitive environment for configuring and deploying their applications.One of the key advantages of STS is its support for popular build tools like

Maven and Gradle. This allows developers to manage project dependencies, build configurations, and package distributions effortlessly. By automating these project management tasks, STS boosts productivity and reduces the likelihood of errors during the development process.STS also offers a range of features aimed at improving the coding experience. These include code completion, syntax highlighting, and integrated debugging, which help developers write clean, efficient code and troubleshoot issues effectively.Moreover, STS boasts a robust ecosystem of tools, plugins, and extensions that further enhance its capabilities. Whether it's integrating with version control systems, accessing databases, or implementing continuous integration and deployment pipelines, STS provides developers with the necessary tools to build and deploy Spring applications with heightened efficiency and reliability.In essence, Spring Tool Suite is an indispensable tool for Spring developers, offering a comprehensive and streamlined development environment that empowers them to build high-quality, robust applications with ease.

4.2 Encryption Modules

4.2.1 Advanced Encryption Standard (AES)

AES is a symmetric encryption algorithm widely used for securing sensitive data in various applications, including communication, storage, and authentication. It was established by the U.S. National Institute of Standards and Technology (NIST) in 2001 as a replacement for the aging Data Encryption Standard (DES).AES operates on blocks of data, typically 128 bits in length, and supports key lengths of 128, 192, and 256 bits. The algorithm consists of several rounds of substitution, permutation, and mixing operations, known as the SubBytes, ShiftRows, MixColumns, and AddRoundKey steps. These operations create confusion and diffusion, making it highly resistant to cryptanalysis.One of AES's key strengths lies in its efficiency and speed, making it suitable for use in both hardware and software implementations. It is significantly faster than its predecessor, DES, while offering a higher level of security.AES's security is based on the strength of its key. With a sufficiently long and random key, AES is considered practically unbreakable through brute force attacks. However, its security can be compromised if weak keys are used or if vulnerabilities are discovered in the implementation.AES has become the de facto standard for encryption worldwide, adopted by governments, businesses, and individuals alike. It is used to secure sensitive

information in various applications, including secure communication protocols like SSL/TLS, encrypted file systems, and secure messaging apps.Overall, AES's combination of security, efficiency, and versatility has cemented its position as one of the most widely used encryption algorithms, playing a crucial role in safeguarding digital communication and data privacy.

