

Enhancing Hadoop distributed storage efficiency using multi-agent systems

Rabie Mahdaoui, Manar Sais, Jaafar Abouchabaka, Najat Rafalia

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

Article Info

Article history:

Received Dec 4, 2023

Revised Jan 24, 2024

Accepted Feb 23, 2024

Keywords:

Big data

Distributed storage

Hadoop distributed file system

Multi-agent systems

Node performance

ABSTRACT

Distributed storage systems play a pivotal role in modern data-intensive applications, with Hadoop distributed file system (HDFS) being a prominent example. However, optimizing the efficiency of such systems remains a complex challenge. This research paper presents a novel approach to enhance the efficiency of distributed storage by leveraging multi-agent systems (MAS). Our research is centered on enhancing the efficiency of the HDFS by incorporating intelligent agents that can dynamically assign storage tasks to nodes based on their performance characteristics. Utilizing a decentralized decision-making framework, the suggested approach based on MAS considers the real-time performance of nodes and allocates storage tasks adaptively. This strategy aims to alleviate performance bottlenecks and minimize data transfer latency. Through extensive experimental evaluation, we demonstrate the effectiveness of our approach in improving HDFS performance in terms of data storage, retrieval, and overall system efficiency. The results reveal significant reductions in job execution times and enhanced resource utilization, thereby offering a promising avenue for enhancing the efficiency of distributed storage systems.

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



Corresponding Author:

Manar Sais

Computer Research Laboratory LaRI, Department of Computer Science, Faculty of Sciences

Ibn Tofail University

Kenitra, Morocco

Email: manar.sais@uit.ac.ma

1. INTRODUCTION

In today's data-driven era, the efficient management of vast amounts of data is imperative for the success of a wide range of applications, from scientific research to business analytics. Distributed storage systems, such as the hadoop distributed file system (HDFS), have emerged as critical components in addressing the scalability and fault-tolerance requirements of these applications. HDFS is primarily designed for distributed data storage management, and its main objective is fault tolerance and balanced data distribution across different cluster nodes. HDFS does not natively have built-in functionality for dynamic decision-making to determine which network path is the most optimal. Fluctuations in the network can make this assessment complex, requiring sophisticated mechanisms to adapt to real-time variations in network connectivity. On the other hand, prioritizing machines with the highest computing capacity poses challenges in accurately determining node performance. Workload variability can make it difficult to predict actual node performance, which can lead to sub-optimal decisions when selecting machines for specific tasks. Combining these two issues, the need to find solutions that optimize the management of large files while considering performance variations between nodes becomes crucial. An approach that enables dynamic, intelligent allocation of storage tasks according to specific node performance could alleviate these challenges and

significantly improve storage efficiency in HDFS. However, optimizing the performance and resource utilization of such distributed storage systems remains a challenging task. Recent research efforts have explored various strategies to enhance the efficiency of distributed storage systems [1]. Previous studies have proposed solutions ranging from load balancing algorithms to data replication strategies, all aimed at improving the overall system performance [2].

Building upon the foundation laid by previous research on distributed storage systems, such as the seminal work on HDFS by Shvachko *et al.* [3] which laid the groundwork for modern distributed file systems, and the research on agent-based systems by Wooldridge [4] showcasing the potential of agents in decentralized decision-making. Another innovative algorithm has been introduced with a dual focus on optimizing resource provisioning in Hadoop MapReduce [5], [6] prioritizing both performance and energy efficiency. This algorithm shows promise in enhancing resource utilization and reducing execution times. However, these approaches often overlook a critical dimension: the dynamic nature of node performance in distributed environments.

Recognizing this gap, our research focuses on a novel paradigm to optimize HDFS using multi-agent systems (MAS) [7]-[9] which provides a decentralized decision-making framework capable of adapting to real-time variations in node performance. This study investigated the effects of dynamic storage task allocation by intelligent agents on performance optimization. While earlier studies have explored various aspects of distributed storage systems, they have not explicitly addressed the impact of intelligent agents in dynamically allocating storage tasks to nodes based on their performance profiles. This approach represents a departure from traditional centralized approaches and holds the potential to effectively mitigate performance bottlenecks while optimizing resource utilization in distributed storage environments.

Incorporating MAS into the Hadoop framework holds promise for boosting performance [10]. This approach utilizes multiple agents' collective capabilities to improve task handling and system coordination [1], [11], potentially optimizing resource allocation and load balancing, consequently leading to a noticeable boost in the system's overall performance [12]. MASs in Hadoop facilitate distributed data mining through agent protocols and message-based communication, enabling scalability and dynamic load balancing [13], [14]. In sum, our study suggests that higher integration of MASs into Hadoop stands as a promising approach with the capability to significantly enhance resource utilization, streamline coordination, and ultimately elevate the performance of data processing and analytical tasks within the framework [15].

