Distributed storage optimization using multi-agent systems in Hadoop

Manar Sais, Najat Rafalia, Rabie Mahdaoui, Jaafar Abouchabaka

Department of Computer Science, Computer Research Laboratory LaRI,
Faculty of Sciences, Ibn Tofail University, Kenitra, Morocco

Abstract. Understanding data and extracting information from it are the main objectives of data science, especially when it comes to big data. To achieve these goals, it is necessary to collect and process massive data sets, arriving at the system in different formats at great velocity. The Big Data era has brought us new challenges in data storage and management, and existing state-of-the-art data storage and processing tools are poised to meet the challenges while posing challenges to the next generation of data. Big Data storage optimization is essential for improving the overall efficiency of Big Data systems by maximizing the use of storage resources. It also reduces the energy consumption of Big Data systems, resulting in financial savings, environmental protection, and improved system performance. Hadoop provides a solution for storing and analysing large quantities of data. However, Hadoop can encounter storage management problems due to its distributed nature and the management of large volumes of data. In order to meet future challenges, the system needs to intelligently manage its storage system. The use of a multi-agent system presents a promising approach for efficiently managing hot and cold data in HDFS. These systems offer a flexible, distributed solution for solving complex problems. This work proposes an approach based on a multi-agent system capable of gathering information on data access activity in the HDFS cluster. Using this information, it classifies data according to its temperature (hot or cold) and makes decisions about data replication based on its classification. In addition, it compresses unused data to manage resources efficiently and reduce storage space usage.

Index Terms— Big Data, Energy consumption, Environmental protection, Storage, Hadoop, HDFS, Multi-Agent system

1 INTRODUCTION

In today's world, the volume of data generated across various sectors is experiencing significant growth. This has led to the emergence of the concept of big data, which offers significant advantages through the utilization of its associated technologies. The practical implementation of big data has become the primary focus for researchers [1], urging us to

carefully consider the most effective approaches for storing, processing, and analyzing such vast amounts of data. As big data technology continues to develop and gain popularity, an increasing number of companies are adopting platform systems based on open-source software like Hadoop[2]. Simultaneously, many companies and applications are transitioning from traditional technical architectures to embrace the potential of big data platforms.[3], [4].

Maximizing the efficiency of data storage is crucial for minimizing energy consumption in big data systems. By implementing efficient data storage methods, companies can significantly reduce the energy required for powering and cooling their data storage equipment. Moreover, organizations can realize cost savings and promote a sustainable, environmentally friendly approach to managing big data by embracing data storage optimization strategies.

Compression and deduplication are key techniques for optimizing data storage, as they eliminate redundant or unnecessary data, resulting in a reduced data footprint. Consequently, storage systems consume less energy and require less storage space. Additionally, data storage optimization enhances data access speed and efficiency, leading to reduced time and energy consumption for retrieving and processing information.

As the storage and processing of big data has become commonplace in recent years, different approaches are being adapted to handle the data and its immensity, such as Hadoop[5]. Apache Hadoop has become the most popular system for processing large datasets, and has become an open-source massive data management software that is adopted in situations where it is necessary to process very large quantities of data. Hadoop provides a scalable, fault-tolerant environment. It enables the processing and storage aspects of a distributed application to be distributed. The Hadoop MapReduce framework serves as the processing component of Hadoop, and the Hadoop Distributed File System (HDFS) serves as the storage component.

The Hadoop project's main component, HDFS (Hadoop Distributed File System), serves as the foundation for distributed computing's data storage management. HDFS is a distributed file system based on a master-slave architecture, enabling the management of large datasets on standard hardware. It was inspired by GFS (Google File System) to guarantee high-speed access to application data and is used to scale a single Apache Hadoop cluster to hundreds or even thousands of nodes. HDFS is based on two daemons, the NameNode and the DataNode. The NameNode runs on the master node, manages system metadata, logically divides files into equal-sized chunks, and controls their distribution within the cluster. Many DataNode processes run on the slave nodes, where they store the data blocks and perform the management tasks assigned to them by the NameNode. They also handle read and write requests from users[6].

As technology advances, many scenarios require new storage constraints, leading HDFS to face new challenges in several areas.