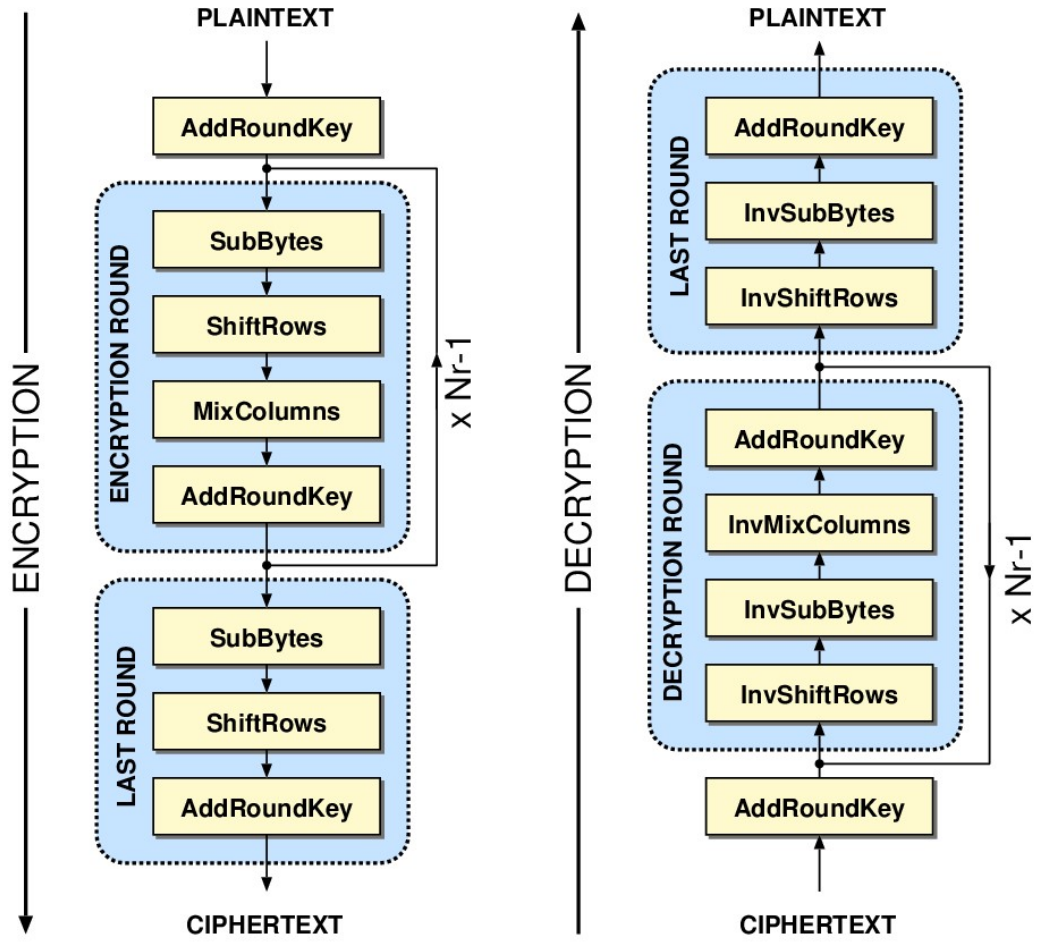

Fig 4.1 AES Diagram

4.2.2 Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric-key encryption algorithm widely used for securing data transmission and storage. Developed by IBM in the early 1970s, DES was adopted as a federal standard for encryption in the United States and became

one of the most widely used encryption methods worldwide.DES operates on blocks of plaintext data, typically 64 bits in size, and employs a symmetric-key approach, meaning the same key is used for both encryption and decryption. The key length for DES is 56 bits, although with parity bits, the total key length is 64 bits. This relatively short key length has led to concerns about the security of DES, especially with the advancement of computing power and the emergence of more sophisticated cryptographic attacks.The algorithm itself consists of multiple rounds of substitution and permutation operations, known as the Feistel network. During each round, the plaintext block is divided into two halves, which undergo a series of operations involving substitution and permutation based on the encryption key. The output of each round is combined with the other half of the plaintext block, and the process is repeated for a set number of rounds.

