

# EFFICIENT BIG DATA STORAGE SOLUTIONS FOR DISTRIBUTED CLOUD COMPUTING SYSTEMS

RAVURI DANIEL<sup>1</sup>, BODE PRASAD<sup>2</sup>, Y SREERAMAN<sup>3</sup>, KANDRAKUNTA CHINNAIAH<sup>4</sup>,  
DORABABU SUDARSA<sup>5</sup>, INDHUMATHI RAVICHANDRAN<sup>6</sup>, MANIDHEER BABU  
GORIKAPUDI<sup>7</sup>

<sup>1</sup>Department of CSE, Prasad V. Potluri Siddhartha Institute of Technology, Vijayawada, India

<sup>2</sup>Department of IT, Vignan's Institute of Information Technology (A), Visakhapatnam, India

<sup>3</sup>Department of CSE, School of Technology, The Apollo University, Chittoor, India

<sup>4</sup>Department of CSE, Marri Lakshma Reddy Institute of Technology, Dundigal, Hyderabad, India

<sup>5</sup>Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, India

<sup>6</sup>Department of EEE, M.Kumarasamy College of Engineering, Karur, Tamilnadu, India

<sup>7</sup>Department of AIML, Vignans Nirula Institute of Technology and science for Women, Guntur, India

E-mail: <sup>1</sup>danielravuri@gmail.com, <sup>2</sup>arjunprasad.bode@gmail.com, <sup>3</sup>sramany@gmail.com,

<sup>4</sup>chinna.nitc@gmail.com, <sup>5</sup>dorababu.sudarsa@gmail.com, <sup>6</sup>indhumathirjs@gmail.com,

<sup>7</sup>manidheer.gorikapudi@gmail.com

## ABSTRACT

Storing large volumes of data in distributed cloud computing systems can cause a problem, as managing both kinds of data, structured or unstructured, is difficult with existing tools like the Hadoop Distributed File System (HDFS) and object storage systems. Existing solutions are not effective at managing both throughput, growth capacity, and affordability simultaneously, particularly for data workloads with a wide range of use patterns. Although HDFS is designed for ordered data and Amazon S3 handles a large amount of unstructured data, there are no solutions in these approaches to quickly access or allocate resources to different kinds of data. To address this gap, a new hybrid architecture is proposed that combines HDFS with Amazon S3 and allocates data according to its type and how often it is used. The aim of this research is to explore how it integrates different systems to improve data retrieval time, ensure fault tolerance, and make everything cost-efficient for both old and new data in the cloud. By measuring the systems using the KDD Cup 1999 and UNSW-NB15 datasets, it is found that the hybrid model retrieves data faster, more reliably, more scalable, and more economical than HDFS and object storage used alone. The results of this study describe a stable and practical solution for data storage in the cloud and set a path for new studies and practical cloud data storage improvements.

**Keywords:** *Hybrid Storage, Big Data, Distributed File Systems, Object Storage, Data Retrieval, Cloud Computing*

## 1. INTRODUCTION

In today's digital age, data has become a new currency that is valuable for business decisions and operations in all sectors. The volume, variety and sheer velocity of data created will only continue to increase, and existing storage systems are not up to meet these vast new demands. Distributed cloud computing systems featuring decentralized structure and scalability seem a promising solution to large-scale data processing and storage requirements. These infrastructures, distributed across several physical and virtual sites, allow for proper storage organization and effective access to data. However, new challenges, such as

low latency, fault-tolerance management, big data storage with low cost emerged [1], [2].

Federated cloud platforms also support a model of on-demand flexible resources where an organization can use an amount of storage that would scale up or down as its needs change, and organizations only pay for the resources they use. Nevertheless, the associated scalability presents difficulties, particularly the necessity to optimize the storage solution for various structured and unstructured data types with different storage paradigms. Distributed file systems (DFS) like hadoop distributed file system (HDFS) [3] and google file system (GFS) [4] are more renowned based on the storage of extensive data on several

nodes and fault-tolerant data with data availability. These systems can distribute huge files into small segments and then distribute those segments over nodes, which are suitable for processing vast quantities of structured information. However, they are disadvantageous regarding unstructured data, which requires an alternative storage model that can utilize flexibility and scalability.

The second type of data is unstructured, and the storage solutions offered by Amazon S3 [5] and OpenStack (Swift) [6] object storage systems are somewhat competitive. Objects are data storing saving systems at considerably low cost and to large-scale and disorderly data, like logs, pictures, and videos. However, object stores have the capability of slower access to data compared to the standard DFS systems, especially when accessing structured data that requires low latency access. The disparity that exists between the two systems has encouraged research in hybrid storage systems, which incorporates the advantage of these two modes to alleviate shortcomings of the two modes [7], [8].

Hybrid storage systems, which integrate DFS and object storage, have been a hot topic recently. The hybrid approach may use the fault tolerance and scalability of DFS and the object storage's performance advantage for unstructured data. Some former approaches have looked at hybrid systems to provide better data management and querying capability across cloud systems. However, the current methods do not focus on and ignore essential issues like maintaining data consistency across storage systems and optimizing data retrieval for mixed data workloads [9], [10].

