Data Management and System Optimization Techniques: A Comparative Study

Kishan Kumar Zalavadia

Masters in computer science

KishanKumarZalavadia@my.unt.edu

**ABSTRACT**

This paper discusses and compares the data management and system optimization methods. It starts by discussing the challenge of managing many small files in Ceph storage systems. There are solutions proposed such as file grouping and merging. Further, the paper explores the advantages of breaking down large applications into smaller services known as microservices, which also focus on choosing between SQL and NoSQL databases based on different factors such as consistency, speed, and scalability. The next part focuses on efficiently handling and managing unstructured short video data which uses SQL database for storing the data. It also highlights the importance of safety in cloud computing. The paper describes the methods for security for cloud-like pattern fragmentation. Moreover, it discusses the fault-tolerant distributed systems and replicated data which depend on the consistency, availability, and latency of large-scale systems. It also describes the method for querying complex data through semi-joins and parallel processing which mainly handles large amounts of datasets. Finally, the paper compares all the methods and gives a brief about the strengths and weaknesses of all the methods.

**Keywords**

**Data management, Microservices, SQL databases, NoSQL databases, Cloud computing, Fault tolerance, Distributed systems, System optimization, Data safety, and Query optimization.**

# INTRODUCTION

Nowadays, the world is completely dependent on the internet. With the massive growth in big data and cloud computing internet-based applications and distributed computer systems have become a critical part of business areas. This system must handle large amounts of data ensuring the high availability of data, efficiency in handling the data, and scalability of the system. There are many traditional methods such as monolithic architecture, synchronous fault-tolerance mechanisms, relational databases, and many more that do not meet the requirements. Monolithic applications do not handle flexibility and scalability, synchronous fault-tolerance methods rely on low latency networks for synchronous communication which are not suited for globally distributed systems due to high network failure risk, and relational databases face difficulties in distributed data management.

To overcome these issues, there are several methods and data storage and processing techniques proposed in this paper. The microservice architecture has become popular because of its ability to independently develop, deploy, and maintain applications. Due to this, it ensures flexibility and scalability are increased for each service separately without disturbing other services. However, this raises a new problem of maintaining data consistency across distributed microservices and trusting the data from other services.

Ceph which is a distributed NoSQL is widely used because of its horizontal scalability, and high availability and it can handle large volumes of unstructured data. But these systems are less efficient in storing and managing small files which leads to problems like data holes, extra metadata, etc. This paper also proposes a few techniques like small file grouping, merging small files, sequential storage and shared metadata, and optimized deletion to overcome the issue.

While storing the data in the cloud, we need to blindly trust the cloud provider to keep our data safe. To overcome this hesitation the paper discusses an alternative approach to encryption which is fragmentation-based data storing. In this method, we divide the data into fragments, shuffle them, and store them in a distributed NoSQL database. The advantage of this is high data security and parallel processing.

To improve the performance (nothing but speed) in distributed databases where the resources are limited such as in IOT and mobile computing the queries must be optimized using techniques such as semi-join and parallel decision tree models which are further discussed in detail in this paper.

To keep the distributed system up and running, replication and system redundancy play a vital role in ensuring fault tolerance. To maintain consistency across the replicate data, there could be significant latency overheads and high energy consumption. Hence it is important to consider proper trade between consistency and availability which can impact response time and overall dependability which will be analyzed in the paper.

This paper compares the proposed solutions and approaches that address scalability, optimal storage methods for small files, choosing databases for microservices, query optimization performance, and the security aspects of distributed systems.

# DESCRIPTIONS OF COMPARED TECHNIQUES

## An optimized storage method for small files in Ceph system

In today’s world the data is increasing rapidly, and to manage the data efficiently and reliably distributed storage systems like Ceph are used which is based on the CRUSH algorithm. Most of the data is in the form of small files due to which the traditional storage system has issues like the formation of data holes and high redundancy of metadata which leads to wastage of storage, apart from that the read-write speed for small files is also reduced due to the use of mechanical disks. There are certain limitations to the existing solutions. There can be new issues introduced if we redesign the whole storage system which is also not an effective solution.

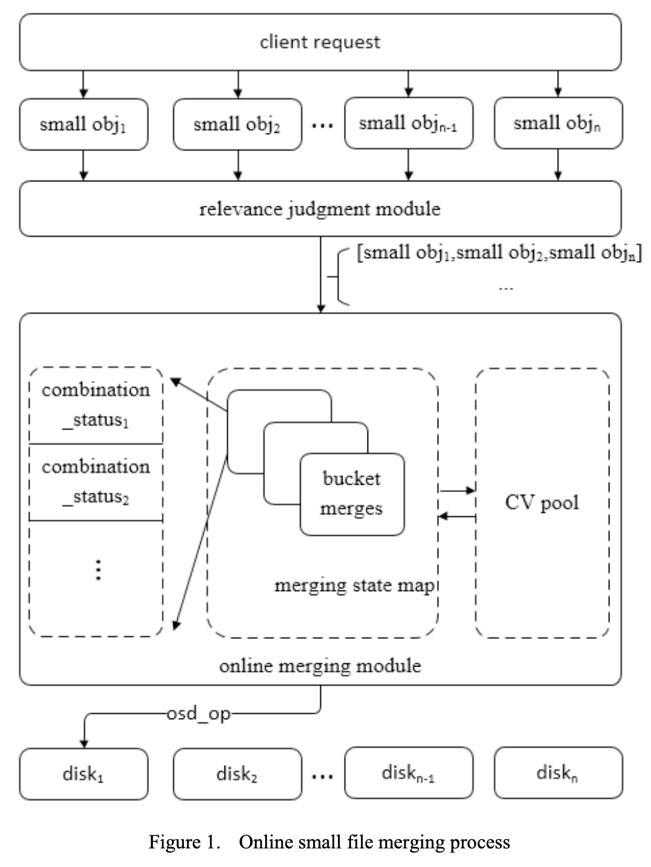
To solve the above problem we can directly merge data files during IO operation avoiding the impact on the user business. These merged files are sequentially requested from the storage hence optimizing the IO path and reducing the network redundancy. To merge the files appropriately the updated model based on the CRUSH algorithm is designed which improves the IO performance. To avoid data holes while deleting the small files, a garbage collection method is designed.

### Online small file merging storage system

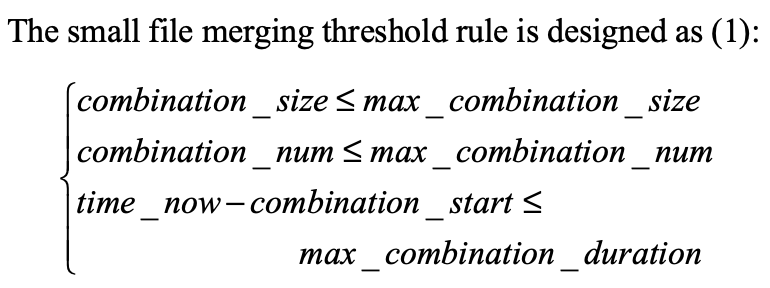
It consists of two main parts: the relevance judgment module which merges small files based on the CRUSH algorithm and the online merging model which merges small file data through the CV.

