Educational Administration: Theory and Practice
2024, 30(5), 13392-13403
ISSN: 2148-2403
https://kuey.net/

Research Article

Improved Job Execution in Hadoop using Task Deduplication Approach

Sachin Arun Thanekar[1*], Ganesh Dagadu Puri[2]

[1*,2]Computer Engineering Dept. Avcoe, Sangamner, Maharashtra-422608
sachin.thanekar@avcoe.org[1] puriganeshengg@gmail.com[2]

Citation: Sachin Arun Thanekar Ganesh Dagadu Puri et al (2024), Improved Job Execution in Hadoop using Task Deduplication Approach , *Educational Administration: Theory and Practice*, *30*(5), 13392-13403
Doi:- 10.53555/kuey.v30i5.5792

ARTICLE INFO

ABSTRACT

These days, nearly all applications are accessible online, and individuals are fascinated by using social media, sensor networks, and public systems. Large data storage systems are therefore required in order to manage and process the large volume of data. Additionally, this data needs to be handled and kept safely. As a result, massively distributed datacenters are built for processing and storing data. In the rapidly evolving big data era, organizations need to be able to handle and analyze massive amounts of data efficiently to learn valuable things. In order to provide better task execution and preserving data integrity, security, allowed data must be used with appropriate authorization. We have suggested a secure metadata-driven strategy that makes use of Hadoop and improves data processing and storage by reducing needless data movement and job execution time.

1. Introduction:

To enhance job execution performance within the Hadoop framework, a systematic approach leveraging metadata can be implemented. Metadata, which encompasses information about data characteristics, storage and processing, can significantly optimize job execution by providing valuable insights to the Hadoop ecosystem. By integrating a robust metadata management system, administrators and developers gain visibility into the underlying data structures, enabling them to make informed decisions on job scheduling, resource allocation and data locality. This metadata-driven approach enhances the efficiency of data processing by minimizing unnecessary data movement and reducing the overall job execution time. Additionally, the system can leverage metadata to dynamically adjust resource allocation based on workload characteristics, ensuring optimal utilization of cluster resources. Through the intelligent utilization of metadata, this system contributes to a more streamlined and efficient execution of jobs within the Hadoop framework, ultimately improving overall performance and accelerating data processing tasks.[1], [2], [3]

To enhance storage efficiency within the Hadoop Distributed File System (HDFS), a dual-pronged approach involving both file-level and block-level deduplication can be implemented. File-level deduplication involves identifying identical files and storing only one instance of each unique file, eliminating redundant copies across the system. This is particularly beneficial for scenarios where multiple users or jobs may generate identical or similar datasets. On the other hand, block-level deduplication focuses on identifying duplicate blocks of data within files, storing a single copy of each unique block and referencing it across multiple files. This approach is effective when there are common data patterns or repeated segments within various files.[4], [5], [6]

By integrating file-level and block-level deduplication mechanisms into the HDFS architecture, storage space can be utilized more effectively. This not only reduces the overall storage footprint by eliminating redundancy but also optimizes data retrieval times since there are fewer physical blocks to read. Additionally, metadata management plays a crucial role in keeping track of the deduplicated files and blocks, ensuring that the system can accurately locate and retrieve the required data when needed. The implementation of deduplication techniques over HDFS thus contributes to improved storage efficiency, reduced storage costs and enhanced overall performance for data-intensive workloads within the Hadoop ecosystem.

Establishing a robust system for user identities and groups is essential to ensure secure and authorized data access as well as job execution within the Hadoop ecosystem. The implementation involves creating a centralized authentication and authorization framework that integrates seamlessly with Hadoop's security mechanisms. Firstly, user identities should be established, typically through integration with an existing authentication system such as Lightweight Directory Access Protocol (LDAP) or Kerberos. This ensures that

users accessing the Hadoop cluster are authenticated, allowing the system to verify their identity before granting access.[7], [8], [9]
Next, the creation of groups becomes crucial for simplifying access control and permissions management. Users can be organized into logical groups based on their roles or responsibilities. This grouping facilitates the assignment of permissions at a group level rather than individually, streamlining the authorization process. For example, groups can be created for data scientists, analysts, administrators, etc.
Once user identities and groups are established, it is essential to define access control lists (ACLs) to assign permissions for a file system and role-based access control (RBAC)[10], [11], [12] to limit system access within Hadoop. This involves specifying which users or groups have permission to access specific data sets or execute particular jobs. Fine-grained control ensures that sensitive data is protected and only authorized individuals or groups can interact with it.[13], [14]