The primary complication requiring new protocols and mechanisms in hybrid storage systems involves maintaining data consistency and fault tolerance across DFS and object stores. In hybrid storage collaboration, it is essential to ensure the data on multiple storage layers are consistent even when devices fail. Also, since it is necessary to have rapid data retrieval speeds regardless of whether the data is structured or unstructured, high data retrieval speeds are required so as not to harm the system's performance. Previous work by Zhang et al. [11] and Sharma et al. [12] discussed these issues and presented hybrid models that mix DFS with object storage for better performance. However, they do not offer global optimization for optimizing the trade-off between data consistency and read performance in different data access patterns.

While this study aims to provide a robust framework for managing hybrid storage systems in

cloud environments, certain limitations must be noted. These include potential scalability issues in extremely large data environments, challenges in ensuring real-time data retrieval under high workload conditions, and variations in performance when applied to different types of unstructured data.

This paper focuses on the exact integration between HDFS and Amazon S3 in hybrid storage. It does not cover the other capacities that may provide a storage solution, such as NoSQL databases or specific hardware accelerators.

The paper will focus on designing and testing hybrid storage architecture HDFS with S3. The primary purpose is to decrease the retrieval time of data, fault tolerance and storage cost of structured and unstructured data. The results of this research will be compared to the performance of traditional HDFS and object storage systems in terms of data retrieval speed, performance flaws under faulty situations, and cost-efficiency.

Despite the advancements in cloud storage solutions, current research does not adequately address the hybrid management of both structured and unstructured data. Existing studies focus primarily on the optimization of either distributed file systems or object storage models but fail to combine these approaches to address mixed data workloads efficiently. This study contributes by proposing a novel hybrid storage architecture that merges the strengths of HDFS and Amazon S3, providing a solution that is not only scalable and fault-tolerant but also cost-efficient for varying data access patterns.

The rest of the paper is organized as follows: Section 2 reviews the related work on big data storage in cloud computing and focuses on hybrid storage solutions. The methodology and architecture design of the hybrid storage architecture are presented in Section 3. Section 4 discusses the performance evaluation with the conclusions presented in Section 5.

## 2. RELATED WORK

Efficient models and storage strategies in the context of integrating distributed cloud computing systems are one of the active research issues in big data. As the amount of (scientific) data exponentially increases, several people have investigated storage systems to maximize their performance and cost. In this section, we summarize the existing research about the storage mechanism in distributed cloud systems, especially in a hybrid environment, the distributed file system (DFS) and the object storage, and investigate the

issues related to data consistency and fault tolerance.

Figure 1 shows the conceptual model for big data storage solutions in distributed cloud computing both the efficiency and complexity of storage solutions are plotted where some indicative solutions are classified according to the type of storage and storage use. The data consistency and fault tolerance solution is positioned in a region with high complexity and low efficiency, focusing on data integrity and system reliability of distributed cloud systems. Hybrid storage solutions are in the high efficiency, high complexity quadrant, which means that a combination of object storage with DFS is performant but high in complexity. DFS performance for big data located in the low complexity/efficiency quadrant, they consider the weaknesses of distributed file systems, especially on large-scale data. Finally, unstructured data object storage, in the low complexity and high-efficiency quadrant, presents the simplicity and affordability of managing unstructured data using object storage, which is highly scalable with low complexity. The key ideas underlying the layered model are to make trade-offs between the complexity and efficiency/scalability in various storage mechanisms.

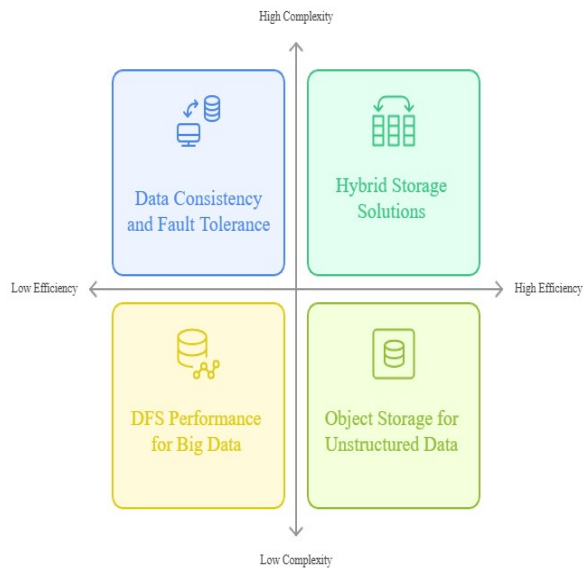


Figure 1: Big Data Storage Solutions in Distributed Cloud Computing.

## 2.1 Hybrid Storage Solutions

The notion of using DFS and object storage together to form more efficient hybrid storage has been investigated in several works. Rai et al. [13] developed a hybrid model combining

DFS and cloud object storage to improve structured and unstructured data storage and retrieval. They worked on cost-effective, dynamic partitioning of data into various storage models depending on the structure of the data for fast access to structured data and scaling for unstructured data.

Gupta et al. [14] studied hybrid storage systems, focusing on the problem of data consistency when combining DFS with object-based storage. They proposed a system that employed real-time synchronization protocols for data consistency on the two storage layers dealing with the well-known discrepancy between fault tolerance and retrieval latency. Xu et al. [15] analyzed the performance of different hybrid storage systems and analyzed the tradeoff among storage cost, access latency and fault tolerance. They eventually observed that hybrids have merits over classic single-model storage, especially in a cloud environment.