#### Relevance judgment module

The main purpose of this module is to combine smaller files and make them a large file to reduce the overhead and improve the performance.



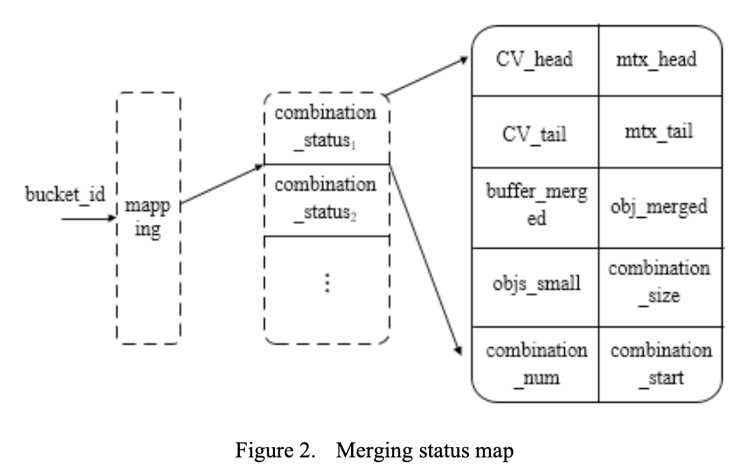
To do so, first, the small files are grouped based on their upload times which is known as 'time locality'. Then each group is further divided into files by the CRUSH algorithm which is based on the disk they will be stored. Now the system checks within a group that combining them should follow certain rules based on the file size, duration, and count should not exceed the threshold.



If the rules are met then the system combines the small files to make a single large file.

#### Merging status map

Once the files are merged the merging status keeps track of the merging tasks for every group. The key to merging the status map is bucked id and the value is the list of combination statuses. The file with the earliest upload time is known as the merge header file, and the corresponding upload thread is the merge header thread. All other files are known as merge tail files and the thread is referred to as merge tail thread. To synchronize the merge head thread we use CV\_head which is the conditional variable and mtx\_head which is the mutex of the merge head thread. Similarly to synchronize the merge tail thread we use CV\_tail and mtx\_tail.



Following is the process of merging small files. First, the merge head thread updates the merge status map, waits for the notification, and sends all the merged files to the storage disk. Finally, the response of the merge head thread is sent to the client notifying CV\_head simultaneously. Once the notification is received, the merge tail thread returns the response to the client.

#### CV Pool

In small file merging I/O performance and time precision can be impacted by frequent allocation and release of synchronization. To improve this the CV pool method is used using a read-write lock and a list of CV groups. The third gets from the pool which reduces the memory operations. Once the merging is done successfully the primitives are returned to the CV pool. The advantage of this is it optimizes synchronization and minimizes performance overhead.

### Garbage data cleanup

When the small files are deleted after merging, they do not immediately get deleted. It creates a GC log and the GC thread completes the deletion process as shown in Figure 3.

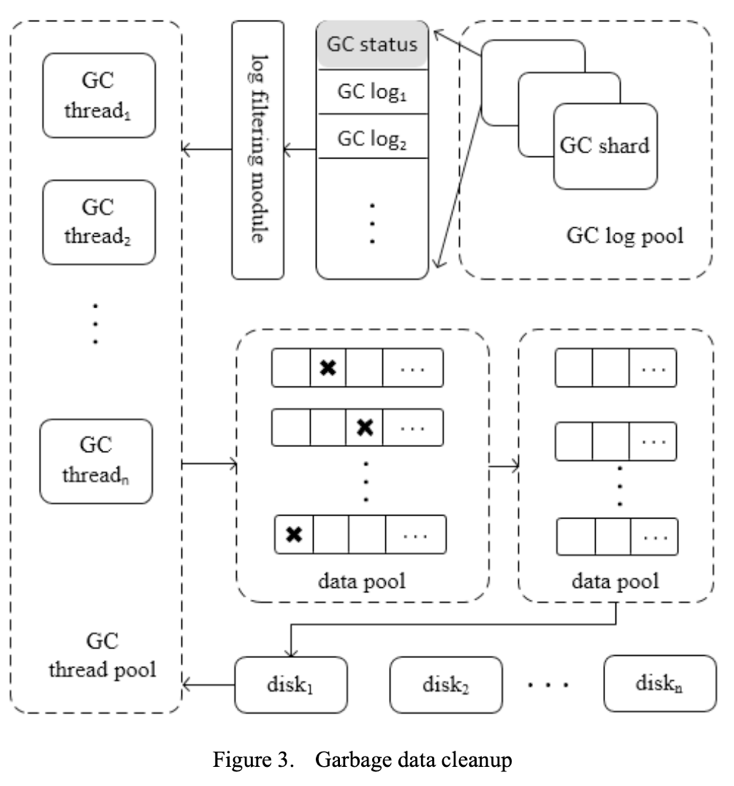
#### GC log

The information like offset, status flags length of the garbage data is in the GC log which is stored in the GC object which is divided into 64 shards for efficient storage. The GC threads periodically clean the garbage data. To identify the log for cleanup, they read the GC status.

#### Garbage Data Cleanup

The process of the GC thread is as follows.

First, select a GC thread shard randomly to read GC status. If the last cleaning task occurred before 24 hours, all the GC logs are listed after the marker on the GC object shard, and GC\_initial is set on the above GC log and only these logs are cleared. If the last cleaning task was within 24 hours, then skip the above process. Now, for the index marker, if the status is GC\_processing, then skip the cleaning and randomly select another GC object shard for cleaning. The thread handles task logs marked for cleaning, which deletes the corresponding data from the disk and updates it in the logs. If there is some problem and it is unsuccessful, then it is done again in the next cleanup cycle, which ensures efficient disk cleanup.



### Experimental Results

To experiment with the setup of a distributed storage cluster was built using CEPH of version 16.2.11which contains 4 storage nodes and 3 management nodes.

Each storage node had 10 OSD (Object Storage Daemon) each with a capacity of 3.7TB. The experiment found the IOPS performance, data hole rate, and metadata amount. The IOPS performance was increased by 62% when compared with the traditional CEPH system. The data hole rate decreased 97.3% which is a significant improvement, which means 97.3% storage wastage was reduced which was caused by file sizes less than 5KB. And finally, the number of metadata is reduced by 57% when compared with the traditional CEPH schema.