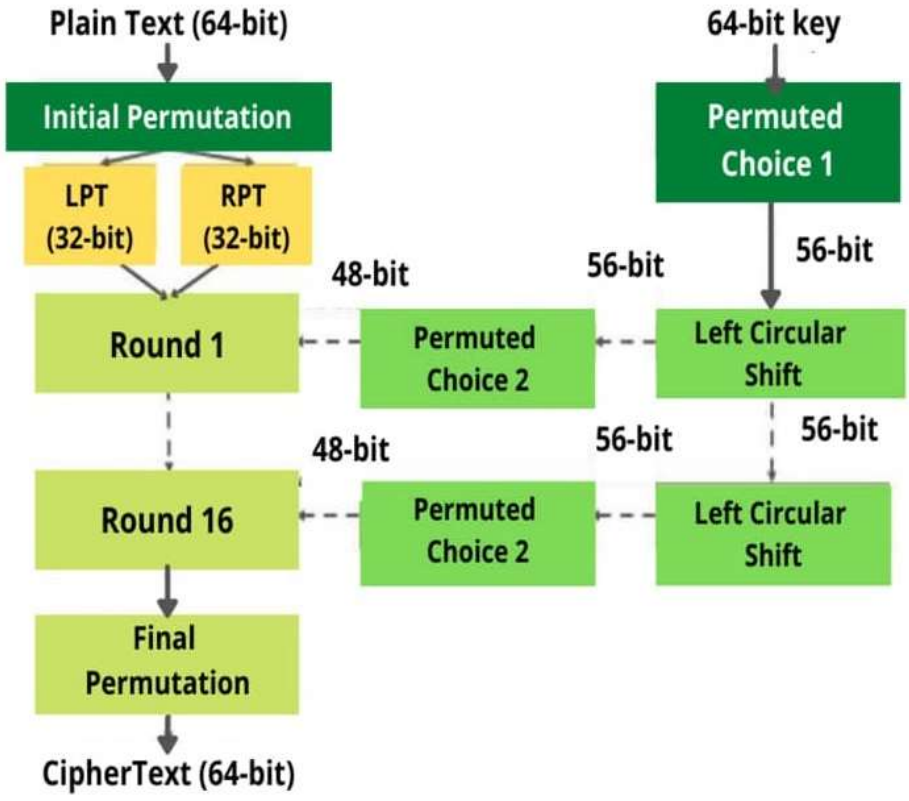

Fig 4.2 DES Diagram

4.3 SAMPLE CODE

```
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import java.util.ArrayList;
import java.util.Arrays;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
@SuppressWarnings("unused")
public class MajorProjectCode {
        static int availrows,n;
static boolean[] booleanArray= {true,true,true,true,true,true};
static double ceilSqrt;
static int[] p= {4, 2, 8, 6, 9, 1, 3, 7, 5, 2, 6, 4, 7, 9, 3, 1, 8, 5, 2, 6,
                 9, 7, 3, 5, 8, 4, 6, 2, 1, 9, 5, 8, 6, 1, 4, 2, 7, 9, 3, 5,
                 3, 1, 9, 7, 2, 6, 4, 8, 5, 3, 7, 8, 1, 2, 5, 4, 6, 9, 3, 2,
                 6, 4, 9, 5, 3, 7, 8, 1, 2, 5, 4, 6, 9, 3, 2, 6, 4, 9, 5, 3,
                 7, 8, 1, 2, 5, 4, 6, 9, 3, 2, 6, 4, 9, 5, 3, 7, 8, 1, 2, 5,
};
@SuppressWarnings({ "unused" })
public static void main(String[] args) throws Exception{
        List<Integer> l=new ArrayList<>();
    @SuppressWarnings("resource")
        Scanner scanner=new Scanner(System.in);
   // Step 1: Take string as input
    System.out.print("Enter a string(Data): ");
    String inputString = scanner.nextLine();
```

```
// Step 2: Find ceil value of square root of length of string
    ceilSqrt = Math.ceil(Math.sqrt(inputString.length()));
      // Step 3: Create n*n matrix (double data type) where n=ceil value of square root of length of string
     n = (int)ceilSqrt;
    double[][] matrix = new double[n][n];
    //erasure remaining
    double[][] erasurematrixremaining = new double[n][n];
      // Step 4: Assign relevant ascii values of string characters sequentially
    int index = 0;
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        if (index < inputString.length()) {
          matrix[i][j] = inputString.charAt(index);
          index++;
        } else {
          break;
        }
      }
      if (index >= inputString.length()) {
        break;
      }
    }

    // Step 5: If some part of matrix is unassigned then consider 'a' as parity fill remaining unassigned places with relevant ascii values
    int z=0;
    while (index < n * n) {
      int row = index / n;
      int col = index % n;
```

```
            matrix[row][col] =p[z];
            z++;
           index++;
          }
 //calling generateerasurematrix
double[][] erasurematrix=generateErasureMatrix();
double[][] codedmatrix = multiplyMatrices(erasurematrix, matrix);
//file storage
double[][] mat=new double[matrix.length][matrix[0].length];
int[] lengths = genDivideMatrix(matrix.length);
double[][] variable1 = null, variable2 = null, variable3 = null,
              variable4 = null, variable5 = null, variable6 = null;
int start = 0;
int end;
int ind = 0;
for (int i = 0; i < lengths.length; i++) {
  end = start + lengths[i];
  switch (ind) {
          case 0:
          variable1 = new double[lengths[i]][matrix[0].length];
          copyRows(codedmatrix, variable1, start, end);
          break;
          break;
        }
String str1 = matrixToString(variable1);
SecretKeySpec aesKeySpec = generateAESKey();
SecretKeySpec desKeySpec = generateDESKey();
String encrypted1 = encrypt(str1, aesKeySpec, desKeySpec);
 // Writing to files
```

```
writeFile("C:\\Users\\boini\\OneDrive\\Documents\\storagefolder\\file1.txt", str1);