As data continues to grow and accumulate, the popularity of data access varies considerably. For example, one platform will always write the latest data, but the written data will generally not be queried at the same frequency. There is always frequently used data, and data that is rarely used but takes up valuable storage space in the cluster. This can lead to inefficient use of resources as storage is consumed by inactive data, reducing the capacity available for active data, increasing storage costs, and reducing cluster performance. If data adopts the same storage strategy whether it's hot data or cold, this is a waste of cluster resources[7]. It is therefore urgent to optimize the HDFS storage system according to the degree of consultation and use of this data.

A multi-agent system (MAS) is a system of several autonomous entities, called agents, which interact with each other to achieve common or distinct goals. Each agent has a certain degree of autonomy, the ability to perceive the environment and the capacity to

make decisions and act autonomously. Multi-agent systems offer significant advantages in diverse fields such as data management, planning, logistics and monitoring. They have been successfully applied to many problems in a variety of fields, including intelligent transportation [8] or demonstration learning by environmental robots[9], thanks to their ability to model complex problems. The ability to leverage multi-agent systems in the optimization of the Hadoop storage system can lead to systemic benefits. With SMA, the management of hot and cold data in HDFS becomes more automated, adaptive and efficient, guaranteeing rapid availability of hot data while saving storage space for less constrained cold data. This approach facilitates large-scale data management in a Big Data environment, improving the overall performance of HDFS systems.

With the proposed system, agents in a multi-agent system collaborate to manage storage in HDFS. One agent is responsible for collecting information on data access activity. This information is then used by another agent to classify data according to access criteria, grouping them into two categories: frequently used data and rarely used data. A third agent is dedicated to managing the replication and compression of unnecessary data. The remaining sections of this paper are organized as follows: Section II provides an overview of the related work, Section III delves into the background information, Section IV introduces the proposed system, and Section VI offers a concluding summary of the paper.

2 RELATED WORK

The development of a storage monitoring and management system for the HDFS cluster has always been an active area of research. The study presented in[10] introduces a data access monitoring and replication control management system that categorizes data as hot, warm, or cold based not only on its age but also on the number of accesses it receives. This approach provides a suitable labeling system for data classification and management.With the proposed system, data can move from one category to another (cold to hot and vice versa) if their number of accesses reaches a certain value, with greater flexibility and storage efficiency.

The authors in [11] present a technique for storing data in HDFS, which takes data temperature into consideration to dynamically and automatically move data between these different storage levels, by moving "cold" data to inexpensive archival storage and "hot" data to faster storage. In this way, the cluster adapts over time to the characteristics of the workload in order to make efficient use of scarce, expensive storage space.

In the same context, Kaushik et al [12] present the GreenHDFS technique. This approach is based on the process of predictive data replication, using file heat prediction for replica creation and deletion. file's heat is calculated as a function of the total number of accesses to the file and its lifetime. Thus, files are classified into hot and cold files according to their heat. Replicas for cold files are deleted, while replicas are created for hot files.

The aim of this paper [13] is to present ERMS (Elastic Replication Management System for HDFS), an elastic replication management system for HDFS that provides an active or backup model for HDFS storage. This system uses a complex event processing engine to classify different types of data, then dynamically places additional copies for data considered hot. When this data becomes cold, ERMS applies erasure coding to it.

In the same context, but using multi-agent systems, this work [14]makes several major contributions. First, it proposes to create an agent-based simulation framework that is scalable, dynamically extensible, fast, fault-tolerant and failure-resistant on Hadoop. In addition, it proposes optimization techniques for implementing this simulation framework on a Hadoop cluster. These techniques include caching intermediate results to speed up operations, fast retrieval of agent data and messaging using Lucene indexes, and a

clustering algorithm for frequently communicating agents. These approaches aim to improve the performance and efficiency of the simulation framework by using the advantages of multi-agent systems in a Hadoop environment.

3 BACKGROUND

The Hadoop framework includes a system that meets the storage needs of Big Data applications, known as the Hadoop Distributed File System (HDFS). HDFS is a distributed, scalable and highly fault-tolerant file system, designed to operate in low-cost hardware environments. Over a decade has elapsed since the inception of Hadoop, and during this time, HDFS technology has continued to evolve. Certain existing HDFS technologies have made significant advancements in addressing specific challenges associated with Hadoop.