## SQL vs NoSQL Databases for the Microservices: A Comparative Survey

In software design one of the most important aspects is flexibility. To maintain the system flexibility the microservice architecture is used because it can provide designing and developing software components separately in a distributed manner. One big software can be divided into smaller independent services that can be updated or replaced separately. Even though the microservice provides various advantages, the service may not fully trust the data, which means data consistency could be a problem. It is important to find a solution to improve the response time especially when handling large datasets.

The idea here is to improve the separation of concerns in the software architecture. Separation of concerns means the capability of breaking down the system into modules that exhibit logical coherence and loose coupling which will allow the distributed and scalable applications.

### Microservice and Databases

Storing events, experiences, and knowledge has been done for a very long time. In the initial days of human history, information was stored in ancient sculptures in caves, now it is advanced to cloud storage. The evolution of technology has played a vital role in advancing storage techniques and technologies, which has resulted in the rapid development of solutions for storing information.

Response time, throughput, error rate, and resource utilization are used to measure the performance of the microservice.

#### Types of Databases Used in Microservices

##### Relational Databases

A relational database has been commonly used on microservices for a very long time due to its consistency, reliability, and durability. The major drawback of using this database is scaling and performance.

RDBMS (Relational Database Management Systems) is designed to organize data in the tables using schema (which is a fixed structure). It also allows only single datatypes which allow the user to have only minor changes in the data which is not very helpful.

The following are the limitations of RDBMS:

- Scalability: SQL does not provide built-in support for distributed data processing and storage. So to handle a high volume of data we need more computational power which can be obtained by powerful servers that leads to an increase in hardware components making it expensive.

- Flexibility and complexity: Relational databases are less flexible because they require a predefined structure to store the data which is called a schema.

#### 2.2.1.1.2 NoSQL Databases

The advantage of a NoSQL database is that it can handle various data types which include structured, semi-structured, and unstructured data. It overcomes the limitations of relational databases. It can handle large amounts of data and can be horizontally scaled.

The following are the limitations of NoSQL databases.

- Data consistency: It is less consistent when compared to a relational database.

- Schema less design: Due to schema less design it can be difficult to enforce data integrity and it can also be challenging to query the data effectively.

- Security: It is more vulnerable to attack when compared with relational databases due to its open nature.

- Backup and recovery: It isn't easy to maintain a backup and recover the data.

#### 2.2.1.1.3 NewSQL Databases

It is a new type of database that offers the scalability of a NoSQL database and the consistency and durability of a relational database.

The following are the limitations of the NewSQL database.

- The applications that handle large amounts of data are not suitable for this purpose.

- It is not compatible with traditional SQL tools.

### Benefits of Using Microservices

- When a company decides to migrate the system, it could take a large amount of time and high cost. By using microservices the company can avoid this process.

- The system release can be done independently.

- Because the system is divided into small services the results of those services are not dependent on each other, which could be beneficial which developing, understanding, and testing the system.

- Every team can make their own decision on developing and deploying a new service.

- All the services are independent of the programming language and platform they are built on.

- The microservices also support methodologies like DevOps, Agile, and CICD(Continuous Integration Continuous Deployment). Each service can be deployed separately.

### Challenges faced while using Databases in Microservice

Unlike monolithic architecture where all the data is stored in a single database maintaining data consistency is a bit difficult task. In microservice, each service has its data storage, and it must be consistent. The challenging part is to maintain data consistency among all the services as a whole which can be even more difficult when the services use different databases, apart from that when developing and deploying microservices, the following parameters also need to be considered.

#### Read and Write Performance

The read performance is measured by the number of operations per second or the combination of query execution speed and result retrieval speed. The write performance is calculated by the number of write operations per second.

#### Efficiency

High efficiency means the database should respond quickly. To do that the low-latency databases must be utilized and the microservices are deployed close to the database proximity. Generally, a response time of less than 1 ms is low latency, and greater than 10 ms is high latency.

It is also important to make the database scalable. The microservice must be developed and deployed efficiently and rapidly.

#### Data Sharing

Data sharing means each service must have its data, sometimes one service needs data from another service, and in that case, the services must share the data. The data precision should also be maintained and the services must remain private and share data via a dedicated API. The concept of data replication can also be used to make it quicker and more efficient.

## Video Website Management System Based on Database SQL

Looking at the rapid development of the internet, especially short video content, an application has been developed that handles and manages short video content which also capitalizes on artificial intelligence and the 5G era.

The aim is to develop an application to manage short video content efficiently storing data in SQL (Structured Query Language) database and maintaining system security. After gathering and analyzing the functional and non-functional requirements of the user, the functional requirements were divided into two main parts: client-side and server-side.

### Related Technology

The server-side programming is done using Java which is a popular language for web development. It uses Spring Boot framework which is based on Java Web.

MySQL, the relational database is used to store the data in this application because of its small size and open-source nature. It is also efficient, and easy to use and maintain.

Other technologies used are Sqagger2 framework, Thymeleaf, Bootstrap framework, CSS, JavaScript, and jQuery.

### Requirement Analysis

The video management application has 3 types of users.

- Guests: They can sign in and can become the individual user.

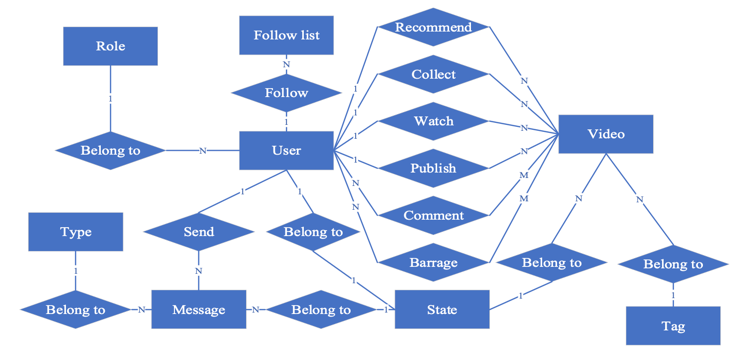
- Individual user: They can log in to their account and manage their information, watch and interact with videos, and publish their videos.

- System Administrators: They manage the whole system’s resources and this is done on the client side. The system administrators manage the information, maintain the data security, and analyze the statistics of the applications.

The aim is to make the system easy to use, it should handle lots of data (video content) efficiently and quickly respond to changes.

### System Design

Below is the ER Diagram for the video management application.



The database contains 12 tables in which the data is stored.

Coming to the system functional design, the spring boot framework consists of 5 layers which are view, controller, service, mapping, and model layers. The way this works the user interacts with the view layer to update the data. The data is then passed to the controller which passes the data to its proper service. The service maps the data to the repository (also known as the mapping layer).