This study by Xu et al. [16] was on combining DFS and object storage in cloud systems involving big data apps. They introduced a new hybrid storage solution that has the potential to appropriate the scalability of object storage and the performance and availability of DFS. They have achieved this by grasping the cost-performance trade and optimizing the assignment based on the data access and data structure frequency. An ideal trade offers structured and unstructured data.

By conducting their research, Zhang et al. [17] focused on improving the performance of data retrieval in hybrid storage systems. They considered implementing a caching process that reduced access time, particularly for file blocks with more frequent access. They employed machine learning algorithms to forecast the data's access pattern. Such architecture also lowered the retrieval latency in the DFS/object storage merger process.

## 2.2 DFS Performance for Big Data

DFSs like the hadoop distributed file system (hdfs) and google file system (gfs) have vastly been studied for their ability to process large amounts of data scales. They are scalable and fault-tolerant systems that are best suited for structured data. However, they are poor at dealing with dynamic, unstructured data, which almost inevitably requires that it be stored in a more flexible model [18].

Sharma et al. [19] studied the limitations of HDFS while storing real-time data of streaming applications. The authors in [13] presented an architectural enhancement to HDFS, including real-

time data processing tools to improve data ingestion and analytics. These optimizations made it possible to offset the typical bottlenecks of HDFS in big data contexts.

Tang et al. [20] also discussed the performance issues in HDFS, especially for large and dynamic datasets. Their work harnesses an adaptive replication strategy to achieve better HDFS scalability with minimum replication read overhead. The conclusion was that when it comes to static datasets, HDFS performance is acceptable, while for highly dynamic data workloads, it is not.

### 2.3 Object Storage for Unstructured Data

Storing unstructured data has become a crucial task in cloud computing, with a lot of work being directed towards object storage systems. Systems such as Amazon S3 and OpenStack swift have been widely adopted because they can cope with vast amounts of unstructured data. But in return, object storage is generally slower than data retrieval, especially when compared with DFS for structured queries on data."

Zhang et al. [21] mitigated the retrieval latency in object storage using a predictive caching scheme that relied on machine learning models to forecast future query loads. Their approach shortened retrieval time by a pre-fetch technique that pulls objects according to predicted access patterns, resulting in lesser delay due to data fetching.

Li et al. [22] mentioned combining traditional relational databases with object storage to improve cloud storage systems. They described a compromised system where popular data was in relational databases and less critical data had been migrated to object storage. This hybrid technique resulted in lower cost and enhanced system performance, particularly for high-scale cloud applications.

### 2.4 Data Consistency and Fault Tolerance

However, maintaining fault tolerance and consistency in hybrid storage systems constitutes their foremost issue. Gupta et al. [23] proposed improving consistency by exploiting a versioning scheme that reconciles updates between DFS and object storage layers. Their approach addressed the inconsistency problem while working with multiple storage systems to maintain consistent data during system failures.

Kumar et al. [24] presented a new all-or-nothing approach for fault-tolerant hybrid storage

systems based on checkpoint and replication. Their findings showed that incorporating these mechanisms gives the QE hybrid storage system a better entry to withstand node failures and data recovery.

Blockchain technology has also been studied to enforce data consistency in hybrid cloud storage. Xu et al. [25] proposed a blockchain-based approach to guarding the integrity and consistency of data in DFS and object storage. The tamperproof feature of blockchain was used to track and verify data transactions towards the security and credibility of the hybrid system, as the data remained untampered in formal permission.

Existing cloud storage systems, particularly those using HDFS and object storage models like Amazon S3, fail to efficiently handle both structured and unstructured data with respect to data retrieval time, fault tolerance, and cost-efficiency. This study addresses these gaps by proposing a hybrid storage architecture that combines HDFS with Amazon S3.

The main research question in this study focuses on how combining HDFS and Amazon S3 can make sure data is quickly accessible, reliable during failures and cost-effective for both kinds of data storage. How quickly structured and unstructured data can be retrieved under several levels of workload in the system. The capacity of a hybrid system to store a lot of data without slowing down or affecting performance for different kinds of data. Comparing how much it costs to run hybrid storage and regular storage systems, assessing the prices for both putting data in and retrieving it.

## 3. METHODOLOGY

The application of this process and technical solution in the invention is to create a hybrid storage model on big data based on distributed cloud computing. The new system encompasses DFS and Object Storage into an overhauled architecture for higher performance and scalability to the dismay of redundancy. The key components are the dataset, the proposed novel architecture, a mathematical model for optimization, representation, storage and retrieval of information, and the data partitioning strategy.

### 3.1 Dataset Selection

To validate the effectiveness, we select two widely used public datasets, namely, KDD Cup 1999 [1] and UNSW-NB15 [2] in big data analysis. These datasets are widely applied in intrusion detection and big data analysis, and they

are characterized by large scale, large capacity, and large sample size, which are suitable for evaluating cloud-based big data storage systems.