2. Literature Survey

P. R. Gawali et al. presented a comprehensive review of security considerations within the Hadoop ecosystem. The authors delve into various facets of Hadoop security, analyzing its importance in contemporary big data environments. Through an extensive literature survey, the paper examines existing research, methodologies, and technologies employed to enhance the security posture of Hadoop deployments. It explores authentication mechanisms, access control models, encryption techniques, and other relevant security measures designed to protect data integrity, confidentiality, and availability within Hadoop clusters. By synthesizing insights from diverse sources, the paper aims to provide valuable insights into the evolving landscape of Hadoop security, offering researchers and practitioners a deeper understanding of the challenges and solutions pertinent to securing big data infrastructures [1].
K. Zhang et al. presents a detailed exploration of security design considerations specific to the Hadoop framework. Published in 2009, this work serves as a foundational piece in understanding the early stages of security enhancements within the Hadoop ecosystem. The authors discuss key security challenges inherent in distributed computing environments, particularly focusing on aspects such as authentication, authorization, data encryption, and secure communication protocols. Additionally, they propose and evaluate various security mechanisms tailored to address these challenges within the Hadoop framework, aiming to establish a robust security architecture that ensures the confidentiality, integrity, and availability of data processed and stored in Hadoop clusters. This paper serves as a significant reference for researchers, developers, and practitioners seeking to implement and strengthen security measures in their Hadoop deployments, especially during the nascent stages of Hadoop's adoption in enterprise environments [2].
P. Revathy and R. Mukesh provides an in-depth examination of security practices specifically tailored for big data environments. Through a detailed analysis, the authors explore the challenges and complexities associated with securing large-scale data processing and storage infrastructures. They investigate various security measures, protocols, and frameworks aimed at mitigating risks related to data breaches, unauthorized access, and other security threats prevalent in big data ecosystems. Furthermore, the paper offers insights into emerging trends and best practices in big data security, drawing upon contemporary research and real-world implementations. By synthesizing findings from the analysis, the paper aims to provide valuable guidance for practitioners and researchers involved in designing, implementing, and managing secure big data systems [3].
P. Johri et al. presents a novel security framework specifically designed to address the unique challenges of securing big data within the Hadoop ecosystem. The authors propose a comprehensive approach that integrates various security mechanisms, protocols, and best practices tailored for Hadoop-based environments. Their framework encompasses aspects such as authentication, authorization, encryption, key management, and secure data transmission, aiming to ensure the confidentiality, integrity, and availability of data processed and stored in Hadoop clusters. Furthermore, the paper discusses the implementation details and evaluation of the proposed security framework, providing insights into its effectiveness and performance in real-world scenarios. By presenting this framework, the authors contribute to the advancement of security solutions for big data infrastructures, offering valuable guidance to practitioners and researchers striving to enhance the security posture of their Hadoop deployments [4].
V. N. Inukollu et al. provides an in-depth analysis of the security challenges and issues arising from the convergence of big data and cloud computing technologies. The authors explore various security concerns such as data privacy, confidentiality, integrity, and availability in the context of large-scale data processing and storage in cloud environments. They discuss the implications of storing and processing massive amounts of sensitive data in distributed cloud infrastructures, highlighting vulnerabilities and potential attack vectors. Furthermore, the paper examines existing security mechanisms and solutions aimed at mitigating these challenges, including encryption, access control, authentication, and monitoring techniques. By addressing these security issues, the authors aim to contribute to the development of robust security strategies for leveraging big data in cloud computing environments, ensuring the protection of data assets and maintaining trust in cloud-based services [5].
S. Suganya and S. Selvamuthukumaran provides a comprehensive review of security considerations within the Hadoop Distributed File System (HDFS). The authors delve into various aspects of HDFS security, including