// Retrieving data and storing in matrices
for(int i=0;i<booleanArray.length;i++)
 {
      System.out.println("Enter true if server "+i+" is available Else enter false");
      booleanArray[i]=Boolean.parseBoolean(scanner.nextLine());
  }
int countserversfailed=0;
for(int i=0;i<booleanArray.length;i++)
{
   if(booleanArray[i]==false)
   {
     countserversfailed++;
   }
}
if(countserversfailed>2)
{
    System.out.println("Sorry, number of servers failed is greater than parity       servers used. We cannot recover data.");
    System.exit(0);
}
int erasurestart=0,erasureend=lengths[0],eras=0;
String fileData1="" ;
if(booleanArray[0])
{

          fileData= readFile("C:\\Users\\boini\\OneDrive\\Documents
                                    \\storagefolder\\file1.txt");
   if(fileData1.trim().length()!=0&&eras<erasurematrixremaining.length)
```

```
            {
               for(inti=erasurestart;i<erasureend&&eras<erasurematrixremaining.
               length;i++)
            {
              for(intj=0;j<erasurematrix[0].length&&
                    eras<erasurematrixremaining.length;j++)
                    {
                        erasurematrixremaining[eras][j]=erasurematrix[i][j];
                    }
                    eras++;          }}}}
System.out.println("Based on Servers Required Erasure matrix is:");
for(int i=0;i<erasurematrixremaining.length ;i++)
{
    for(int j=0;j<erasurematrixremaining[0].length ;j++)
      {
          System.out.print(erasurematrixremaining[i][j]+"  ");
      }
System.out.println();
}
if(booleanArray[0])
stringToMatrix(decrypt(fileData1, aesKeySpec, desKeySpec),l);
int  k=0;
for(int i=0;i<mat.length;i++)
{
     for(int j=0;j<mat[0].length;j++)
       {
 mat[i][j]=l.get(k);
 k++;}}
// Find determinant
```

```
double det = determinant(erasurematrixremaining);
if(det==0)
{
        System.out.println("Determinant is zero i.e;matrix is scalar please consider
          differrent parity values in erasure matrix");
}
// Find adjoint
double[][] adj = adjoint(erasurematrixremaining);
double[][] result = multiplyMatrices(adj, mat);
// Convert the integers to relevant strings based on ASCII values
for (int i = 0; i < result.length; i++) {
         for (int j = 0; j < result[0].length; j++) {
       result[i][j]=result[i][j]/det;}}}
}
Public static String[] retrieveAvailableRows(String[] originalMatrix,double[][]
erasureMatrix,double[][] availableErasureMatrix) {
 Scanner scanner = new Scanner(System.in);
 String[] availableMatrix = new String[n];
 int l=0;
 for(inti=0;i<originalMatrix.length&&l<availableErasureMatrix.length; i++) {
          System.out.print("Is server(row) " + i + " available? (1 for       available, 0 for
not available): ");
          int choice = scanner.nextInt();
          if (choice == 1) {
             availableMatrix[l] = originalMatrix[i];
             availableErasureMatrix[l]=erasureMatrix[i];
               l++;} }
            return availableMatrix;
}
public static double[][] multiplyMatrices(double[][] matrix1, double[][] matrix2) {
```

```
    if (cols1 != rows2)
     {
      throw new IllegalArgumentException("Cannot multiply matrices  of incompatible
dimensions")
}
              double[][] result = new double[rows1][cols2];
              for (int i = 0; i < rows1; i++) {