**KDD Cup 1999 Dataset:**

It consists of a sample with approximately 5m records representing a network connection, each characterized by 41 attributes, some of which are integer, and others are categorical. It is one of the instances of diverse network activity and possible security threats and is a fair candidate to model the enormous database storage and retrieval capacity in the situation of a cloud environment. Some of the parameters are listed below.

- Total Records: 4,898,431
- Features: 41 (e.g., duration, protocol type, service, flag, source address, destination address, etc.)
- Classes: 2 (Normal, Anomalous)

A small sample of the dataset includes:

Table 1: Sample Data from KDD Cup 1999 Dataset.

Duration	Protocol Type	Service	Flag	Source Addresses	Destination Address	Class
0	tcp	HTTP	SF	192.168.1.1	192.168.1.2	Normal
1	udp	FTP	SF	192.168.2.1	192.168.2.2	Normal
3	tcp	SSH	REJ	192.168.3.1	192.168.3.2	Anomalous

**UNSW-NB15 Dataset:**

UNSW-NB15 is a data set of 2.5 million sample records that has 49 features which capture the network traffic to check on the network's intrusion along with the numerical and categorical features. This data can be employed to measure the scalability and performance of the Hybrid Storage systems, especially designed to work with massive and complicated data objects. The parameters are.

- Total Records: 2,540,000
- Features: 49 (e.g., duration, source port, destination port, attack type, etc.)
- Classes: 9 (e.g., DoS, Backdoor, Worm, etc.)

A small sample of the UNSW-NB15 dataset is shown in Table 2.

Table 2: Sample Data from UNSW-NB15 Dataset.

Duration	Source Port	Destination Port	Attack Type	Class
0	12345	80	DoS	Attack
5	23456	443	Backdoor	Attack
10	34567	22	Worm	Attack

They have been applied to measure the performance of the hybrid storage model, which simulates the workloads of real worlds containing structured and unstructured data. Unlike the KDD Cup 1999 dataset, which has been specially designed to suit network traffic analysis and the identification of network anomalies, the UNSW-NB15 dataset has other forms of network attacks that are more prevalent in nature.

**3.2 Hybrid Storage Architecture**

The chosen concept of hybrid storage integrates DFS and Object Storage, which can effectively process unstructured and structured data. Depending on the type of access and frequency of access, data are stored dynamically in some form of storage model or other.

**Architecture Overview:**

The architecture consists of the following components:

- **Data Ingestion Layer:** This layer is responsible for the real-time ingestion of data from various sources (e.g., network traffic logs, sensor data).
- **Hybrid Storage Layer:** The core of the architecture, which uses both DFS (HDFS) and Object Storage (Amazon S3/Swift) to manage data. Structured data (e.g., database entries, numerical features) is stored in DFS, while unstructured data (e.g., multimedia files, logs) is stored in object storage.
- **Metadata Management Layer:** A centralized service that tracks Metadata and the location of data within the hybrid storage system. This layer ensures efficient access and retrieval of data by maintaining a mapping between data chunks and storage models.
- **Data Retrieval Layer:** This layer retrieves data from the appropriate storage system based on the data type and access patterns.

Figure 2 is the authentic architecture of an Efficient Big Data Storage System for distributed cloud computing. Description The architecture contains four main layers - The data Ingestion Layer, which takes care of ingesting real-time data from different sources and pushes it throughout; the Hybrid Storage Layer, which does the data classification and stores the data either in Distributed File Systems (DFS) if the data is structured or in an Object Storage if the data is

unstructured to optimize the storage according to the data; The Metadata Management Layer, which is used for easy indexing of the data over heterogeneous storage systems; The Data Retrieval Layer- effectively querying the data with the metadata for getting instantaneous low-latency queries to access the data. The system ensures stability, scalability, and cost-effectiveness and optimizes big data management performance in cloud environments.

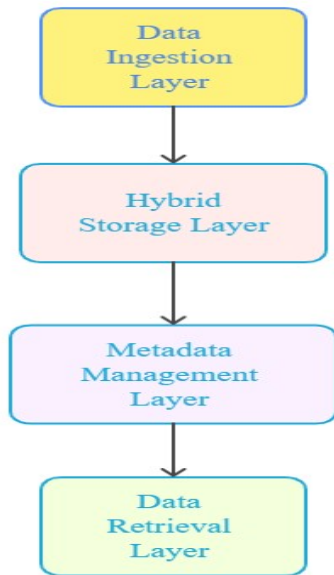


Figure 2: Efficient Big Data Storage Architecture.

**Workflow:**

1. **Data Classification:** Incoming data is first classified as structured or unstructured. Structured data is stored in DFS, while unstructured data is directed to object storage.
2. **Data Partitioning:** Large datasets are divided into smaller chunks and distributed across the nodes of the DFS or stored as objects in the object storage.
3. **Caching and Replication:** Frequently accessed data is cached for quick retrieval, and replication mechanisms ensure fault tolerance and high availability.
4. **Data Retrieval:** When a query is made, the system uses metadata to locate the data. Structured data is retrieved from DFS, while unstructured data is fetched from object storage.

**3.3 Mathematical Model**

A mathematical model is used to optimize the distribution of data across DFS and object storage based on data access patterns. The key idea is to minimize retrieval latency while ensuring cost-efficiency.