3.1 HDFS storage architecture

Hadoop Distributed File System (HDFS) is a distributed storage system part of the Hadoop ecosystem. It is designed to store and manage large amounts of data on server clusters. The Hadoop Distributed File System architecture[15], including the daemons, is illustrated in Figure 1. Three main daemons can form a standard Hadoop cluster. These are the master daemon (Namenode), the client daemon, and the slave daemon (Datanode). Each daemon has its own participants. The Namenode stores files metadata such as directory structure, block namespace, and access permissions, as well as the location of replicas of each block[16]. NameNode stores this information in memory to speed up read/write operations. It also handles client requests such as file creation, file deletion, and so on. DataNodes store actual data at the same time. Each DataNode manages a set of data blocks, which are stored on the local node's file system, and it also manages read and write operations on the data blocks assigned to them.

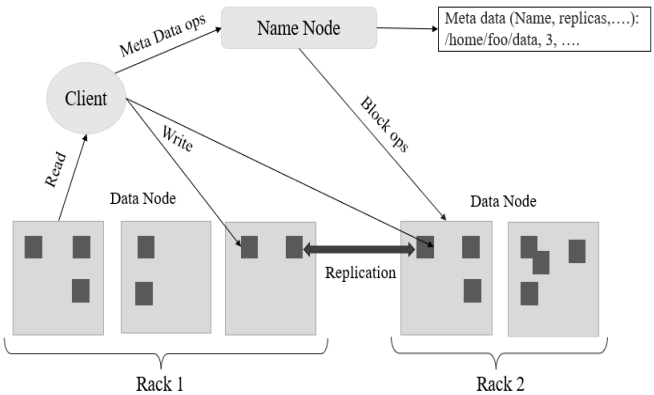

Fig. 1. Apache HDFS architecture.

HDFS includes three types of file operations, read, write and delete, which are managed by NameNode and DataNode in the system's distributed architecture. When a client needs to store data, it sends a written request to the NameNode. The Namenode checks the file's metadata and selects three Datanodes to replicate the data blocks. The client divides the file into data blocks and sends them to the corresponding DataNodes. Once the data blocks have been received and stored, each block is hosted on multiple DataNodes to provide fault tolerance[17]. When reading data, the client sends a read request to the NameNode,

specifying the file to be read, and the NameNode checks the file's metadata and performs certain operations to tell the client where to read the data. Clients generally read each block from the first data node[18].

3.2 HDFS replication

A key feature of HDFS is data replication, which offers several advantages. Firstly, it ensures greater fault tolerance thanks to data availability and system resilience. What's more, this feature makes the system more effective by simultaneously distributing read and write loads across several nodes. It's important to note that, despite the advantages of data replication on the system, such duplication can lead to storage overload. HDFS does not differentiate between data at the replication level. Frequently used data and less-used data are distributed across the cluster with the same replication factor. Replicating unused data can lead to problems such as inefficient use of storage space, additional cost and management complexity

HDFS uses a three-way replication strategy, where there are two additional copies, resulting in a storage overhead of 200%. The first replica is placed on the same node, while the other two replicas are placed on different racks [10]. Storage efficiency is calculated as follows:

$$SE = \frac{\text{Size of original data blocks}}{(\text{Size of original data blocks} + \text{Size of replicated data})}$$

3.3 Hot vs Cold data

In HDFS, the two concepts hot data and cold data refer to the frequency of use of data in a Hadoop cluster. Cold data is old data that is either inactive for a long time or rarely accessed or used. For example, it may be inactive server logs, old archived data or historic This type of cold data still takes up storage space, but is considered a lower priority in terms of access. Hot data, on the other hand, is fresh data that is frequently accessed or actively used by applications or processing tasks in the Hadoop cluster. Hot data requires fast availability and accessibility. Additional copies are therefore needed for hot data to manage node failures smoothly [10].

In a Hadoop cluster, the distinction between hot and cold data enables optimized use of storage resources in HDFS. Cold data does not require additional replication, thus saving storage space, while hot data benefits from higher replication to guarantee availability. This data classification makes it possible to better manage resources and adjust the replication strategy to the specific needs of the data in the Hadoop cluster. The categorization of data varies from organization to organization, and there is no standard approach.