This paper presents the culmination of our research efforts in the realm of distributed storage efficiency enhancement through MAS. Through an empirical evaluation, we evaluate our approach against existing methods and demonstrate its effectiveness, particularly in terms of reducing job execution times, resource utilization, and overall system performance. Our contribution not only extends the existing body of research on distributed storage optimization but also opens new horizons for addressing the evolving challenges of managing distributed data efficiently. In the following sections, we will first offer an overview of the related work. Then, we will dive into the specifics of our MAS-based approach, conduct a thorough analysis of our experimental results, and finally, explore the implications of our findings for the broader context of optimizing distributed storage systems.

2. RELATED WORK

Several research papers have made substantial contributions, each aimed at bolstering the performance, resource allocation, and fault tolerance of these distributed computing frameworks [16]. This section delves into key research papers, providing insight into their contributions while also addressing the inherent limitations associated with their respective approaches. Rasooli and Down [17] presented an adaptive scheduling algorithm, tailored to address the unique challenges posed by dynamic and heterogeneous Hadoop systems. The primary aim is to enhance the mean completion time of submitted jobs. This adaptive approach is invaluable for accommodating varying workloads and resource availability, yet it may confront challenges in effectively handling extreme workload fluctuations and in preserving the delicate balance between resource allocation and job execution times. Looking ahead, a hybrid scheduling algorithm has been introduced within the Hadoop MapReduce framework [18]. It places particular emphasis on the optimization of data locality rates and job completion times. While this research strives to strike a balance between resource utilization and job execution durations, it may encounter challenges in situations marked by highly dynamic and unpredictable data processing workloads.

Focusing on the central concern of data locality in Hadoop, dynamic schedulers have been introduced to optimize makespan and communication for MapReduce tasks involving replicated inputs [19]. These dynamic scheduling strategies show promise in augmenting the system's overall performance and resource utilization. Nevertheless, their effectiveness might be influenced by irregular data access patterns or the administration of extensive clusters accommodating diverse workloads. Moving forward, An innovative algorithm has been presented for enhancing the scheduling of MapReduce tasks on Hadoop [20],

demonstrating substantial performance enhancements. Nonetheless, its appropriateness for exceptionally dynamic and diverse settings, along with the possible resource overhead during implementation, requires thoughtful evaluation.

With a focus on fault tolerance and failure handling, an evaluation has been conducted to assess the performance of Hadoop's schedulers when faced with failures [21]. While this examination delves into potential avenues for improving Hadoop's performance and resource allocation strategies in the event of unforeseen failures, it also underscores the need for additional exploration into the scalability of these strategies and their potential influence on the overall system performance. A study has been conducted on data placement policies in Hadoop, presenting a modified approach to enhance system performance, especially in environments characterized by heterogeneity [22]. However, striking a harmonious balance between optimizing data placement and accommodating real-time data access patterns may present its own set of challenges. In the wider landscape of cloud computing data centers, an exploration has been conducted concerning the optimization of software architecture by integrating Hadoop and MapReduce [23]. This research offers valuable insights into the interplay between these frameworks and cloud computing.

A modeling-based optimization approach is presented to improve the performance of Hadoop [24]. It promises a reduction in execution time and provides information on the selection of execution techniques and parameters. However, limitations may arise due to the complexities involved in creating and maintaining accurate models, especially in dynamic computing environments. One study examines different job scheduling algorithms to improve MapReduce job execution in Hadoop [25], offering insights into approaches that can be used to improve job scheduling. While this work has enriched the field of Hadoop and distributed computing, its limitations highlight the need for innovative solutions to address the multiple complexities of modern data-intensive applications and distributed systems, as presented in this article.

3. BACKGROUND

Within the realm of Hadoop, there exists a remarkable system tailored to quench the insatiable thirst for storage that big data applications possess—the HDFS for short [26], [27]. Picture HDFS as a sprawling, distributed giant, capable of seamlessly expanding to meet your data storage needs. It's not just a file system; it's a juggernaut of distribution, scalability, and resilience, custom-made to thrive in the world of budget-friendly hardware. Now, let's journey through the annals of time, more than a decade since Hadoop's inception. During this epoch, the HDFS technology has been a phoenix, perpetually reborn and refined. Certain facets of HDFS have undergone substantial transformations, evolving to tackle the unique and formidable challenges that Hadoop and big data bring to the table. These advancements are nothing short of remarkable in their relentless pursuit of enhancing the Hadoop ecosystem.

3.1. HDFS storage architecture

HDFS is a pivotal component in the field of distributed data storage, playing a fundamental role in the efficient management and retrieval of large-scale data across distributed clusters [28]. HDFS employs a unique architecture designed for fault tolerance, scalability, and high availability. At its core, HDFS divides large data files into smaller blocks, distributing them across a cluster of commodity hardware. This distributed approach ensures that data is both redundant and accessible, even in the face of hardware failures. HDFS relies on a master-slave architecture, comprising two main components: the NameNode and DataNodes. The NameNode serves as the central coordinator, storing metadata and namespace information, while DataNodes store the actual data blocks. Additionally, HDFS utilizes a replication strategy, replicating data across multiple DataNodes to ensure data durability and availability. As a result, HDFS has become the cornerstone of many big data applications, enabling researchers and organizations to harness the power of distributed computing for processing and analyzing vast datasets. Understanding the intricacies of HDFS storage architecture is crucial for optimizing the performance and reliability of such data-intensive research endeavors.