authentication, authorization, encryption, and access control mechanisms. Through a detailed analysis, they examine the existing literature, research, and practices related to securing data stored and processed in HDFS, aiming to identify challenges and propose potential solutions. Furthermore, the paper discusses the implications of HDFS security on the broader Hadoop ecosystem and its relevance in contemporary big data environments. By synthesizing insights from the review, the authors contribute to the understanding of HDFS security and provide valuable guidance for researchers, practitioners, and organizations seeking to deploy and manage secure Hadoop-based infrastructures [6].

S. A. Salunkhe and A. B. Rajmane provides an overview of both performance and security aspects within the Hadoop ecosystem. In terms of performance, the authors examine various factors affecting the efficiency and scalability of Hadoop clusters, including data processing speed, resource utilization, fault tolerance, and optimization techniques. They discuss different approaches and technologies used to enhance the performance of Hadoop, such as data partitioning strategies, task scheduling algorithms, and distributed computing frameworks. Regarding security, the survey explores the challenges and solutions related to securing Hadoop deployments. This involve discussions on authentication mechanisms, access control policies, encryption methods, and auditing mechanisms implemented in Hadoop to protect data confidentiality, integrity, and availability. The authors also analyze the impact of security measures on the overall performance of Hadoop clusters and evaluate trade-offs between security and performance considerations. Overall, this survey paper provides insights into the interplay between performance and security aspects in Hadoop environments, offering valuable information for researchers, practitioners, and organizations seeking to deploy and manage Hadoop-based big data infrastructures [7].

P. Adluru et al. explores the various components and capabilities of the Hadoop ecosystem in addressing security and privacy concerns related to big data. The authors delve into the different tools, frameworks, and approaches available within the Hadoop ecosystem that contribute to enhancing security and privacy in big data environments. This includes discussions on Hadoop's built-in security features, such as authentication, authorization, and auditing mechanisms, as well as complementary technologies like Apache Ranger or Apache Sentry for fine-grained access control. Furthermore, the paper examines techniques for ensuring data privacy within Hadoop clusters, such as data encryption, tokenization, and anonymization methods. The authors also discuss emerging trends and best practices in leveraging the Hadoop ecosystem to mitigate security risks and protect sensitive information in large-scale data processing and storage environments. Overall, this paper provides insights into how organizations can leverage the capabilities of the Hadoop ecosystem to address security and privacy challenges associated with big data, offering valuable guidance for researchers, practitioners, and industry professionals working in this domain [8].

W. Rajeh focuses on the security challenges associated with the Hadoop Distributed File System (HDFS) and specifically examines the issue of unauthorized access. Rajeh discusses various security challenges that HDFS faces, including authentication, authorization, encryption, and auditing. The article explores how unauthorized access can occur within HDFS and the potential impact it can have on data integrity and confidentiality. Additionally, Rajeh examined different attack vectors and vulnerabilities that could lead to unauthorized access within HDFS. Furthermore, the article discusses strategies and techniques to mitigate the risk of unauthorized access within HDFS. This involves the implementation of access control mechanisms, encryption protocols, and monitoring tools to detect and prevent unauthorized activities. Rajeh also proposes recommendations for improving the overall security posture of HDFS deployments. Overall, this article provides valuable insights into the security challenges faced by Hadoop Distributed File System and offers recommendations for addressing unauthorized access issues, contributing to the broader understanding of Hadoop security within the research community [9].