Table 1. Organization Policy Example

Data category	Data age	Access Frequency
Hot	< 7 days	10x/day
Cold	1 month < age	2x/month

To illustrate, consider an organization that adopts the following policy: data younger than 7 days and accessed 10 times a day is categorized as hot data. Conversely, data older than 1 month and accessed twice a month is classified as cold data.

4 SYSTEM DESIGN AND IMPLEMENTATION

The proposed system is based on the use of a multi-agent system (MAS) to manage and optimize storage in HDFS. It consists of autonomous agents that interact with each other to achieve specific goals. Each agent plays a specific role and has a certain degree of autonomy and the ability to make decisions and act independently. The main objective of this system is to improve the efficiency and resource utilization of Hadoop clusters by identifying and managing frequently used data (hot data) and rarely used data (cold data).

The system consists of three agents, as shown in the figure 2. In HDFS, read and write operations performed by clients are recorded in log files. These logs record all operations and transactions performed on the files, including reading, writing, deleting, moving, etc. In our system, Agent 1 is responsible for data monitoring. It gathers information on file access activity in an HDFS cluster from log files, and extracts file metadata to determine access frequency and time of last use.

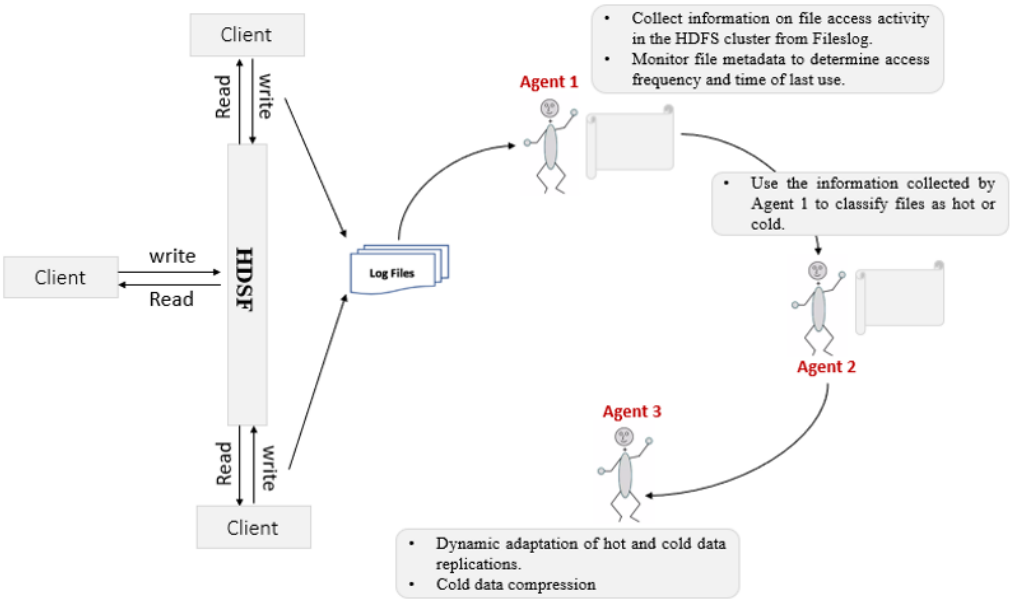

Fig. 2. Descriptive diagram of proposed system

The information entered by Agent 1 is used by Agent 2 to classify files as hot or cold. Here's the scenario: if a file is less than 7 days old and is accessed several times a day, it is classified as hot. On the other hand, if the age of a file exceeds one month and it is accessed once or twice a month, the file is classified as cold. This classification enables us to distinguish between data requiring rapid availability and accessibility, and data that may be of lower priority.

Agent 3 is responsible for two tasks: managing the data replication factor, based on the classification performed by Agent 2, and data compression. When a file is added to HDFS, its replication factor is initially set to 3 and remains at 3 until the category age limit is reached, regardless of the number of accesses. The system manages replication as follows: if a file is classified in the hot category, the replication factor is maintained. On the other hand, if a file is classified as cold, existing replicates are reduced and one replicate is replaced by a compressed copy. Agent 3 applies data compression techniques to save storage space, as this data does not require immediate accessibility.