3.2. DataNode selection in HDFS

HDFS is an astute decision-maker when it comes to selecting DataNodes for storing data. It relies on two fundamental principles to guide its choices, each contributing to the efficiency of data storage and retrieval [29]. One of HDFS's foremost considerations is minimizing network traffic. Imagine it as a traffic controller for data, orchestrating the flow in the most efficient manner possible. To achieve this, HDFS favors DataNodes based on their proximity to the client. In simpler terms, it selects DataNodes that are physically closer to where the data is needed, effectively reducing the need for data to traverse long and congested network routes. By doing so, HDFS significantly optimizes data access speeds and, at the same time, alleviates the network's burden. In addition to its network-savvy approach, HDFS is also resource-

conscious. It carefully evaluates the capabilities of potential DataNodes in terms of the resources required to handle data storage tasks. These resources encompass factors such as available disk space and processing power. By ensuring that selected DataNodes possess the necessary resources, HDFS guarantees the efficient execution of data storage processes without encountering capacity-related bottlenecks. However, it's crucial to acknowledge that HDFS doesn't consistently pursue the absolute optimal DataNode for every task. This approach, while pragmatic, can be seen as a trade-off between efficiency and resource maximization. By not obsessively seeking the absolute best DataNode for each task, HDFS may occasionally fall short of squeezing every last drop of resource efficiency from the system.

4. METHOD AND SYSTEM DESIGN

In contrast to the conventional HDFS, which primarily relies on proximity and resource availability for task assignment, our research presents a novel approach that seeks to enhance the task allocation process. HDFS simply selects machines that are geographically close to the client and possess adequate resources for task execution, our system deploys a more sophisticated methodology. Our proposed system, in contrast, goes beyond mere proximity considerations. It employs an intricate selection mechanism that prioritizes machines with the most optimal network route to the client. It is important to note that geographical proximity alone does not guarantee the shortest data transmission route. Therefore, our system takes into account not just the physical closeness of machines but also their network efficiency in reaching the client. Furthermore, our system doesn't stop at network optimization, it also incorporates a performance-oriented component. After identifying machines with the most efficient network routes, it further refines its selection by prioritizing those machines that exhibit the highest computational performance capabilities. This dual-stage selection process ensures that tasks are allocated to machines that not only minimize data transmission overhead but also maximize computational efficiency, and enhance overall system performance.

Within our research framework, we have implemented a comprehensive monitoring system to assess the performance of each DataNode within the distributed architecture. This monitoring mechanism involves the deployment of an individual agent on every DataNode. These agents continuously track various performance metrics, including task completion times, read/write speeds, central processing unit (CPU) processing speeds, core count, random-access memory (RAM) capacity, RAM speed, CPU utilization, and available free RAM. Subsequently, this agent transmits this critical performance data to a dedicated agent situated on the master node, adhering to a predefined regular reporting schedule.

Figure 1 shows a description of the agent tasks used in our system. The master node, being the central orchestrator of our distributed system, features a dual-agent infrastructure to evaluate the performance of the DataNodes and the network. The first agent assumes the responsibility of collecting performance reports generated by the agents residing on the DataNodes. Upon receiving this wealth of data, the master node agent employs a scoring mechanism to assign a performance score to each DataNode. This scoring system enables the master node to classify and categorize the DataNodes based on their performance metrics, ensuring that resources are allocated optimally within the system. In parallel, at the same time, the second agent located on the master node concentrates on network performance evaluation and is responsible for conducting exhaustive tests on the network to assess various parameters such as latency, throughput and reliability of connections between the client and the various DataNodes in the cluster. To do this, the network agent sends test requests to each DataNode in the cluster, measuring response latency, data transfer speed and connection stability. These tests enable the agent to gather valuable data on the network performance of each node. Then, based on the results of these network assessments, the agent classifies the DataNodes according to their respective network performance. This classification of DataNodes according to their network performance enables the master node to make informed decisions about data routing and the selection of nodes to respond to customer requests. Thanks to this crucial network performance evaluation step, the HDFS system can optimize data transmission and deliver reliable, efficient user performance.

This robust monitoring and classification system, encompassing both individual DataNode performance and network efficiency, contributes significantly to the intelligent resource allocation and task distribution within our distributed system, ultimately enhancing its overall performance and efficiency. When a client seeks access to a file in the system, such as in the case of file retrieval, the master node undertakes a comprehensive process to fulfill this client request, as schematized in Figure 2. This process involves identifying the most efficient machines among the selected DataNodes. This critical step is designed to ensure that the most efficient resources are used to meet the client's request. To achieve this, the master node identifies the DataNodes hosting the required file blocks and uses the information presented by the two evaluation agents to predict the performance of each selected DataNode and the most efficient network connectivity for the client's needs. This meticulous evaluation determines which machines can handle the client's request as quickly and reliably as possible. Once these machines have been identified, the client receives essential information such as the addresses of these selected machines and the identifiers of the

corresponding file blocks. These details enable the customer to locate and access the required data seamlessly while ensuring optimum efficiency and maximum network performance. In this way, they ensure that the client can access the desired file smoothly and efficiently, contributing to overall system optimization and user satisfaction.