S. Osborn et al. focuses on the configuration and implementation of Role-Based Access Control (RBAC) to enforce both mandatory and discretionary access control policies within information systems. They explore the concept of RBAC and its advantages in managing access control in complex computing environments. They discuss how RBAC can be configured to enforce mandatory access control policies, which are typically based on security classifications and labels, as well as discretionary access control policies, which allow users some level of control over access to resources. Furthermore, the paper delves into the technical details of configuring RBAC systems, including the definition of roles, assignment of permissions to roles, and mapping of users to roles. It also discuss how RBAC can be integrated with other security mechanisms to provide a comprehensive access control solution. Overall, this paper provides valuable insights into the design and implementation of RBAC systems for enforcing both mandatory and discretionary access control policies, contributing to the field of information and system security [10].

Y. B. Reddy discusses various access control mechanisms specifically tailored for big data processing environments. Reddy explores the unique challenges and requirements of access control in the context of big data, considering factors such as the massive volume, variety, and velocity of data processed in modern big data systems. The paper discuss traditional access control models, such as role-based access control (RBAC) and attribute-based access control (ABAC), and their applicability to big data environments. Furthermore, Reddy examine how access control mechanisms can be integrated into popular big data processing frameworks like Hadoop and Spark. This involve discussions on access control policies, authentication mechanisms, authorization mechanisms, and auditing mechanisms implemented within these frameworks to ensure data

security and privacy. Overall, this paper provides insights into the design and implementation of access control mechanisms in big data processing, offering valuable guidance for researchers, practitioners, and organizations seeking to manage access to large-scale data processing infrastructures securely [11].

S. Ding et al. introduces a novel access control scheme that utilizes attribute-based access control (ABAC) and blockchain technology to enhance security in the context of the Internet of Things (IoT) environments. The authors address the challenges of access control in IoT systems, where a multitude of devices with varying capabilities and access requirements interact within dynamic and heterogeneous networks. They propose an ABAC scheme that leverages blockchain technology to provide a decentralized and tamper-resistant mechanism for managing access control policies and permissions. The paper discusses the design and implementation of the proposed scheme, detailing how attributes associated with IoT devices and users can be used to enforce access control policies dynamically. This involve the use of smart contracts on the blockchain to automate access control decision-making based on predefined rules and conditions. Furthermore, the authors provide an evaluation of the proposed scheme, assessing its effectiveness in addressing security and scalability concerns in IoT environments. They also discuss the potential implications and benefits of integrating blockchain-based access control mechanisms into IoT deployments. Overall, this paper contributes to the research on securing IoT systems by introducing a novel access control scheme that harnesses the combined capabilities of attribute-based access control and blockchain technology [12].

M. Gupta et al. explores various authorization frameworks tailored for smart and connected ecosystems, including the Internet of Things (IoT) and cyber-physical systems (CPS). They discuss the unique challenges associated with authorization in these complex and dynamic environments, where a multitude of devices, sensors, and actuators interact within interconnected networks. They examine different authorization models, such as role-based access control (RBAC), attribute-based access control (ABAC), and policy-based access control (PBAC), and their applicability to smart and connected ecosystems. Furthermore, delves into the design and implementation of authorization frameworks specifically tailored for IoT and CPS environments. This involve discussions on access control policies, enforcement mechanisms, and the integration of security protocols and standards to ensure the confidentiality, integrity, and availability of data and resources. Additionally, explore emerging technologies and approaches for enhancing authorization in smart and connected ecosystems, such as blockchain-based access control and decentralized identity management. Overall, provides valuable insights into the design and deployment of authorization frameworks for ensuring security and privacy in smart and connected ecosystems, offering guidance for researchers, practitioners, and organizations working in the field of IoT and CPS security [13].

N. Sirisha and K. V.D. Kiran focuses on how to manage and control access to data stored in Hadoop clusters using Apache Sentry. They delve into the importance of data authorization in Hadoop environments, considering the large volumes of data processed and stored in these distributed systems. They discuss the challenges of ensuring data security and privacy in such environments and introduce Apache Sentry as a solution for addressing these challenges. They provides an overview of Apache Sentry, including its features and capabilities for implementing fine-grained access control policies within Hadoop clusters. It discuss how Sentry allows administrators to define and enforce access control rules based on users, groups, roles, and permissions. Furthermore, explain the implementation process of Apache Sentry in a Hadoop environment, detailing the steps required to configure and manage access control policies effectively. They also provide examples or case studies to demonstrate the practical application of Apache Sentry for securing data in Hadoop clusters. Overall, serves as a valuable resource for researchers, practitioners, and organizations interested in enhancing data security and access control in Hadoop environments using Apache Sentry [14].