To improve system performance, this optimization task has been automated for a period to be specified. Agents gather information, classify files and take appropriate action for data replication and compression. This automation ensures that the system remains up-to-date and reacts proactively to changes in data usage in the HDFS cluster.

Prior to running the multi-agent system for storage optimization, a Hadoop cluster operates in a traditional configuration where data is evenly distributed across multiple storage nodes, regardless of usage or frequency of access. Our storage system (DataNodes) comprises 1049 Blocks, and the Block pool used is 9.55 BG as shown in figure 3.

Fig. 3. The cluster state before running the system

However, once the multi-agent system is implemented and run on the Hadoop cluster, storage dynamics are completely transformed. Agents begin to collect information on file access activity, monitor file metadata, and classify data into hot and cold based on the frequency of use.

To do this, the agents generate three files. Firstly, Agent 1 generates the "FileTimestamp" file, which contains the time of every time each file is accessed. Next, agent 2 generates the "FilesAccessTimes" file, which records the frequency of access to each file, making it possible to determine whether a file is hot (frequently accessed) or cold (rarely accessed). Finally, file 3 "coldFiles, lists files identified as cold, i.e. those not frequently used.

4.1 Replica reduction and cold file compression

As shown in Figure 4, the system also reduces the number of replicas for cold files from three to just two. if the file is very unlikely to be used, it will have just 1 replicate and a compressed version. This compression reduces the size of the file, optimizing the use of storage space. The compressed copy preserves data integrity and accessibility while saving valuable resources.

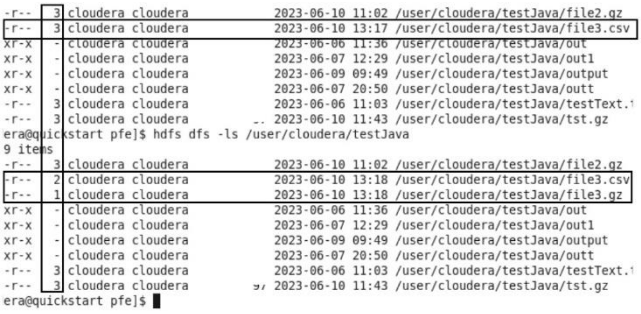

Fig. 4. Reducing Replicas of a Cold File and data compression

4.2 Storage space reduction

Through analysis and optimization of our storage infrastructure, we freed up approximately 4GB of storage space from a total of 10.384228116GB. By consolidating redundant files and implementing compression algorithms on inactive data, we reduced

storage requirements across the pools. The combined storage savings from the pools amounted to 4337548776 bytes, equivalent to 4.337548776 GB of storage space now available for active data and new storage needs, the result is illustrated in Figure 5.

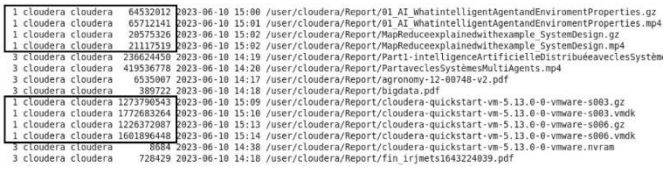

Fig.5. Reduced storage space

5 CONCLUSION

This work proposes an approach to Hadoop storage system management that integrates the concept of multi-agent systems into data management in HDFS. The system offers an automated, proactive and innovative approach to hot and cold data management in Hadoop environments. Using a set of three autonomous agents, the system enables proactive data management based on data age, frequency of access and usage. Future prospects for integrating multi-agent systems into data optimization on Hadoop may include the integration of advanced machine learning techniques where agents can use learning algorithms to better understand data usage patterns and make more informed and intelligent decisions about data replication, compression and placement.

This work was supported in part by the National Center for Scientific and Technological Research (CNRST) and this within the program of the research grants initiated by the Ministry of National Education, Higher Education, Management Training and Scientific Research.

References

[1] C. Kacfah Emani, N. Cullot, and C. Nicolle, "Understandable Big Data: A survey," Computer Science Review, vol. **17**, pp. 70–81, Aug. (2015), doi: 10.1016/j.cosrev.2015.05.002.

[2] A. Caggiano, "Cloud-based manufacturing process monitoring for smart diagnosis services," International Journal of Computer Integrated Manufacturing, vol. **31**, pp. 1–12, Jan. (2018), doi: 10.1080/0951192X.2018.1425552.