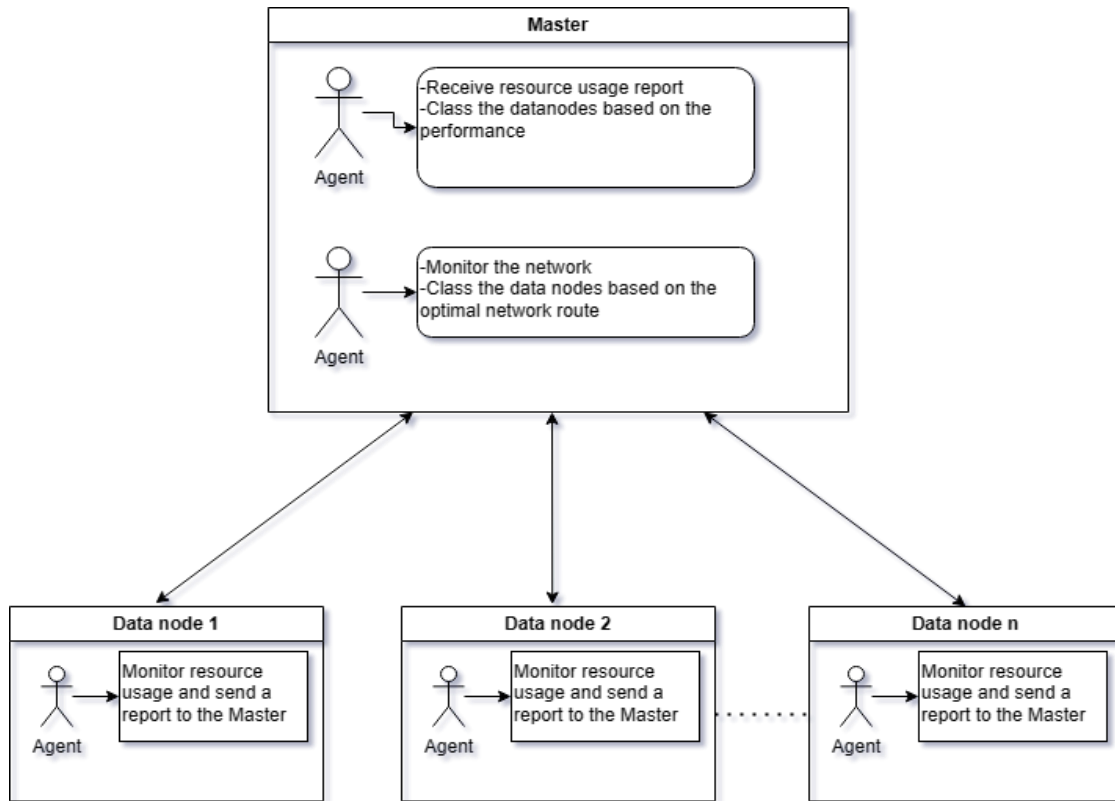


Figure 1. Descriptive of agents tasks

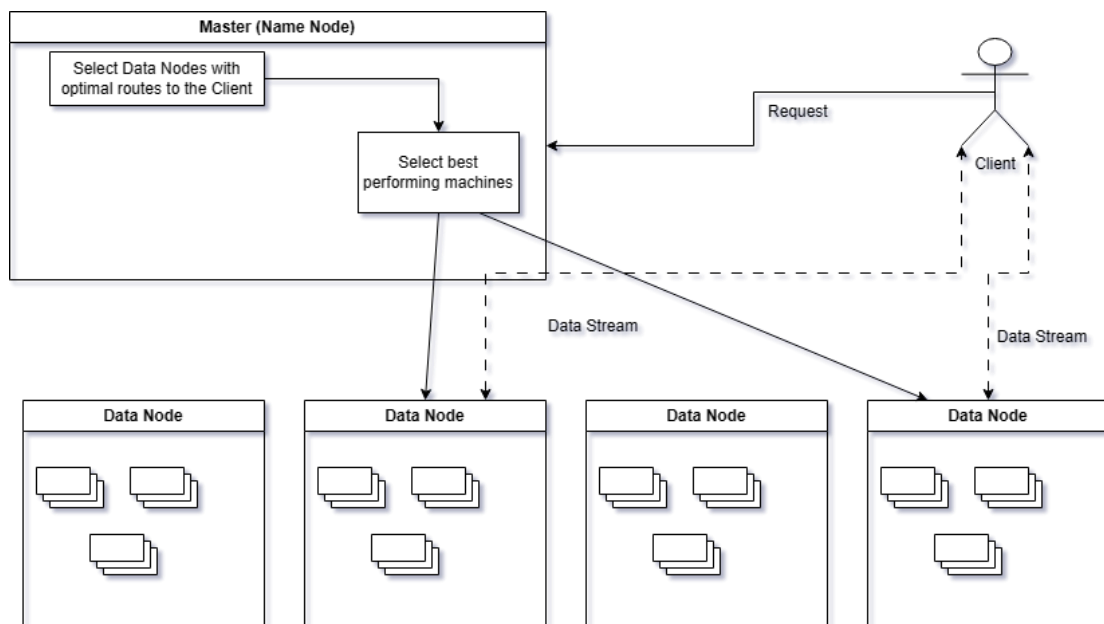


Figure 2. Procedure of DataNodes selection

In our study, we set up an experimental environment consisting of a NameNode, two DataNodes and a client. The NameNode is the central coordination point, while the two DataNodes store and manage the data blocks. The client acts as an external entity requesting access to files stored in the system. Each component of this environment is specified in hardware terms, with details such as the number of processors, the amount of RAM, and other essential features as shown in Table 1. This diverse configuration has been carefully crafted to reflect realistic conditions and enable an assessment of our system’s performance. The specific details of each component will be essential for interpreting the results of our experiments and understanding the impact of different configurations on the performance of the distributed system we are evaluating.

As part of our comparative evaluation between Hadoop HDFS and our system, we conducted tests using files of various sizes, ranging from 20 MB to 1 GB. The aim of these tests was to measure and compare the times required to save and retrieve these files in the two systems, in order to quantify and put into perspective their respective performances. The screenshots presented Figures 3 and 4 illustrate our test results in detail, highlighting the saving and retrieval times for each file size when using our proposed system.

Table 1. Experimental environment

Type of node	No of servers	Configuration
NameNode	1	Intel® Core™ i5-6300U CPU @ 2.40 GHz × 2,4 GB RAM system type: 64-bit operating system, OS Ubuntu 22.04.2
DataNode	2	Intel® Core™ i5-6300U CPU @ 2.40 GHz × 2,4 GB RAM system type: 64-bit operating system, OS Ubuntu 22.04.2
DataNode	1	Intel® Core™ i5-6300U CPU @ 2.40 GHz × 1,3 GB RAM system type: 64-bit operating system, OS Ubuntu 22.04.2
Client	1	Intel® Core™ i5-6300U CPU @ 2.40 GHz × 2,4 GB RAM system type: 64-bit operating system, OS Ubuntu 22.04.2

```