         for (int j = 0; j < cols2; j++) {
              for (int k = 0; k < cols1; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];}}}
         return result;
}
//function to generate erasure matrix
public static double[][] generateErasureMatrix()
{
      int numRows =(int)ceilSqrt ;
      //firstMatrix.length;
      int numParityRows = calculateNumParityRows(numRows);
      double[][]parityMatrix=newdouble[numRows+
      numParityRows][numRows];
      // Generate identity matrix
      for (int i = 0; i < numRows; i++)
      {
          parityMatrix[i][i] = 1;
      }
      // Fill parity rows
      int x=0;
      for (int i = numRows; i < numRows + numParityRows; i++)
      {
```

```
            for (int j = 0; j < numRows; j++)
              {
                  parityMatrix[i][j] = p[x];
                  x++;}}
        return parityMatrix;
}
//matrix inverse
// Function to calculate the cofactor of matrix[i][j]
public static double[][] getCofactor(double[][] matrix, int i, int j)
{
        int n = matrix.length;
        double[][] cofactor = new double[n - 1][n - 1];
        int row = 0, col = 0;
        for (int k = 0; k < n; k++)
        {
    for (int l = 0; l < n; l++)
    {
            if (k != i && l != j)
            {
                  cofactor[row][col++] = matrix[k][l];
                  if (col == n - 1)
                  {
                      col = 0;
                      row++;}}}}
    return cofactor;
}
// Function to find the adjoint of a matrix
public static double[][] adjoint(double[][] matrix)
{
```

```
    int n = matrix.length;
    double[][] adjoint = new double[n][n];
    int sign;
    for (int i = 0; i < n; i++)
    {
  for (int j = 0; j < n; j++)
  {
        sign = ((i + j) % 2 == 0) ? 1 : -1;
        double[][] cofactor = getCofactor(matrix, i, j);
        adjoint[j][i] = sign * determinant(cofactor);
    } }
     return adjoint;
}
public static String encryptWithAES(String message, SecretKeySpec key) throws
Exception
{
    Cipher aesCipher = Cipher.getInstance("AES");
    aesCipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] encryptedBytes = aesCipher.doFinal(message.getBytes());
    return Base64.getEncoder().encodeToString(encryptedBytes); }
public static String decryptWithAES(String encryptedMessage, SecretKeySpec key)
throws Exception
{
     Cipher aesCipher = Cipher.getInstance("AES");
     aesCipher.init(Cipher.DECRYPT_MODE, key);
     byte[] decodedBytes = Base64.getDecoder().decode(encryptedMessage);
     byte[] decryptedBytes = aesCipher.doFinal(decodedBytes);
     return new String(decryptedBytes); }
        return new String(decryptedBytes); }
public static int calculateNumParityRows(int numRows)
```

```
{
      if (numRows <= 4)
      {
  return 2;
      }
      else if (numRows <= 8)
{
     return 4;
}
 else if (numRows <= 12)
{
     return 6;
}
else if (numRows <= 16)
{
         return 8; // Assuming pattern continues with increments of 4 }}
public static int[] genDivideMatrix(int n)
{
         switch (n)
         {
     case 4: return new int[]{1, 1, 1, 1, 1, 1};
     case 5: return new int[]{1, 1, 1, 2, 2, 2};
     case 6: return new int[]{1, 1, 2, 2, 2, 2};
     case 7: return new int[]{1, 2, 2, 2, 2, 2};
}
return null;
}}
```