The repository maintains and manages data with the help of a model which is finally stored in the database. Similarly, if the data needs to be read, the data from the database is fetched in the form of the model by the repository. This repository passes the data to the service that sends the data to the controller, and it displays the data in the view layer so that the user can see the data in the UI.

The UI contains different pages like login, registration page, home page, video playback page, and video publishing page.

The application also includes a backend management system that handles and manages the data and helps to visualize the tasks. The job of this system is to manage users, roles, videos, and tags. It performs operations like changing the state of a video from pending approval to listing after it has been reviewed. The visualization helps to get the number of views, users, and all other numeric related information and it displays in the form of a pie chart and bar graph.

### Performance Evaluation

First, the functional evaluation is performed where all the functions are checked to ensure that they are working as expected. The test is performed all the functionalities are working efficiently and the visualization also displays all the information accurately.

To perform the test, 500 videos with different tags were uploaded to the application. Apart from functional testing, the CPU occupancy, time consumption, and response time were also monitored. The first run of the system took less than 100ms to respond and less than. 10% CPU was occupied which means the application performed very well in terms of response time and CPU utilization.

## Enhancing Data Security in the Cloud using Random Pattern Fragmentation and a Distributed NoSQL Database

This paper discusses a secure data storage method in the cloud. Cloud computing has become popular because it focuses on development rather than managing and maintaining hardware and software which results in effective utilization of time. But at the same time, cloud providers can also get attacked by outsiders or insiders which can be a threat to data security.

So to protect the data from attackers encryption is used which can also have some drawbacks if the encryption is compromised then the data is exposed. Encryption can sometimes be complicated and can add overheads when dealing with large volumes of data.

So to solve this issue the paper proposes a method using a fragmentation algorithm combined with a distributed NoSQL database to ensure cloud data security. The idea is to divide the data into chunks and scramble them across different nodes in the database. This method is faster than encryption and it can be advantageous in applications that have limited resources by allowing simultaneous processing across different nodes.

This method is also suitable in places where speed is an important factor. The benefit of this method is that even if the attackers get access to the databases, they will not be able to reconstruct the data.

### Related Works

#### Data Anonymization

Anonymization of data is a technique used to keep the cloud data safe. Some anonymization techniques are as follows.

- K-anonymity: It generates a large set of classes of records. An attacker can identify individuals in the records who are in the same class and have sensitive values.

- L-Diversity: This technique overcomes the drawback of K-anonymity, but it is difficult to implement and is insufficient in preventing attribute disclosure.

t-closeness: It overcomes the drawback of L-Diversity. However, this method limits the amount of information which is useful that can be extracted.

The paper compares different MapReduce-based anonymization techniques. Additionally, the semantic labeling technique is used for locations mappable to predefined semantic vocabulary. However, it cannot add or modify the categories dynamically.

#### Encryption

Encryption is the most common method used to ensure the security of data. There can be many ways to protect data using encryption such as combining DES and AES using RSA, combining symmetric and asymmetric encryption along with hashing and salting techniques to increase the level of protection in cloud data. But if we use such a high level of encryption techniques, it can affect the efficiency of the system. One more drawback of these techniques is they cannot be used on multi-cloud systems.

A fully homomorphic encryption technique was proposed to allow the user to compute their data on the public cloud, but this method increased overhead and needed a large number of resources.

Even if encryption is widely used it has the following drawbacks:

- In symmetric encryption, it is required to share the secret key which can be compromised. And the data or the schema is also rendered insufficiently.

- While performing encryption the computing can be prone to overhead, which can be a bad solution for devices with limited resources like mobile phones.

- Encryption stops queries running on encrypted data, so to run the query it needs to decrypt the data and then run which is not a fast process. So applications where speed is important like read-time system encryption can cause a time delay.

Due to these limitations, encryption can reduce performance which makes it less suitable for some cloud computing scenarios. To overcome this issue the paper discusses a method that can be used.

#### Data Fragmentation

Data fragmentation ensures data security at much lower costs which also allows multiple fragments to be accessed simultaneously with the help of parallel computing. Data fragmentation is mainly adapted in relational databases and multi-cloud architectures. The data fragmentation can be done using bitwise and structure-wise. To analyze the performance of data fragmentation including encryption, a combination of random pattern fragmentation with AES is encrypted. The result indicated the trade-off between performance and security. Some of the researchers used fragmentation with the permutation of a particular file type which was JPEG. Some combined fragmentation with NoSQL databases which caused a failure when hosting on a single instance.

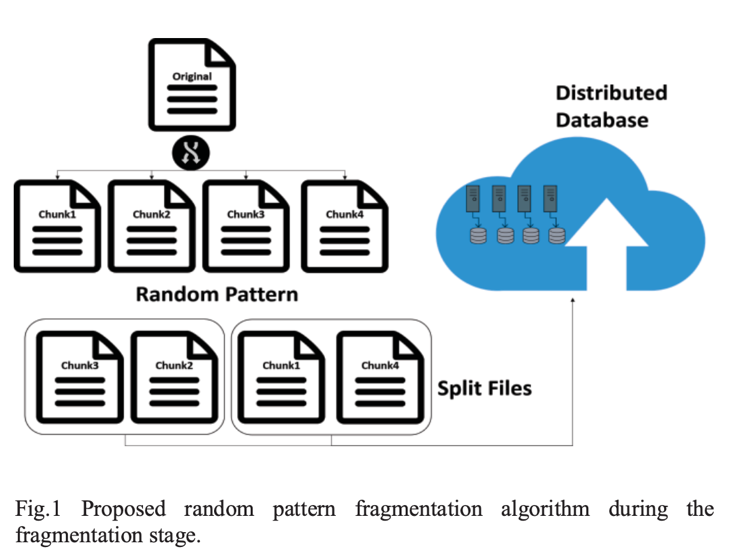
The idea here is to divide the data into chunks and store them in different nodes to add an extra data security level.

### Methodology

As discussed previously the aim is to improve data security by dividing the data into chunks/fragments and storing them in a database which is distributed over different nodes. The main advantage of this method is that even if a node gets compromised, the attacker cannot get complete data. Unlikely even if all the nodes get compromised the attacker would not be able to reconstruct the data with either the pattern key or the brute force method which would take a very large amount of time.