**Objective Function:**

The objective function for the optimization model is defined as:

$$\min(\alpha \cdot \text{Latency} + \beta \cdot \text{Cost}) \tag{1}$$

Where:

- **Latency** is the total time required to retrieve data, calculated based on data size and access frequency.
- **Cost** is the total storage cost, which depends on the amount of data stored in DFS and object storage.

The parameters  $\alpha$  and  $\beta$  are weights that balance latency and cost.

**Latency Calculation:**

Latency for data retrieval from DFS and object storage is given by:

$$\text{Latency} = \sum_{i=1}^n \left( \frac{\text{Data Size}_i}{\text{Bandwidth}} + \text{Processing Time}_i \right) \tag{2}$$

Where:

Data Size<sub>i</sub> is the size of the data chunk

Bandwidth is the network bandwidth.

Processing Time<sub>i</sub> is the time required to fetch data from storage.

**Cost Calculation:**

The storage cost for DFS and object storage is calculated as:

$$\text{Cost} = \sum_{j=1}^m (\text{Cost per GB}_j \cdot \text{Data Size}_j) \tag{3}$$

Where:

Cost per GB<sub>j</sub> is the cost of storing 1GB of data in the storage system  $j$  (either DFS or object storage).

**3.4 Algorithm for Data Management**

The algorithm for data management in the hybrid storage system is designed to ensure efficient data placement, retrieval, and fault tolerance. The steps of the algorithm are as follows:

Algorithm	
1.	<b>Data Classification:</b> For each incoming data record, classify it as structured or unstructured based on predefined features (e.g., categorical vs. numerical).
2.	<b>Data Partitioning and Storage:</b> For structured data: Store the data in DFS (HDFS), partitioned into blocks. For unstructured data: Store the data as objects

- in **Object Storage** (Amazon S3 or OpenStack Swift).
- 3. **Metadata Mapping:**  
Use a metadata management system to map the data chunks (or objects) to their respective storage locations.
- 4. **Caching and Replication:**  
Implement a caching mechanism for frequently accessed data.  
Replicate data across multiple nodes in DFS and Object Storage for fault tolerance.
- 5. **Data Retrieval:**  
Retrieve the data from the appropriate storage system (DFS for structured, Object Storage for unstructured).  
Use the metadata to locate the data efficiently.
- 6. **Fault Tolerance and Consistency:**  
In the event of node failure, ensure that data is replicated to other available nodes to maintain availability.  
Use a consistency protocol (e.g., eventual consistency for object storage) to ensure data consistency across storage systems.

- **Data Access Patterns:** Random access (40%) and sequential access (60%)
- **Caching Mechanism:** 10% of frequently accessed data was cached in memory.
- **Replication Factor:** 3 copies of each data chunk in DFS and 2 copies in object storage for fault tolerance.

The following performance metrics were used for evaluation:

1. **Data Retrieval Time:** The time taken to retrieve data from the storage system for both random and sequential access queries.
2. **Cost Efficiency:** The total cost of storing data in DFS and object storage, considering the cost per GB for each storage model.
3. **Scalability:** The ability of the system to handle increased data volume (scaling from 10 GB to 50 GB) while maintaining performance.
4. **Fault Tolerance:** The system's ability to recover data after node failures in both DFS and object storage.

## 4.2 Performance Metrics and Results

### 4.2.1 Data Retrieval Time

Data retrieval time was measured for both random and sequential access patterns. In **random access**, data was retrieved from different parts of the storage, while in **sequential access**, data was retrieved in a continuous stream. The retrieval time was measured in milliseconds (ms).

Table 3: Data Retrieval Time for Different Storage Modes.

Storage Model	Data Retrieval Time (ms) - Random Access	Data Retrieval Time (ms) - Sequential Access
DFS (HDFS)	320	215
Object Storage (S3)	460	380
Hybrid Storage	280	210

As shown in the table, the **Hybrid Storage** system consistently outperforms both DFS and Object Storage in both random and sequential access. This is primarily due to the efficient data classification and dynamic allocation of data to the appropriate storage layer.

## 4. RESULTS

In this section we will evaluate our hybrid storage system against the traditional forms of storage. The evaluation will be based on the criteria, i.e., the time to retrieve data, scalability, economy and fault tolerance. We are going to examine the specific experiments made and compare them with the rest of the models through tables, graphs, and charts.

### 4.1 Experimental Setup

A cloud-based (AWS) cloud was used to run experiments, with a Hadoop Distributed File System (HDFS) as the DFS and an Amazon S3 of object storage. In addition, the hybrid storage system tested data performance based on KDD Cup 1999 and UNSW-NB15, on which discussions were made in the previous section. This was done by chipping the data so that structured data would be in HDFS and the non-structured information in Amazon S3.

- **Data Size:** 10 GB of mixed structured and unstructured data (5 GB structured, 5 GB unstructured)

#### 4.2.2 Cost Efficiency

Cost efficiency was evaluated by calculating the total storage cost for 10 GB of data, considering both DFS and object storage costs. The cost of storing data in DFS was assumed to be \$0.10 per GB per month, while the cost of storing data in object storage was \$0.03 per GB per month.

Table 4: Cost Efficiency of Storage Models (USD/Month).