bigdata@master:~/pfe$ ./put.sh ./files/20mbfile.txt
Execution time: 3792 milliseconds
File stored successfully
bigdata@master:~/pfe$ ./put.sh ./files/100mbfile.txt
Execution time: 4155 milliseconds
File stored successfully
bigdata@master:~/pfe$ ./put.sh ./files/200mbfile.txt
Execution time: 6459 milliseconds
File stored successfully
bigdata@master:~/pfe$ ./put.sh ./files/1gbfile.txt
Execution time: 8998 milliseconds
File stored successfully
bigdata@master:~/pfe$
```

Figure 3. Saving a 20 MB/100 MB/200 MB/1 GB file with our system

```
bigdata@master:~/pfe$ ./copyToLocal.sh 20mbfile
Files trasferd successfully.
Execution time: 4770 milliseconds
bigdata@master:~/pfe$ ./copyToLocal.sh 100mbfile
Files trasferd successfully.
Execution time: 5050 milliseconds
bigdata@master:~/pfe$ ./copyToLocal.sh 200mbfile
Files trasferd successfully.
Execution time: 7774 milliseconds
bigdata@master:~/pfe$ ./copyToLocal.sh 1gbfile
Files trasferd successfully.
Execution time: 10426 milliseconds
```

Figure 4. Obtaining a 20 MB/100 MB/200 MB/1 GB file with our system

5. EXPERIMENTAL RESULTS AND DISCUSS

The obtained results from our study on enhancing hadoop distributed storage efficiency using MASs are promising. We conducted measurements to assess the time required for both writing and reading (downloading) complete files. Figures 5 and 6 illustrate the results of the write operation, while Figures 7 and 8 present the results of the read operation.

Figures 5 and 6 illustrate a comparative analysis of our system, both before and following the integration of the MAS. The preceding figure serves as a clear testament to a notable improvement in system performance when operating with a MAS across the entire range of file sizes. Through the application of the percentage optimization formula $[(\text{initial value} - \text{new value}) / \text{initial value}] * 100$, consistently positive values were obtained, signifying enhanced performance in the presence of MAS. The aggregate percentage

optimization, averaging around 26.57% underscores the substantial performance enhancement observed in the context of the MAS scenario.