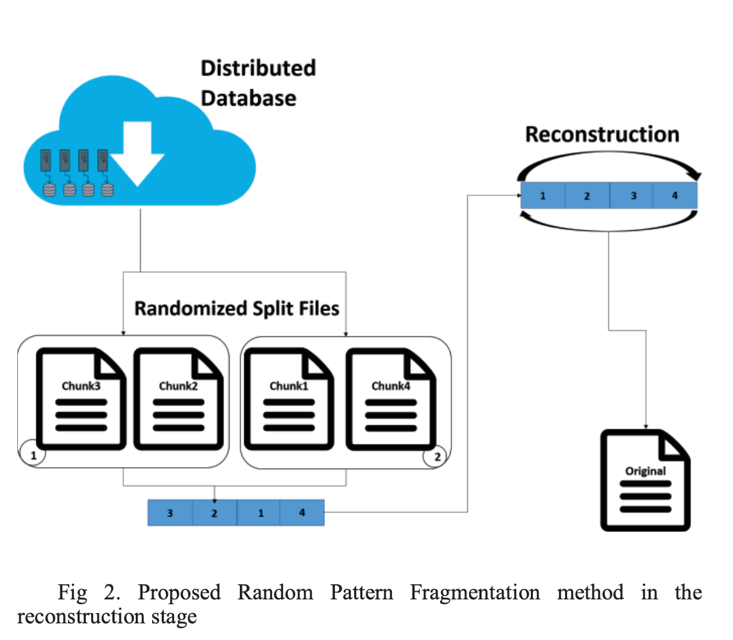
#### Random Pattern Fragmentation (RPF)

The random pattern fragmentation algorithm first divides the original file into N chunks which are determined by the user. Then the chunks are scrambled in a random order. The scrambled chunks are stored in a special file containing metadata which is then serialized in an array. Finally, these spit files are stored in the database where each file is stored in a row in the table. This method does not track the header and footers of the chunks to avoid unwanted performance overheads. This method depends on the metadata in the spilled file and the order of the pattern stored in the client machine to identify the correct order of the chunks. All the communication occurs via VPN (virtual private network) which encrypts all the data in the traffic.



The above figure indicates a pattern fragmentation algorithm during the fragmentation stage. The original file is broken down into random patterns and then split files and then it is stored in a distributed database.

During the reconstruction stage, the database is queried on the metadata which is on the split file. The split files are downloaded in the client machine and it is reconstructed based on the metadata present in the files. There is a directory that maps unique IDs to its raw data. After the chinks are arranged in the correct order, the chinks are converted to a byte array and de-serialized and the original file is stored in the client machine. Like fragmentation, all the communication between the client machine and the database is done through a VPN.



The above diagram represents a pattern fragmentation method in the reconstruction stage. The random split files are obtained from the distributed database and these files are reconstructed to get the original file.

#### Cassandra Distributed Database

Cassandra distributed database is a NoSQL open-source database that is built to handle large amounts of data. The data can be across servers which is managed with high availability and no single point failure. This database consists of different nodes that communicate using peer-to-peer communication without a master node. There is a component in Cassandra databases known as a partitioner that is responsible for the internal partitioning of data and a hashing mechanism assigns the row to a particular cluster based on the primary key. It is built to handle large volumes of data. In this approach of data fragmentation, the split files are stored and the same number of tables are stored and the user selects the number of splits. After fragmentation, the threads concurrently insert the metadata and split file chunks in their respective tables. Similarly while downloading the file also the threads are created which allows concurrency download.

### Experiment and results

To test the performance of the model an experiment has been conducted that contains a dataset of 1000KB files in the form of .bmp, jpeg, .pdf, and .docx. The chink size is 1000 bytes and the number of splits per file is 2. There are 3 nodes considered in the Cassandra cluster on Azure virtual machines. The client is connected via VPN to ensure secure communication between the client machine and the database.

Based on the result we can tell that the Cassandra method has a latency of around 0.56s which is better. The performance also has increased by parallel processing of split files on the database and client machine. The sequential block-wise operations of encryption are avoided resulting in an improvement of performance. Also, Cassandra’s distribution and replicated architecture avoids a single point of failure.

## Analysis of Trade-offs in Fault-tolerant Distributed Computing and Replicated Database

Most business application uses distributed computer systems that can be scaled by the internet. These systems are also vulnerable to system failure which can affect the businesses. The large-scale distributed systems that are running over the internet can be subjected to component failure, network disconnections, packet loss, and other accidents which can be critical and may cause harm to the data.

To ensure the high availability of the data for many applications we can use methods like data replication and system redundancy. To make sure that the data is consistent over the replicas the traditional fault tolerance system uses synchronous communication over a short amount of time. This could be efficient for those distributed systems whose components are located in the same data center or within the same local area network. It could be difficult for the systems that are distributed globally to maintain proper and consistent communication due to the network delay.

When deploying replication and fault tolerance architecture over the internet or cloud-based system additional time delay must be considered which may cause the energy overhead.

To make the data highly available, we need a large amount of data replicated which takes more energy from the system. The amount of energy required to maintain data consistency is high, which makes it important to understand the trade-off between energy consumption, durability, and system latency, along with availability, consistency, and partition tolerance.

### CAP and PACELC Theorems and Their Implications

The CAP theorem explains the quantitative trade-off between system consistency, availability, and partition tolerance. According to this theorem, only two of these are preserved a time in a distributed replicated system.

It is not possible to have both high consistency and high availability together in an internet-scale distributed system. To make system data highly available, we need to make more data replicated hence the system overhead increases resulting in decreased consistency. And to maintain high consistency in the system the availability gets compromised.

The AP systems give high priority to availability and partition tolerance by sacrificing consistency. The best example of AP systems is NoSQL databases. The CP Systems prioritizes consistency and partition tolerance compromising availability. Examples of CP systems are MongoDB. The choice of which features to have and which to compromise depends on the application’s needs.

PACELC theorem is an extension and an improved version of the CAP theorem. According to this algorithm, we need to choose between availability and consistency in the case of the partitioning algorithm, and in the absence of partitions of the replicated systems we still need to trade-off between latency and consistency.

The PC/EC prioritizes consistency over latency and availability whereas PA/EA systems prioritize availability and latency over consistency.

### Trade-offs in Fault-Tolerant Distributed Computing and Replicated Databases.

The main purpose of this study is to understand the idea of viewing CAP/PACELC properties as more continuous than binary.

Availability is measured from 0% to 100% and the latency (noting but response time) is practically measured from minimal response time which is greater than zero to application timeout time. Consistency is scales from weak to strong.

The below sub-sections describe the trade-offs between core QoS properties of distributed computing systems in different consistency models.

#### Consistency Models and Levels

In a distributed data storage system, the consistency of a model is defined as how to system performs when multiple users try to read and write data at the same time. There are two types of consistency models.

- Data-centric model: This model makes sure that all the users of the same type view the same data state.

- Client-centric model: This model helps to view variations in data seen by different users.

The two models differ in how they balance consistency and performance.

In a distributed system, achieving a very high consistency is difficult, which means all the users see the same data. So that is why many systems use relaxed or eventual consistency models, which prioritize performance.