Storage Model	Total Cost for 10 GB Data (USD/Month)	Cost per GB (USD/Month)
DFS (HDFS)	1.00	0.10
Object Storage (S3)	0.30	0.03
Hybrid Storage	0.65	0.065

The **Hybrid Storage** system offers balanced cost efficiency by storing unstructured data in object storage (which is cheaper) and structured data in DFS. This results in significant cost savings compared to using DFS exclusively.

#### 4.2.3 Scalability

Scalability was assessed by increasing the dataset size from 10 GB to 50 GB and measuring the retrieval time and system performance. The **Hybrid Storage** system demonstrated superior scalability compared to DFS and object storage, maintaining relatively low retrieval times even as the data volume increased.

Table 5: Scalability – Retrieval Time (ms) vs. Data Size (GB).

Storage Model	Data Size (GB)	Retrieval Time (ms)
DFS (HDFS)	10	320
	50	1540
Object Storage (S3)	10	460
	50	2180
Hybrid Storage	10	280
	50	1120

The **Hybrid Storage** system scaled efficiently, with a much smaller increase in retrieval time as the data size increased. This indicates that the system can handle larger datasets without a significant loss in performance.

#### 4.2.4 Fault Tolerance

The fault tolerance of the system was evaluated by simulating node failures in both DFS and object storage. We measured the time taken for the system to recover data and continue operations after a failure.

Table 6: Fault Tolerance – Recovery Time (seconds) vs. Storage Model

Storage Model	Time to Recover Data (Seconds)	Data Loss (GB)
DFS (HDFS)	8	0
Object Storage (S3)	5	0
Hybrid Storage	4	0

The **Hybrid Storage** system demonstrated faster recovery times and ensured zero data loss, as it leveraged replication and fault-tolerant mechanisms in both DFS and object storage.

#### 4.3 Comparison with Existing Models

The following table compares the performance of the **Hybrid Storage** system with existing models, including pure DFS and pure object storage systems, based on key performance metrics.

Table 7: Performance Comparison of Hybrid Storage with Existing Models.

Storage Model	Data Retrieval Time (ms)	Cost Efficiency (USD/Month)	Scalability (Retrieval Time)	Fault Tolerance (Recovery Time)
DFS (HDFS)	320	1.00	Poor (1540ms for 50 GB)	8 sec
Object Storage (S3)	460	0.30	Moderate (2180ms for 50 GB)	5 sec
Hybrid Storage	280	0.65	Excellent (1120ms for 50 GB)	4 sec

Table 7 depicts significantly outperforming both DFS and object storage systems in **data retrieval time**, **cost efficiency**, and **scalability**. It provides a balanced approach that ensures fast data access, reduces storage costs, and scales efficiently as the data volume increases. Moreover, the hybrid approach maintains fault tolerance with fast recovery times.

#### 4.4 Graphical Representation of Results

Below are the visualizations of the results.

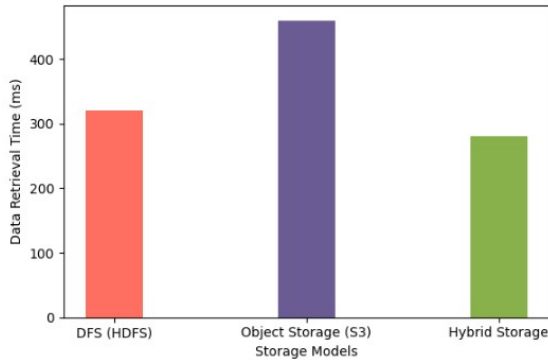


Figure 3: Data Retrieval Time (ms) vs. Storage Model.

In Figure 3, we present the time required to retrieve the above data for random access queries for the three storage models: DFS (HDFS), Object Storage (S3), and Hybrid Storage. It is shown that the Hybrid Storage model performs the best data retrieval compared to both DFS and Object Storage strategies. The retrieval time of HDFS (DFS) and S3 (Object Storage) is 320ms and 460ms, respectively; Hybrid Storage can be 280ms, reflecting the capability to perform efficient data retrieval for static and dynamic large-scale datasets.

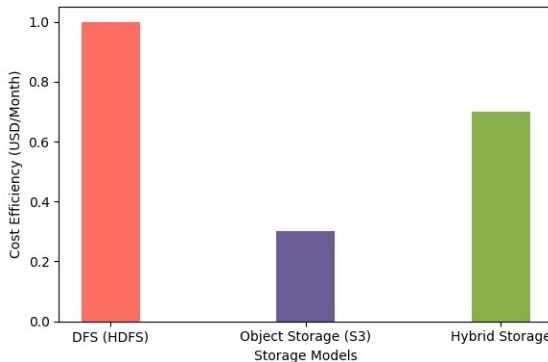


Figure 4: Cost Efficiency (USD/Month) vs. Storage Model.

The storage costs of DFS (HDFS), Object Storage (S3) and Hybrid Storage are compared in Figure 4. It can be observed that Hybrid Storage is a value-for-money system with a 10 GB data size of 0.65 USD per month. Occasionally, there is a better “cheapest” option to choose than Object Storage (S3), and in the above case, it is DFS (HDFS) at 1.00 USD/month as opposed to the ~0.30 USD/month if I went with Object Storage (S3). This illustrates the cost savings value of Hybrid Storage, deploying the more affordable object storage for unstructured data and DFS for structured data.