Similar to Figure 5, Figure 6 demonstrates a consistent superiority of the system when MAS is employed as opposed to its absence. The integration of MAS results in a notable optimization of the reading operation, achieving an approximate improvement of 22%. Significantly, we attained enhanced performance by employing a MAS, primarily attributable to the concurrent execution of tasks. Each agent autonomously undertakes its assigned responsibilities and subsequently consolidates its outcomes for utilization by the central system.

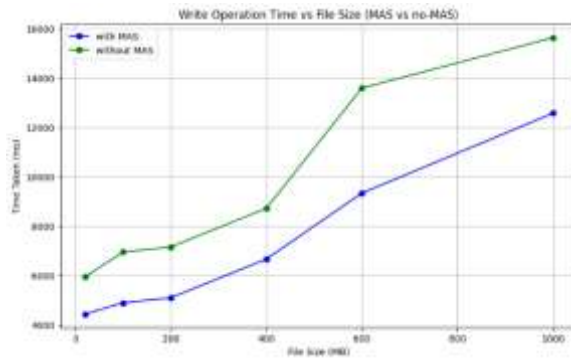


Figure 5. Write operation time (MAS vs no MAS)

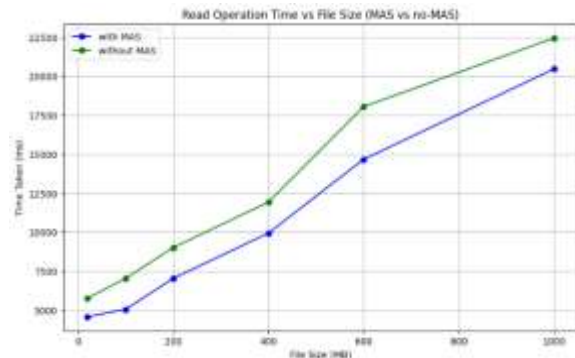


Figure 6. Read operation time (MAS vs no MAS)

As depicted in Figure 7, HDFS exhibits superior performance for small files. However, as the file size increases, our system demonstrates significantly improved performance. This improvement is noteworthy, as it showcases the scalability and efficiency of our system when dealing with larger data volumes. For instance, HDFS took a respectable 2.8 seconds to pen a 20 MB file, but as the challenge scaled to a colossal 1,000 MB file, it required a substantial 16.8 seconds. In striking contrast, our system demonstrated remarkable agility, with a mere 4.4 seconds for the 20 MB file and a lightning-fast 12.3 seconds for the 1,000 MB file. The substantial performance gain as file sizes augment highlights the robustness and adaptability of our system in accommodating the growing demands of big data applications. Turning our attention to Figure 8, we observe a similar trend. HDFS maintains its edge in performance for small files, thanks to its optimized mechanisms for handling such data. Nevertheless, as the file size continues to grow, our system consistently outperforms HDFS, underscoring its reliability and efficiency in handling the challenges posed by larger and more complex files. Notably, HDFS outperforms our system when it comes to handling smaller files. This advantage arises from the fact that smaller files do not necessitate the selection of the best-performing machine; instead, one can simply opt for the nearest available machine.

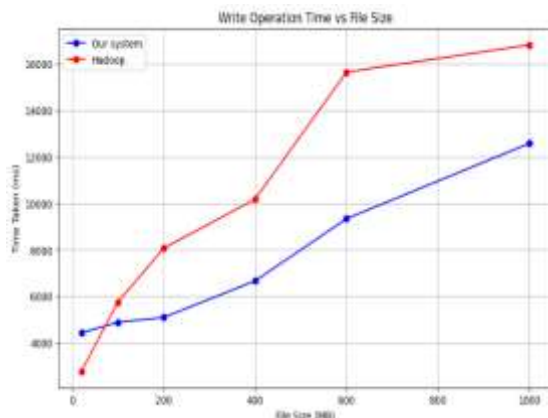


Figure 7. Write operation time (Hadoop vs MAS-HDFS)

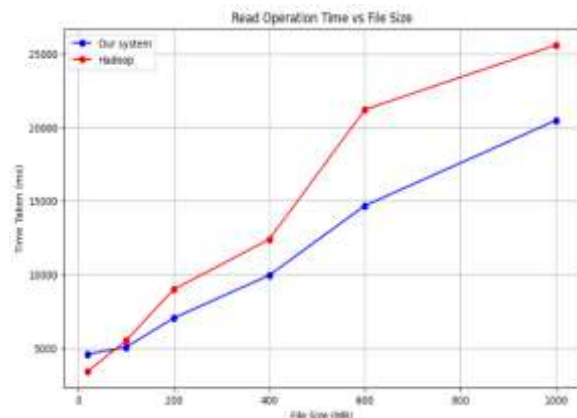


Figure 8. Read operation time (Hadoop vs MAS-HDFS)