There are different ways by which consistency can be achieved. Different system uses different methods to achieve consistency. For example, MongoDB uses a consistency model that relies on adjusting parameters for a write operation. Apache uses a consistency model which controls how many replicas are involved in read and write operations. Each approach has its advantages and drawbacks which makes it difficult to combine all the approaches.

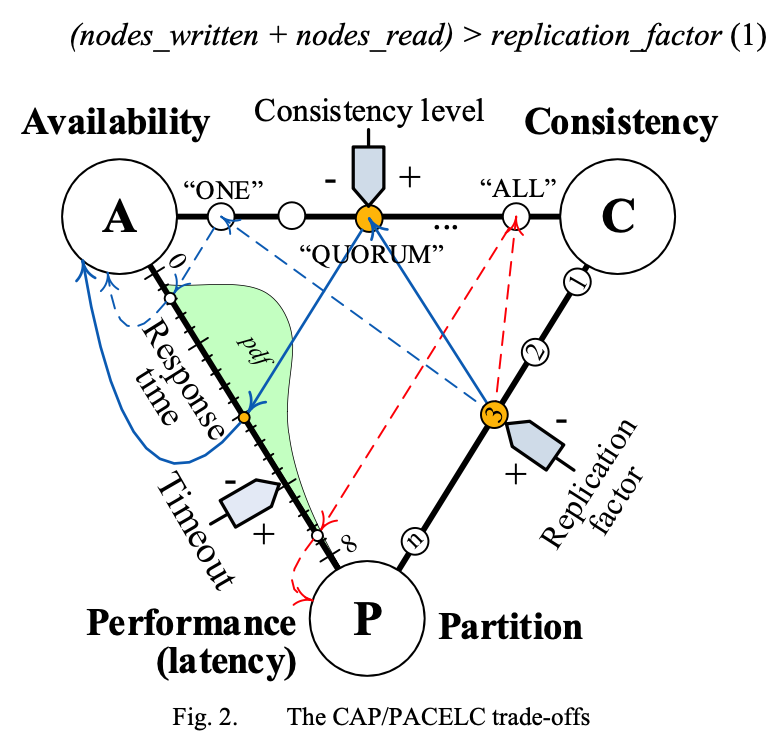
#### Trade-offs Between Consistency, Availability and Latency

Consistency, availability, and latency are closely related to each other in distributed systems. The availability of the system means the receiving of a response within a certain time. Latency directly impacts the use experience. To maintain proper latency the system defines a timeout period beyond which the system considers the response as a failure.

Large-scale systems like Facebook prioritize availability and latency over consistency, by allowing asynchronous data updates that allow some consistency. and high availability.

The consistency model like in Cassandra is based on how many replicas are made for read and write operations. For example, in "ONE: level there is a single replica, which ensures minimal latency compromising consistency. For users who require low latency with less consistency can use level "ONE".

Apart from these, there are also application timeouts that act as the boundary between availability and performance. If the replica fails to respond in a certain time (known as application timeout time) then it is partitioned. The designers decide the time based on the desired response time keeping the trade-off between consistency and availability in mind.



#### Trade-offs Between Performance, Consistency, and Durability

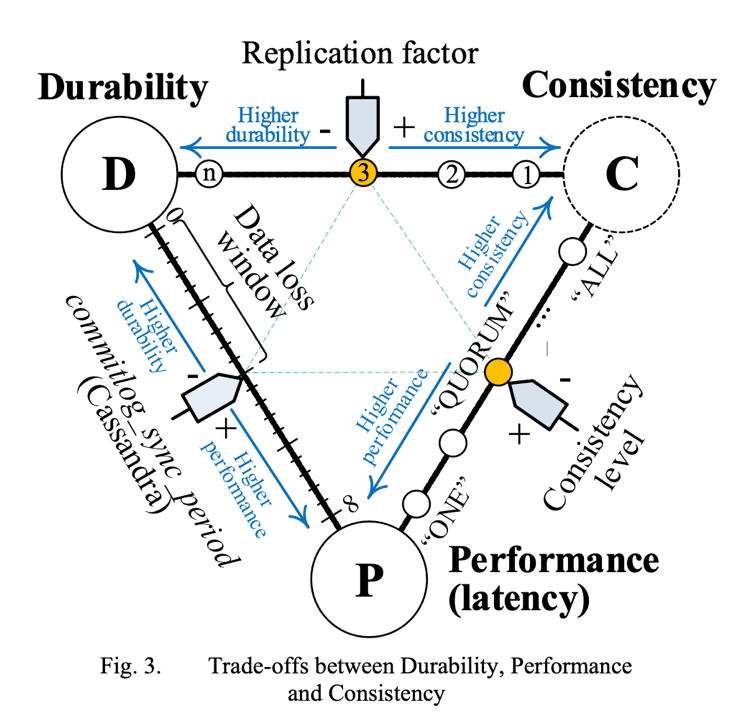
The durability of data is nothing but storing the data in memory and keeping it consistent even in case of system failure or power cuts.

Many NoSQL databases store the data in memory which improves the performance by giving up durability. Many RDBMS like Postgres, MySQL, and Oracle can be configured to perform commit operations in certain time intervals to maintain durability.

The durability of the system can be ensured by commit logging, which records the changes of the database to the user that can be used in rollback operation. The database can also be configured to store the commit log in memory and flush it to disk periodically to improve performance. For example, Cassandra NoSQL flushed a commit logging to disk every 10 seconds.

In the case of disk failure, the logging cannot prevent data loss. The lost data cannot be retrieved on the crashed node when it is being replaced only if the system replicates data across multiple nodes.

This means that to improve durability we need to have high replication of the system which can affect the consistency and latency depending on the chosen consistency level as shown in Fig 3.



#### Trade-offs Between CAP properties, Fault-Tolerance and Energy Consumption

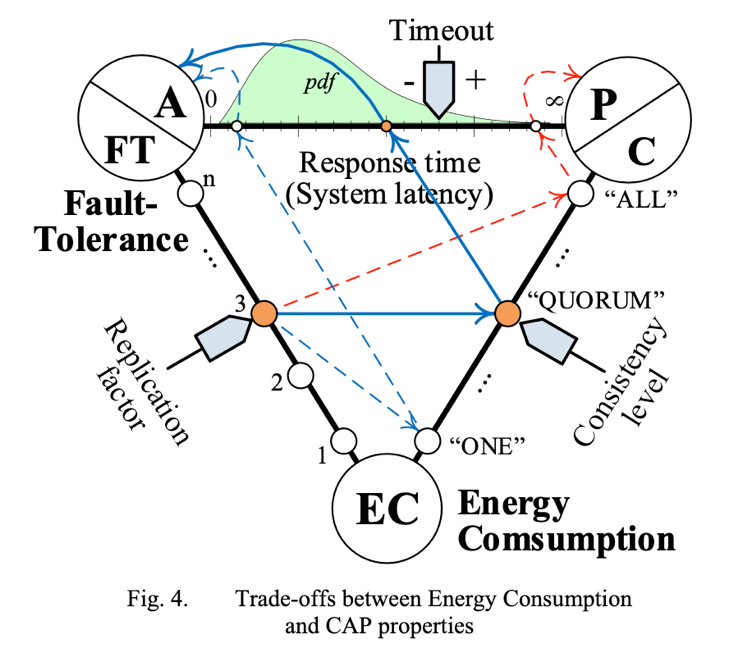
Fig 4 shows the trade-off between CAP properties, fault tolerance, and Energy consumption. To keep these in balance replication plays a vital role.