5. TEST CASES

Experiment 1: Cost Comparison

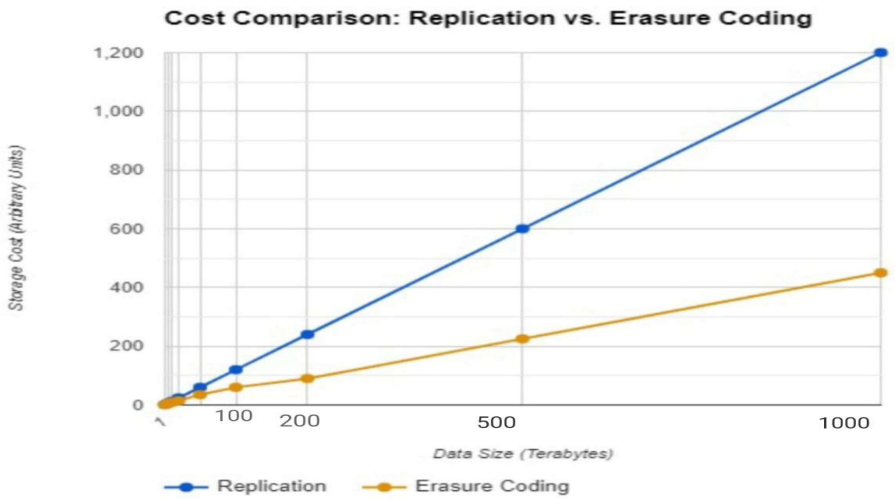

Fig 5.1 Cost Comparison

Experiment 2: Comparison of single vs Double Encryption

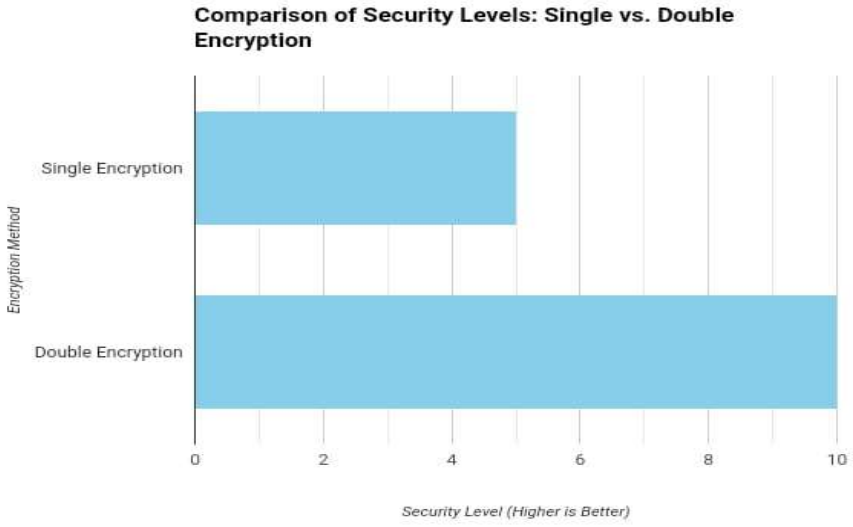

Fig 5.2 Comparison of single vs Double Encryption

6.SCREENSHOTS

6.1 Successful Data Retrieval

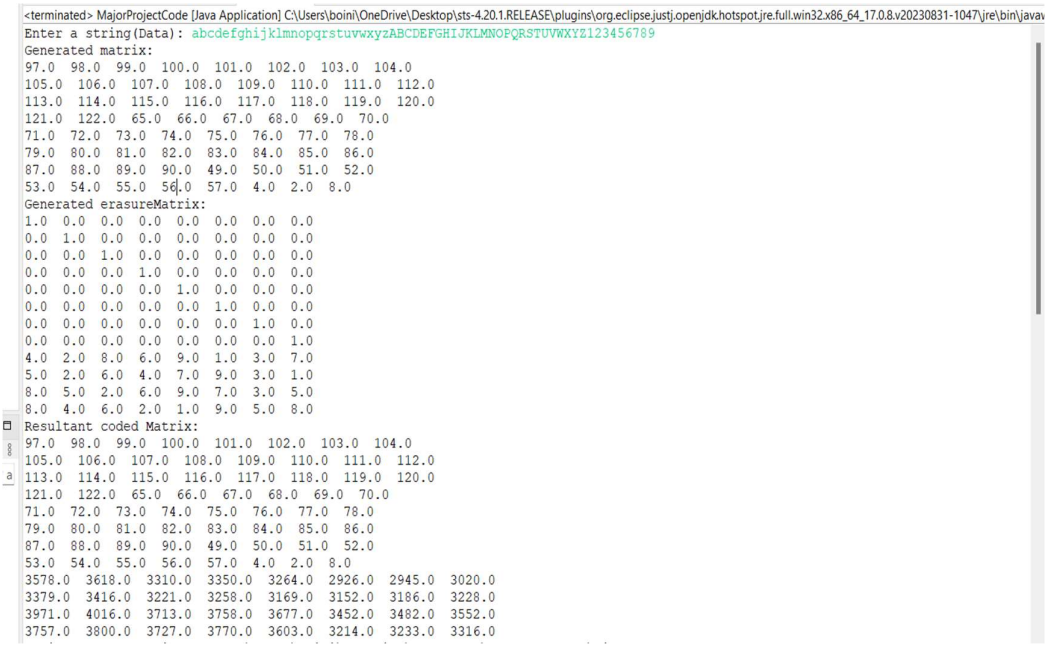

Fig 6.1 Successful Data retrieval

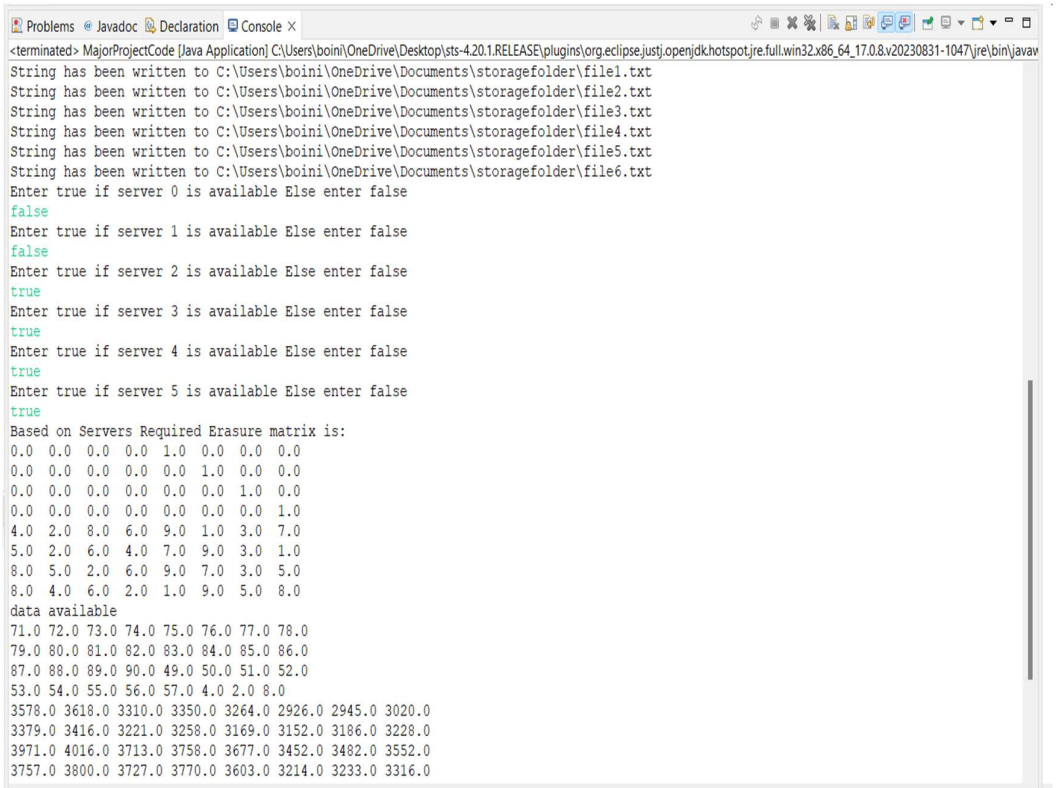

Fig 6.2 Successful Data retrieval Continuation

```
Based on Servers Required Erasure matrix is:
0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
4.0  2.0  8.0  6.0  9.0  1.0  3.0  7.0
5.0  2.0  6.0  4.0  7.0  9.0  3.0  1.0
8.0  5.0  2.0  6.0  9.0  7.0  3.0  5.0
8.0  4.0  6.0  2.0  1.0  9.0  5.0  8.0
data available
71.0 72.0 73.0 74.0 75.0 76.0 77.0 78.0
79.0 80.0 81.0 82.0 83.0 84.0 85.0 86.0
87.0 88.0 89.0 90.0 49.0 50.0 51.0 52.0
53.0 54.0 55.0 56.0 57.0 4.0 2.0 8.0
3578.0 3618.0 3310.0 3350.0 3264.0 2926.0 2945.0 3020.0
3379.0 3416.0 3221.0 3258.0 3169.0 3152.0 3186.0 3228.0
3971.0 4016.0 3713.0 3758.0 3677.0 3452.0 3482.0 3552.0
3757.0 3800.0 3727.0 3770.0 3603.0 3214.0 3233.0 3316.0
Matrix Multiplication Result or actual matrix after data re-construction
97.0 98.0 99.0 100.0 101.0 102.0 103.0 104.0
105.0 106.0 107.0 108.0 109.0 110.0 111.0 112.0
113.0 114.0 115.0 116.0 117.0 118.0 119.0 120.0
121.0 122.0 65.0 66.0 67.0 68.0 69.0 70.0
71.0 72.0 73.0 74.0 75.0 76.0 77.0 78.0
79.0 80.0 81.0 82.0 83.0 84.0 85.0 86.0
87.0 88.0 89.0 90.0 49.0 50.0 51.0 52.0
53.0 54.0 55.0 56.0 57.0 4.0 2.0 8.0
Data Stored is:abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789
```

Fig 6.3 Successful Data retrieval