6. CONCLUSION

This research paper presents a pioneering MAS that enhances the HDFS and boasts a range of capabilities, including dynamic selection of DataNodes based on network and machine performance, as well as continuous monitoring of DataNode performance. The incorporation of intelligent agents into our MAS-HDFS approach not only facilitates the transparent allocation of storage tasks to nodes, but also adapts to real-time conditions, effectively alleviating performance bottlenecks and reducing data transfer latency. This innovation has the potential to revolutionize the efficiency of distributed storage systems, particularly in the context of data-intensive applications. In addition, we envisage an interesting synergy between our MAS-HDFS and a random read/write enhanced HDFS (REHDFS). By combining our system with REHDFS, which exploits strategies such as random or load-based selection of nodes, we anticipate a substantial improvement in performance. This fusion of intelligent agent-driven decision making and advanced node selection strategies promises to further optimize data storage and retrieval, as well as system efficiency in distributed storage environments. Against a backdrop of ever-increasing demand for efficient data management, our research lays the foundations for transformative improvements in distributed storage systems. By maximizing the use of resources and exploiting their capabilities, we are paving the way for improvements in this field.





REFERENCES

- [1] M. Sais, N. Rafalia, R. Mahdaoui, and J. Abouchabaka, "Distributed storage optimization using multi-agent systems in Hadoop," *E3S Web of Conferences*, vol. 412, Aug. 2023, doi: 10.1051/e3sconf/202341201091.
- [2] 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.
- [3] 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.
- [4] P. G. Balaji and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in Multi-Agent Systems and Applications - I*, D. Srinivasan and L. C. Jain, Eds., in *Studies in Computational Intelligence*. Berlin, Heidelberg: Springer, 2010, pp. 1–27, doi: 10.1007/978-3-642-14435-6_1.
- [5] P. P. Nghiem and S. M. Figueira, "Towards efficient resource provisioning in MapReduce," *Journal of Parallel and Distributed Computing*, vol. 95, pp. 29–41, Sep. 2016, doi: 10.1016/j.jpdc.2016.04.001.
- [6] V. Sontakke and D. R. B., "Memory aware optimized Hadoop MapReduce model in cloud computing environment," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 3, Art. no. 3, Sep. 2023, doi: 10.11591/ijai.v12.i3.pp1270-1280.
- [7] T. J. Grant, "A review of multi-agent systems techniques, with application to columbus user support organisation," *Future Generation Computer Systems*, vol. 7, no. 4, pp. 413–437, May 1992, doi: 10.1016/0167-739X(92)90056-H.
- [8] M. Falco and G. Robiolo, "A systematic literature review in multi-agent systems: patterns and trends," in *2019 XLV Latin American Computing Conference (CLEI)*, Sep. 2019, pp. 1–10, doi: 10.1109/CLEI47609.2019.235098.
- [9] N. E. Khalidi, F. Benabbou, and N. Sael, "Distributed parking management architecture based on multi-agent systems," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, Art. no. 4, Dec. 2021, doi: 10.11591/ijai.v10.i4.pp801-809.
- [10] P. Sethia and K. Karlapalem, "A multi-agent simulation framework on small Hadoop cluster," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1120–1127, Oct. 2011, doi: 10.1016/j.engappai.2011.06.009.
- [11] K. Gogineni, P. Wei, T. Lan, and G. Venkataramani, "Towards efficient multi-agent learning systems," arXiv, May 23, 2023. Accessed: Nov. 29, 2023. [Online]. Available: <http://arxiv.org/abs/2305.13411>
- [12] V. Julian and V. Botti, "Special issue on multi-agent systems," *Applied Sciences*, vol. 13, no. 2, Art. no. 2, Jan. 2023, doi: 10.3390/app13021021.
- [13] K.-C. Jim and C. L. Giles, "How communication can improve the performance of multi-agent systems," in *Proceedings of the fifth international conference on Autonomous agents*, in AGENTS '01. New York, NY, USA: Association for Computing Machinery, May 2001, pp. 584–591, doi: 10.1145/375735.376455.
- [14] M. Oprea, "Applications of multi-agent systems," in *Information Technology*, vol. 157, R. Reis, Ed., in IFIP International Federation for Information Processing, vol. 157, Boston: Kluwer Academic Publishers, 2004, pp. 239–270, doi: 10.1007/1-4020-8159-6_9.
- [15] G. D. Fatta and G. Fortino, "A customizable multi-agent system for distributed data mining," in *Proceedings of the 2007 ACM symposium on Applied computing*, in SAC '07. New York, NY, USA: Association for Computing Machinery, Mar. 2007, pp. 42–47, doi: 10.1145/1244002.1244012.
- [16] M. Sais, N. Rafalia, and J. Abouchabaka, "Enhancements and an intelligent approach to optimize big data storage and management: random enhanced HDFS (REHDFS) and DNA storage," vol. 14, no. 1, pp. 196–203, 2022.
- [17] A. Rasooli and D. Down, "An adaptive scheduling algorithm for dynamic heterogeneous Hadoop systems," in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, 2011, pp. 30–44.
- [18] A. Gandomi, M. Reshadi, A. Movaghar, and A. Khademzadeh, "HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework," *Journal Big Data*, vol. 6, no. 1, p. 106, Dec. 2019, doi: 10.1186/s40537-019-0253-9.
- [19] B. Heintz, A. Chandra, and R. K. Sitaraman, "Optimizing MapReduce for highly distributed environments," arXiv, Jul. 30, 2012, doi: 10.48550/arXiv.1207.7055.
- [20] A. A. Abdallat, A. I. Alahmad, D. A. A. Amimi, and J. A. AlWidian, "Hadoop MapReduce job scheduling algorithms survey and use cases," *Modern Applied Science*, vol. 13, no. 7, Art. no. 7, Jun. 2019, doi: 10.5539/mas.v13n7p38.
- [21] O. Beaumont, T. Lambert, L. Marchal, and B. Thomas, "Data-locality aware dynamic schedulers for independent tasks with replicated inputs," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018, pp. 1206–1213, doi: 10.1109/IPDPSW.2018.00187.
- [22] S. Hedayati, N. Maleki, T. Olsson, F. Ahlgren, M. Seyednezhad, and K. Berahmand, "MapReduce scheduling algorithms in Hadoop: a systematic study," *Journal of Cloud Computing*, vol. 12, no. 1, p. 143, Oct. 2023, doi: 10.1186/s13677-023-00520-9.