Replication provides high availability and high performance, while redundancy improves resilience against system failure and crashes. A combination of high replication and strong consistency leads to increased latency reducing responsiveness.

A high level of consistency requires more amount of query processing and replication simultaneously, which means the system requires an extra amount of energy.

This concludes that replication and consistency levels together influence the energy consumption in a distributed system.

The below Fig shows a trade-off between Energy Consumption and CAP properties.



## Research and Implementation of Parallel CART Algorithm Based on Distributed Database

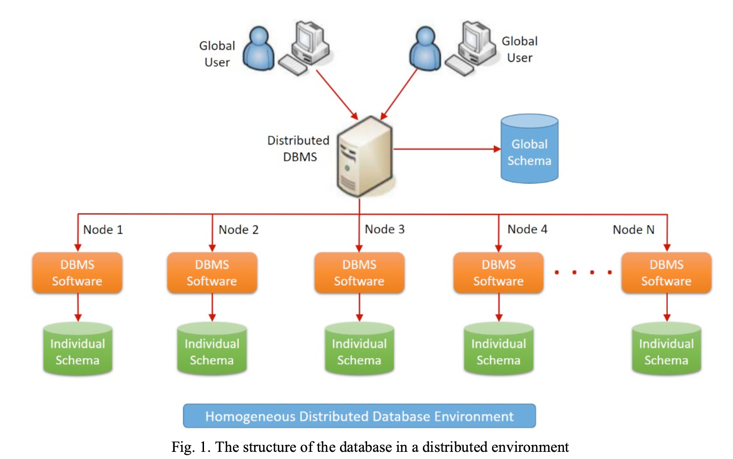
A distributed database is based on a centralized database system but can store data at different locations in the network. They also differ in data processing capabilities due to their storage difference. Apart from scattering and redundant data, distributed query and optimization are also important in actual distributed database applications. The purpose of finding the best database is to reduce the search overhead and reduce the response time to make the system work fast. To decrease the response time we can use multiple computers that can process queries in parallel. The general method to improve the performance in a distributed database is to decompose the query into subqueries, compress the data to multiple locations, and send the intermediate results of each website to the inquiry website to get the final result.

To improve the query efficiency, the communication overheads must be reduced by query processing and query optimization, for example using the semi-join technique.

This project uses the parallel CART decision tree method in the Spark environment to improve its classification accuracy and learning efficiency. It uses vertical data segmentation which reduces communication costs by allowing each node to perform only its calculations. For adjacent attributes, the Fayyad algorithm is used to determine the classification of its edge points which helps in improving the learning speed of the decision tree.

### The Process Model of Executing Data Query in a Distributed Environment

The data is spread across various nodes or sites instead of centralized in one place. For each site, there is a global data model that provides the overall view of the data and a local data model that represents the data at each site in a distributed database environment. Figure 1 shows the structure of the database in a distributed environment.



The above figure shows the structure of the database in a distributed environment. The global users request the data from a distributed database management system that has a global schema. The Distributed DBMS system is connected to multiple nodes which have their own distributed database management system which is associated with an individual schema of their own.

The different components of global data models are the global external model, global conceptual model, and fragmentation model.

Below are the two types of queries when executing in a distributed database:

- Remote query: The job of the remote query is to fetch data from a single remote site and the site close to the querying node to reduce communication costs.

- Global query: The job of the global query is to retrieve data efficiently from multiple sites considering communication cost, efficiency, and speed.

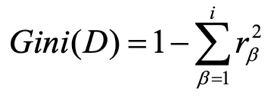
To improve data parallelism and reduce the data transfer overhead the binary semi-join reduction algorithm can be used.

The idea here is, for example, if there are 2 tables R and S which are at different sites A and B. If R and S need to be joined on a common attribute X, then instead of sending the whole table, we can send only send X attribute from site A to site B. Then join the S table with the X attribute table which was sent to reduce the S table to S'. Now send S' to site 1 and join with R table. In this way then communication costs are reduced because we are not sending the whole table. It filters out unwanted data to reduce the amount of transfer from one site to another. In this way, the performance is optimized and communication overhead is reduced.

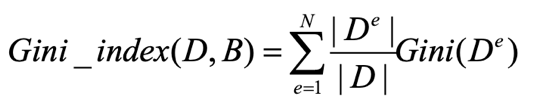
### Algorithm Design

#### CART decision tree algorithm

A classification method is proposed that adopts the classification method of the CART decision tree based on the Gini index. Two types of decision trees are generated by the CART algorithm: classification tree and regression tree. A decision tree works by recursively splitting the data based on certain attributes and the Gini index measures the impurity of a dataset. The Gini index can be calculated as follows.



In the above formula, i ranges from 1 to N which is the number of different values present in dataset D, and rβ is the probability of taking a category β among all the categories. The Gini(D) value is inversely proportional to the purity of a dataset, which means a higher Gini index lowers the purity of a dataset and vice versa. The Gini index of the current attribute β is defined as follows.

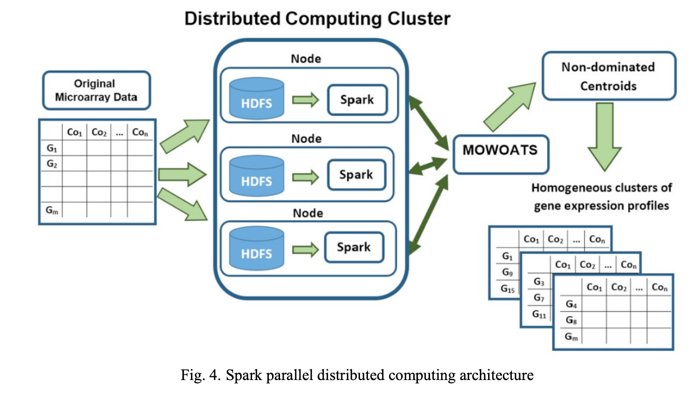


Here e ranges from 0 to N representing the value category of current attribute B. D power e represents all the instances in D taking the value e on the current attribute B. When training the data choose the node with a smaller Gini index.