[3] S. Ibrahim, T. Phan, A. Carpen-Amarie, H.-E. Chihoub, D. Moise, and G. Antoniu, "Governing Energy Consumption in Hadoop through CPU Frequency Scaling: an Analysis," Future Generation Computer Systems, vol. **54**, Feb. (2015), doi: 10.1016/j.future.2015.01.005.

[4] D. M. Nascimento, M. Ferreira, and M. L. Pardal, "Does Big Data Require Complex Systems? A Performance Comparison Between Spark and Unicage Shell Scripts." arXiv, Dec. 27, (2022). Accessed: Jun. 10, 2023. [Online]. Available: http://arxiv.org/abs/2212.13647

[5] S. Chandrasekar, R. Dakshinamurthy, P. G. Seshakumar, P. Balasundaram, and C. Babu, A novel indexing scheme for efficient handling of small files in Hadoop Distributed File System. (2013), p. 8. doi: 10.1109/ICCCI.2013.6466147.

[6] V. Rao Chandakanna, "REHDFS: A random read/write enhanced HDFS," Journal of Network and Computer Applications, vol. **103**, pp. 85–100, Feb. (2018), doi:

10.1016/j.jnca.2017.11.017.

[7] B. Mao, H. Jiang, S. Wu, Y. Fu, and L. Tian, "Read-Performance Optimization for Deduplication-Based Storage Systems in the Cloud," ACM Transactions on Storage (TOS), vol. **10**, Mar. (2014), doi: 10.1145/2512348.

[8] Liu Jiang, Bing Li, and Meina Song, "THE optimization of HDFS based on small files," in 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China: IEEE, Oct. (2010), pp. 912–915. doi: 10.1109/ICBNMT.2010.5705223.

[9] N. Verstaevel, J. Boes, J. Nigon, D. d'Amico, and M.-P. Gleizes, Lifelong Machine Learning with Adaptive Multi-Agent Systems. (2017). doi: 10.5220/0006247302750286.

[10] M. Guériau, F. Armetta, S. Hassas, R. Billot, and N.-E. E. Faouzi, "A constructivist approach for a self-adaptive decision-making system: application to road traffic control," presented at the 28th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Nov. (2016), p. 670. doi: 10.1109/ICTAI.2016.0107.

[11] S. Shamraj and P. J. Kulkarni, "Data Access Monitoring and Replication Control Management System for HDFS Clusters," in 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India: IEEE, May (2018), pp. 2342–2345. doi: 10.1109/RTEICT42901.2018.9012567.

[12] R. Subramanyam, "HDFS Heterogeneous Storage Resource Management Based on Data Temperature," in 2015 International Conference on Cloud and Autonomic Computing, Boston, MA, USA: IEEE, Sep. (2015), pp. 232–235. doi: 10.1109/ICCAC.2015.33.

[13] R. Kaushik, T. Abdelzaher, R. Egashira, and K. Nahrstedt, "Predictive data and energy management in GreenHDFS," Jul. (2011), doi: 10.1109/IGCC.2011.6008563.

[14] Z. Cheng et al., ERMS: An elastic replication management system for HDFS. (2012), p. 40. doi: 10.1109/ClusterW.2012.25.

[15] R. Sanchez, "A Multi agent Simulation Framework on Small Hadoop Clusters", Accessed: Jun. 10, (2023). [Online]. Available: https://www.academia.edu/19597670/A_Multi_agent_Simulation_Framework_on_Small_Hadoop_Clusters

[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), May (2010), pp. 1–10. doi: 10.1109/MSST.2010.5496972.

[17] Y. Tian and X. Yu, "Trustworthiness study of HDFS data storage based on trustworthiness metrics and KMS encryption," in 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China: IEEE, Jan. (2022), pp. 962–966. doi: 10.1109/ICPECA51329.2021.9362537.

[18] M. Sais, N. Rafalia, and J. Abouchabaka, "Intelligent Approaches to Optimizing Big Data Storage and Management: REHDFS system and DNA Storage," Procedia Computer Science, vol. **201**, pp. 746–751, Jan. (2022), doi: 10.1016/j.procs.2022.03.101.