- [23] S. Ibrahim, T. A. Phuong, and G. Antoniu, "An eye on the elephant in the wild: a performance evaluation of hadoop's schedulers under failures," F. Pop and M. Potop-Butucaru, Eds., in *Lecture Notes in Computer Science*, vol. 9438. Cham: Springer International Publishing, 2015, pp. 141–157, doi: 10.1007/978-3-319-28448-4_11.
- [24] C.-W. Lee, K.-Y. Hsieh, S.-Y. Hsieh, and H.-C. Hsiao, "A dynamic data placement strategy for hadoop in heterogeneous environments," *Big Data Research*, vol. 1, pp. 14–22, Aug. 2014, doi: 10.1016/j.bdr.2014.07.002.
- [25] P. Varalakshmi and S. Subbiah, "Optimized scheduling of multi-user MapReduce jobs in heterogeneous environment," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 27, p. e7316, 2022, doi: 10.1002/cpe.7316.
- [26] P. R. Giri and G. Sharma, "Apache hadoop architecture, applications, and hadoop distributed file system," *Semiconductor Science and Information Devices*, vol. 4, no. 1, Art. no. 1, May 2022, doi: 10.30564/ssid.v4i1.4619.
- [27] M. R. Ghazi and D. Gangodkar, "Hadoop, MapReduce and HDFS: a developers perspective," *Procedia Computer Science*, vol. 48, pp. 45–50, Jan. 2015, doi: 10.1016/j.procs.2015.04.108.
- [28] M. Saroha and A. Sharma, "Big Data and hadoop ecosystem: a review," in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Nov. 2019, pp. 1–5, doi: 10.1109/ICSSIT46314.2019.8987848.
- [29] Q. Chen, "Massive data storage algorithm based on node performance evaluation," in *2018 14th International Conference on Computational Intelligence and Security (CIS)*, Nov. 2018, pp. 408–411, doi: 10.1109/CIS2018.2018.00097.

BIOGRAPHIES OF AUTHORS







Rabie Mahdaoui     was born in 1999 in fez. He received his Master's degree in computer science, big data cloud computing from Ibn Tofail University, Kenitra, Morocco. He is a Ph.D. student in Computer Research Laboratory (LaRI) at Ibn Tofail. His research interests include big data, data storage, cloud computing, distributed computing. He can be contacted at email: rabie.mahdaoui@uit.ac.ma.







Manar Sais     was born in 1996 in fez. She received his Master's degree in computer science, big data cloud computing from Ibn Tofail University, Kenitra, Morocco. She is a Ph.D. student in Computer Research Laboratory (LaRI) at Ibn Tofail. Her research interests include big data, data storage, cloud computing, distributed computing. She can be contacted at email: manar.sais@uit.ac.ma.



Jaafar Abouchabaka     was born in Guersif, Morocco, 1968. He has obtained two doctorates in Computer Sciences applied to mathematics from Mohammed V University, Rabat, Morocco. Currently, he is a Professor at Department of computer Sciences, Ibn Tofail University, Kenitra, Morocco. His research interests are in concurrent and parallel programming, distributed systems, multi agent systems, genetics algorithms, big data and cloud computing. He can be contacted at email: jaafar.abouchabaka@uit.ac.ma.



Najat Rafalia     was born in Kenitra, Morocco, 1968. She has obtained three doctorates in Computer Sciences from Mohammed V University, Rabat, Morocco by collaboration with ENSEEIHT, Toulouse, France, and Ibn Tofail University, Kenitra, Morocco. Currently, she is a Professor at Department of Computer Sciences, Ibn Tofail University, Kenitra, Morocco. Her research interests are in distributed systems, multi-agent systems, concurrent and parallel programming, communication, security, big data, and cloud computing. She can be contacted at email: arafalia@yahoo.com.