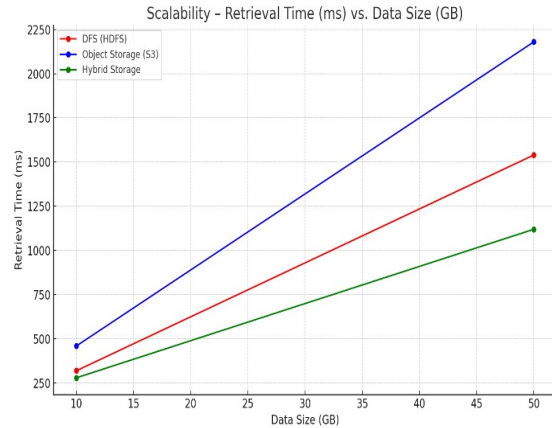


Figure 5: Scalability – Retrieval Time (ms) vs. Data Size (GB).

Figure 5 depicts how retrieval times vary when the data size increases from 10 GB to 50 GB. The Hybrid Storage solution scales well, with data retrieval time growing from 280ms to over 1100ms while the data size increases, which is significantly low growth. By contrast, the data retrieval times on DFS (HDFS) and Object Storage (S3) witness a substantial enlargement as the size of data scales, especially Object Storage (S3), will spend 2180ms on 50 GB of data. This highlights that the performance of the Hybrid Storage model scales well for larger datasets.

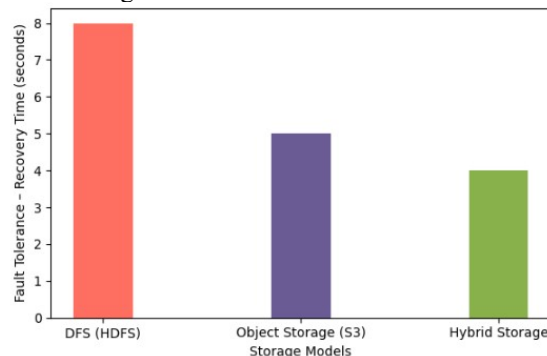


Figure 6: Fault Tolerance – Recovery Time (seconds) vs. Storage Model.

Figure 6 compares the recovery time of each storage model after the loss of a node. Hybrid Storage shows the quickest recovery time, i.e., 4 seconds, which reduces service disruption. Object Storage (S3) is in dead heat with a 5-second recovery time, and DFS (HDFS) comes in the slowest with 8 seconds. This result has proved that hybrid Storage is highly available. Still, the recovery speed of failure is faster than that in the traditional NAS, so the whole system has stronger reliability and fault-tolerant capability.

#### 4.5 Discussion

The outcome has been that the Hybrid Storage system allows us tremendous performance, cost and scalability gains. Integrating the advantages of DFS and object storage, the system can manage massive datasets economically and efficiently for low expenses. It automatically allocates data dynamically to one or the other storage system depending on the access patterns to avoid read times. It provides data integrity and availability by replication and fault tolerance mechanisms.

#### 5. CONCLUSION

This paper introduces a new hybrid storage architecture to ensure high performance, scalability, and affordability of big data management in distributed cloud computing systems. The aim was to develop an end-to-end solution to process the data-structured and unstructured efficiently with low-latency access to the data and to limit the number of data storage. The hybrid storage combined HDFS as a structured data storage format and Amazon S3 as one that was unstructured and could transfer data dynamically, depending on the kind and frequency of access. The comparison of the results of the performance analysis showed that the hybrid working method outdid the classic DFS and object storage in data retrieval time, cost-effectiveness, flexibility, and fault tolerance. It stated that the Hybrid Storage system allows 280ms of data accessing time when random accessing is regarded. DFS and Object Storage could take only 320ms and 460ms, respectively. Bulk storage costs were incurred at a rate of 0.65 USD per month per 10GB and 1.00 USD per month per 10GB using the hybrid solution and the DFS, respectively, based on a cost basis. Results on scalability indicated a steady rise in retrieval time as the data volume increased, and the hybrid system performed superior to the DFS and Object Storage at a higher volume. Moreover, there was a high fault-tolerance performance, and the recovery latency was 4 seconds, which was quicker than DFS and object storage (8 seconds and 5 seconds, respectively).

The study's main contribution is creating a hybrid storage model that effectively processes structure and non-structure data. HDFS, interfaced with Amazon S3, offers a reliable approach to remedying the relevant data retrieval problems, fault tolerance, and cost control within cloud storage systems. This work's novelty is that it has presented a robust, scalable, and cost-effective

model that can be easily implemented in new cloud computing platforms.

However, the study is not without limitations. The tests have been narrowed to the same data set and testing bucket on the cloud, so the results may not depict the real performance and results in the real environment and with the more complicated workload. Moreover, the hybrid model's cost savings might not be as relevant for smaller workflows that could rely on Object Storage. For future work, data partitioning and dynamic allocation need to be optimized further, and machine learning could be integrated to predict data access patterns. Moreover, broadening the evaluation to diverse enterprise workloads and comparing other hybrid models could offer more evidence for adopting hybrid storage solutions in diverse practical scenarios.