6.2 Unsuccessful Data Retrieval

```
<terminated> MajorProjectCode [Java Application] C:\Users\boini\OneDrive\Desktop\sts-4.20.1.RELEASE\plugins\org.eclipse.justj.openjdk.hot
Enter a string(Data): abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789
Generated matrix:
97.0  98.0  99.0  100.0  101.0  102.0  103.0  104.0
105.0  106.0  107.0  108.0  109.0  110.0  111.0  112.0
113.0  114.0  115.0  116.0  117.0  118.0  119.0  120.0
121.0  122.0  65.0  66.0  67.0  68.0  69.0  70.0
71.0  72.0  73.0  74.0  75.0  76.0  77.0  78.0
79.0  80.0  81.0  82.0  83.0  84.0  85.0  86.0
87.0  88.0  89.0  90.0  49.0  50.0  51.0  52.0
53.0  54.0  55.0  56.0  57.0  4.0  2.0  8.0
Generated erasureMatrix:
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
4.0  2.0  8.0  6.0  9.0  1.0  3.0  7.0
5.0  2.0  6.0  4.0  7.0  9.0  3.0  1.0
8.0  5.0  2.0  6.0  9.0  7.0  3.0  5.0
8.0  4.0  6.0  2.0  1.0  9.0  5.0  8.0
Resultant coded Matrix:
97.0  98.0  99.0  100.0  101.0  102.0  103.0  104.0
105.0  106.0  107.0  108.0  109.0  110.0  111.0  112.0
113.0  114.0  115.0  116.0  117.0  118.0  119.0  120.0
121.0  122.0  65.0  66.0  67.0  68.0  69.0  70.0
71.0  72.0  73.0  74.0  75.0  76.0  77.0  78.0
79.0  80.0  81.0  82.0  83.0  84.0  85.0  86.0
87.0  88.0  89.0  90.0  49.0  50.0  51.0  52.0
53.0  54.0  55.0  56.0  57.0  4.0  2.0  8.0
3578.0  3618.0  3310.0  3350.0  3264.0  2926.0  2945.0  3020.0
3379.0  3416.0  3221.0  3258.0  3169.0  3152.0  3186.0  3228.0
3971.0  4016.0  3713.0  3758.0  3677.0  3452.0  3482.0  3552.0
3757.0  3800.0  3727.0  3770.0  3603.0  3214.0  3233.0  3316.0
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file1.txt
```