#### Spark Parallel Distributed Computing Architecture

Apache Spark is a distributed computing architecture that is used to process large amounts of data. SparkCore is the core concept of Spark architecture. The data is stored in space and the intermediate results which are obtained by calculations are rescued by Spark, the benefit is it reduces the frequency of HDFS. reading and writing data and it also improves the efficiency of the system. Spark has multiple components and interfaces that support multiple languages allowing users to meet their requirements at one time which brings them a good experience.

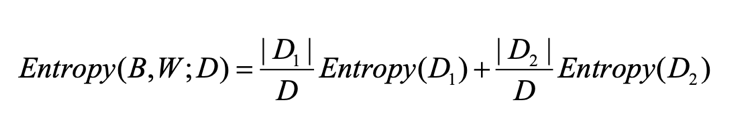
The Spark cluster is based on master/slave architecture. The master node also known as the driver node allocates work to all the slave nodes also known as executors. Figure 4 shows a Spark parallel distributed computing architecture.



#### Fayyad Algorithm

A decision tree deals with continuously splitting the data into smaller chunks. Fayyad algorithm is used to find the best split which minimizes the average class entropy called boundary point.

The working of the algorithm is as follows, first, we sort the data in ascending order then a boundary point is a point where we can split the data such that all the data values less than the boundary point are on the left side and the data values greater than the boundary point is on the right side. The point here is to reduce the entropy between the subsets by minimizing the entropy within each subset. Below is the formula to calculate the entropy.



To implement this method, we first need to find the boundary points, and based on these points we split the data into subsets and then calculate the entropy and choose the optimal point to build the decision tree.

# COMPARISON OF COMPARED TECHNIQUES

## Speed

The speed means the time taken for a query to execute and fetch data, it can also be called response time.

The speed of Paper 2 and Paper 5 depends on the database used and the consistency model. It does not specify the amount of speed improved because it can also depend on the use case where it is used.

In paper 3, the short video management system's speed depends on the initial step. Quick start-up speed is the primary reason to increase the system speed, whereas all other paper's speeds depend on the overall continuous speed of the models/methods. The paper does not mention anything about scalability or how the system can handle large amounts of data.

To handle large volumes of data paper 4 and paper 6 is parallel processing. Both the paper uses parallel processing and multi-node parallel processing which improves the speed. Due to multi-node parallel processing, the communication overheads are also reduced.

## Accuracy

The accuracy of the small file storage method based on Ceph’s discussion in paper 1 depends on the storage utilization. It solves the problem of storage inefficiency due to data holes and metadata. The accuracy of the model is 80% which is higher than all other papers.

Paper 4 defines a model to enhance the accuracy of data protection in the cloud by using fragmentation and distributed storage. The space utilization of the model is not as good as the efficiency and accuracy. In paper 6 the CART method has a very effective way of handling the data query which increases the accuracy of the model.

## Space

Space here means the amount of memory utilized by the system when performing certain tasks.

Paper 1 solves the problem of inefficient utilization of space while storing small files. Small file storage methods based on Ceph, reduce data holes by 805 and inefficient storage of metadata by 50%, which is a significant improvement in space utilization.

Paper 2 which deals with a video management system lacks space efficiency when compared with the NoSQL database due to the presence of a relational database model.

There is a high storage overhead for the method discussed in paper 4 because of the existence of multiple fragments of databases. It has a lot of replicated data in distributed data which is good for faster access but due to replication, the space gets much utilized.

## Performance and Utilization

Paper 1 which discusses storing small files optimally has an improved write performance of small files by 62% by using optimal file merging technique and sequential I/O path. But, it does not mention anything about the read speed and scalability of the model.

For the video management system (Paper 3) the performance in terms of response time is less than 100ms and CPU utilization is less than 10%. The low CPU utilization also means that the resource usage is efficient.

There is a significant performance improvement mentioned in paper 4 when compared with SES 256 encryption because it utilizes parallel processing. It also ensures better resource utilization which is done across distributed nodes.

In the case of paper 5, the performance is impacted by the resource utilization in the distributed system along with the trade-offs between consistency, availability, and energy consumption.

# CONCLUSION

We have seen different data management and system optimization methods. To solve the problem of poor IO performance and waste of data in the Ceph scheme, a small file merging method is proposed which groups small files based on merging. We have also seen how microservices are used to build scalable software systems. The choice of database between SQL and NoSQL must be done based on the application needs to keep the system consistent and optimize performance. Next, we also say a video management system that handles many short videos efficiently, and the advantage over other similar systems is data visualization. We have also discussed the methods to improve security in cloud computing like splitting the data into fragments and storing it at different nodes. Then we analyzed the trade-off in fault-tolerant distributed using and replicated databases. We have seen and compared how performance, durability, consistency, and energy consumption affect their systems and the trade-off of these characteristics in the system. Then we discuss about CART algorithm and the CART decision tree algorithm.

Finally, we have compared all the methods and algorithms based on the speed of the algorithm, accuracy of the system, space utilization, overall performance of the model, and CPU utilization.

# REFERENCES

1. Chao Li; TengFei Wang; Lei Cao; TaoTao Xie, “An Optimized Storage Method for Small Files in Ceph System”, *2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, DOI: https://doi.org/10.1109/ISPDS58840.2023.10235504
2. Madhavi Soni; Varshapriya Jyotinagar, "SQL vs NoSQL Databases for the Microservices: A Comparative Survey", *2023 2nd International Conference on Edge Computing and Applications (ICECAA),* DOI: https://doi.org /10.1109/ICECAA58104.2023.10212190
3. Yixuan Gao; Xinran Ba, "Video Website Management System Based on Database SQL", *2023 International Conference on Culture-Oriented Science and Technology (CoST)*, DOI: https://doi.org/10.1109/CoST60524.2023.00037
4. Nelson L. Santos; Bogdan Ghita; Giovanni L. Masala, "Enhancing Data Security in Cloud using Random Pattern Fragmentation and a Distributed NoSQL Database", *2019 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, DOI: https://doi.org/10.1109/SMC.2019.8914454
5. Anatoliy Gorbenko; Andrii Karpenko; Olga Tarasyuk, "Analysis of Trade-offs in Fault-Tolerant Distributed Computing and Replicated Databases", *2020 IEEE 11th International Conference on Dependable Systems, Services, and Technologies (DESSERT),* DOI: https://doi.org/10.1109/DESSERT50317.2020.9125078
6. Jie Wang, "Research and Implementation of Parallel CART Algorithm Based on Distributed Database", *2023 IEEE 6th International Conference on Information Systems and Computer Aided Education (ICISCAE),* DOI: https://doi.org/10.1109/ICISCAE59047.2023.10392962