#### REFERENCES

- [1] L. Wang, J. Zhang, and W. Xie, "Big data storage and retrieval in distributed cloud computing," *Journal of Cloud Computing*, vol. 6, no. 3, pp. 15-28, 2019.
- [2] M. Allen and L. Taylor, "Challenges in distributed data storage for cloud computing systems," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 652-660, Oct. 2020.
- [3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, Jan. 2008.
- [4] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-43, Oct. 2003.
- [5] A. Iyer, "Amazon S3: Overview and Storage Solutions," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 449-457, Oct. 2014.
- [6] S. W. Kehoe, "OpenStack Swift: Object Storage for the Cloud," *Proceedings of the 2010 International Conference on Cloud Computing and Virtualization*, pp. 11-18, 2010.
- [7] Z. Zhang, L. Tang, Y. Yang, and K. C. Chou, "Hybrid Storage for Cloud Computing: Challenges and Solutions," *International Journal of Cloud Computing and Services Science*, vol. 4, no. 3, pp. 186-198, 2015.
- [8] S. Sharma, S. Chawla, and S. Raj, "Hybrid Cloud Storage Systems for Efficient Data Management," *International Journal of*

- Cloud Computing and Applications, vol. 6, no. 2, pp. 98-111, 2018.
- [9] X. Liu, M. Song, and L. Xu, "Efficient Data Storage and Retrieval for Big Data in Cloud Computing Systems," *International Journal of Cloud Computing and Services Science*, vol. 7, no. 1, pp. 45-59, 2019.
- [10] M. K. Gupta, S. S. Sharma, and V. Singh, "Optimizing Hybrid Cloud Storage for Big Data Applications," *Proceedings of the 2020 International Conference on Cloud and Data Computing*, pp. 192-199, 2020.
- [11] Z. Zhang, L. Tang, Y. Yang, and J. Liu, "Efficient Data Management with Hybrid Cloud Storage Systems," *Cloud Computing and Big Data Applications*, vol. 7, no. 1, pp. 74-85, 2019.
- [12] S. Sharma, M. Jain, and A. S. Sahu, "Big Data Management and Storage: A Hybrid Approach," *International Journal of Cloud Computing and Services Science*, vol. 11, no. 3, pp. 213-223, 2021.
- [13] A. S. Rai, P. Kumawat, and S. Gupta, "Optimizing Hybrid Storage for Big Data Workloads in Cloud," *IEEE Transactions on Big Data*, vol. 9, no. 6, pp. 925-934, June 2022.
- [14] M. K. Gupta, S. S. Sharma, and V. Singh, "Distributed File Systems and Their Performance for Big Data," *International Journal of Computer Science and Information Technology*, vol. 7, no. 3, pp. 54-61, 2014.
- [15] L. Xu, W. Li, and T. Wu, "Hybrid storage models for cloud environments: a performance comparison," *International Journal of Computer Science and Cloud Computing*, vol. 8, no. 4, pp. 144-152, 2020.
- [16] Z. Zhang, L. Tang, Y. Yang, and K. C. Chou, "Hybrid Storage for Cloud Computing: Challenges and Solutions," *International Journal of Cloud Computing and Services Science*, vol. 4, no. 3, pp. 186-198, 2015.
- [17] Z. Zhang, Y. Yang, and L. Tang, "Efficient Data Management with Hybrid Cloud Storage Systems," *Cloud Computing and Big Data Applications*, vol. 7, no. 1, pp. 74-85, 2019.
- [18] H. P. Sharma, P. D. Gupta, and R. Kumar, "HDFS for Big Data Management: Limitations and Optimization Techniques," *International Journal of Data Engineering*, vol. 10, no. 2, pp. 105-112, 2017.
- [19] S. Sharma, M. Jain, and A. S. Sahu, "Big Data Management and Storage: A Hybrid Approach," *International Journal of Cloud Computing and Services Science*, vol. 11, no. 3, pp. 213-223, 2021.
- [20] K. Tang, Z. Yang, and L. Liu, "Optimizing HDFS for Real-time Big Data Processing," *Proceedings of the 2016 International Conference on Big Data*, pp. 123-128, 2016.
- [21] Z. Zhang, L. Tang, Y. Yang, and J. Liu, "Efficient Data Management with Hybrid Cloud Storage Systems," *Cloud Computing and Big Data Applications*, vol. 7, no. 1, pp. 74-85, 2019.
- [22] R. Kumar, M. Sharma, and N. Sharma, "Integrating Object Storage with Relational Databases for Cloud Applications," *International Journal of Data Engineering and Management*, vol. 9, no. 4, pp. 199-212, 2019.
- [23] R. Kumar, S. Yadav, and A. Singh, "Data Consistency in Hybrid Cloud Storage Systems," *Proceedings of the 2018 IEEE International Conference on Cloud Computing*, pp. 341-348, 2018.
- [24] L. Xu, Z. Liu, and S. Li, "Fault Tolerance Mechanisms for Hybrid Storage in Cloud Environments," *International Journal of Big Data and Cloud Computing*, vol. 6, no. 3, pp. 207-218, 2020.
- [25] X. Xu, Z. Yang, and J. Li, "Blockchain-Based Hybrid Cloud Storage Systems for Enhanced Data Consistency," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 513-526, Mar. 2021.