Fig 6.4 Unsuccessful Data Retrieval

```
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file1.txt
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file2.txt
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file3.txt
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file4.txt
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file5.txt
String has been written to C:\Users\boini\OneDrive\Documents\storagefolder\file6.txt
Enter true if server 0 is available Else enter false
false
Enter true if server 1 is available Else enter false
false
Enter true if server 2 is available Else enter false
false
Enter true if server 3 is available Else enter false
true
Enter true if server 4 is available Else enter false
true
Enter true if server 5 is available Else enter false
true
Sorry, number of servers failed is greater than parity servers used. We cannot recover data.
```

Fig 6.5 Unsuccessful Data Retrieval

7. CONCLUSION

The project for Cost Effective Data Placement in Edge Storage Systems With Erasure Code represents a significant contribution to the advancement of edge computing infrastructures. Through comprehensive research, algorithm development, and empirical validation, the project has successfully addressed the pressing need for efficient data placement strategies in edge environments. By leveraging erasure coding techniques, the project has demonstrated the potential to significantly reduce storage costs while ensuring data reliability, availability, and low latency at the network edge. The methodologies and algorithms developed as part of this project offer practical solutions for optimizing data placement in edge storage systems, taking into account factors such as limited storage resources, intermittent connectivity, and dynamically changing network conditions. Through rigorous simulation and experimentation, the effectiveness of the proposed solutions has been empirically validated, providing organizations with confidence in their ability to deploy cost-effective edge computing applications. Furthermore, the insights gained from this project can inform future research and development efforts in the field of edge computing, driving innovation and advancements in data management and optimization strategies. By delivering practical guidelines and solutions, the project empowers organizations to harness the potential of edge computing while effectively managing costs and ensuring high levels of data integrity and accessibility. Overall, the project represents a significant step towards realizing the full potential of edge computing in various industries and applications.

8. FUTURE ENHANCMENT

The project for Cost Effective Data Placement in Edge Storage Systems With Erasure Code could focus on several key areas to further improve the efficiency and effectiveness of data placement in edge environments. Firstly, incorporating machine learning and AI-driven algorithms could enable more adaptive and predictive data placement strategies, taking into account dynamic changes in workload patterns and network conditions. This would enhance the agility and responsiveness of edge storage systems, optimizing resource utilization and minimizing latency. Secondly, exploring advanced erasure coding techniques and hybrid approaches could offer opportunities to achieve even greater storage efficiency and fault tolerance in edge environments. By integrating erasure coding with other redundancy mechanisms such as replication or RAID, it may be possible to strike a finer balance between storage overhead and data reliability.

9. BIBLIOGRAPHY

[1] M. Noormohammadpour and C. S. Raghavendra, “Datacenter traffic control: Understanding techniques and tradeoffs,” IEEE Commun. Surv. Tut., vol. 20, no. 2, pp. 1492–1525, Apr.–Jun. 2018.

[2] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobileedge computing–a key technology towards 5G,” ETSI White Paper, vol. 11, no. 11, pp. 1–16, 2015.

[3] A. Ceselli, M. Premoli, and S. Secci, “Mobile edge cloud network design optimization,” IEEE/ACM Trans. Netw., vol. 25, no. 3, pp. 1818–1831, Jun. 2017.

[4] H. Badri, T. Bahreini, D. Grosu, and K. Yang, “Energy-aware application placement in mobile edge computing: A stochastic optimization approach,” IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 4, pp. 909–922, Apr. 2020.

[5] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, “A provably efficient online collaborative caching algorithm for multicell-coordinated systems,” IEEE Trans. Mobile Comput., vol. 15, no. 8, pp. 1863–1876, Aug. 2016.

[6] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin,“Cost-effective app data distribution in edge computing,” IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 1, pp. 31–44, Jan. 2021.

[7] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, “Dynamic service placement for mobile micro-clouds with predicted future costs,” IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[8] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, “Learningaided computation offloading for trusted collaborative mobile edge computing,” IEEE Trans. Mobile Comput., vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[9] R. Luo, H. Jin, Q. He, S. Wu, Z. Zeng, and X. Xia, “Graph-based data deduplication in mobile edge computing environment,” in Proc. Int. Conf. Serv.-Oriented Comput., 2021, pp. 499–515.

[10] F. Zhang et al., “Online learning offloading framework for heterogeneous mobile edge computing system,” J. Parallel Distrib. Comput., vol. 128, pp. 167-183, 2019.