H. Alshammari et al. presents a novel approach, called H2hadoop, aimed at enhancing the performance of Hadoop systems by leveraging metadata from related jobs. They identify the challenge of optimizing job execution in Hadoop clusters, particularly when multiple jobs share common characteristics or dependencies. They introduce the concept of using metadata, such as job profiles, input data characteristics, and execution history, to inform scheduling and resource allocation decisions. The paper describes the design and implementation of the H2hadoop framework, detailing how it collects, analyzes, and utilizes metadata from past and ongoing jobs to improve the performance of subsequent job executions. This involve techniques such as job similarity analysis, workload prediction, and resource provisioning based on historical patterns. Furthermore, present experimental results or performance evaluations to demonstrate the effectiveness of H2hadoop in optimizing job scheduling, reducing execution times, and enhancing resource utilization in Hadoop clusters. Overall, this paper contributes to the research on performance optimization in Hadoop environments by proposing a metadata-driven approach to job scheduling and resource management [15].

S. A. Thanekar et al. focuses on strategies to enhance the performance of Hadoop clusters by improving the capabilities of the Name Node, a critical component in the Hadoop Distributed File System (HDFS). They address the challenge of Name Node scalability and performance bottlenecks that can arise in large-scale Hadoop deployments. They introduce techniques and optimizations aimed at reducing Name Node overhead, improving its responsiveness, and increasing its ability to handle a higher volume of metadata operations. The describes the proposed enhancements to Name Node capabilities, which include optimizations in metadata management, caching mechanisms, and distributed processing techniques. These enhancements aim to reduce Name Node contention, alleviate storage and processing bottlenecks, and enhance overall cluster performance.

Furthermore, present experimental results or performance evaluations to demonstrate the effectiveness of their approach in improving Hadoop performance. This involve benchmarking against standard Hadoop configurations and assessing metrics such as Name Node throughput, latency, and scalability. Overall, contributes to the research on Hadoop performance optimization by focusing on enhancements to the Name Node component, offering insights and techniques to improve the efficiency and scalability of Hadoop clusters [16].

S. A. Thanekar et al. explores methods to optimize storage space utilization within the Hadoop Distributed File System (HDFS) through the implementation of deduplication techniques. They address the challenge of storage inefficiency caused by redundant data stored across files and blocks within HDFS. They propose and evaluate strategies for deduplicating data at both the file level and the block level to reduce storage overhead and improve overall resource utilization. They describes the implementation and integration of file-level and block-level deduplication mechanisms into HDFS. File-level deduplication involve identifying and eliminating duplicate files or data segments within the file system, while block-level deduplication focus on identifying and removing duplicate blocks of data stored across different files. Furthermore, present experimental results or case studies to demonstrate the effectiveness of their deduplication techniques in reducing storage consumption within HDFS. This involve assessments of storage savings achieved, performance impacts, and considerations for scalability and system overhead. Overall, this paper contributes to the research on storage optimization in Hadoop environments by proposing and evaluating deduplication strategies tailored for HDFS [17].

T. Sachin Arun et al. explores methods to improve job execution efficiency in Hadoop by leveraging authorized deduplicated data. They address the challenge of optimizing job execution in Hadoop environments while ensuring data security and minimizing storage overhead. They propose a framework that combines authorization mechanisms with deduplication techniques to enable efficient and secure processing of data within Hadoop clusters. They describe the design and implementation of the proposed framework, detailing how authorized users can access deduplicated data within Hadoop while adhering to access control policies and permissions. This involve integration with existing Hadoop components for authentication, authorization, and data management. Furthermore, present experimental results or performance evaluations to demonstrate the effectiveness of their approach in improving job execution efficiency. This involve assessments of job completion times, resource utilization, and storage savings achieved through deduplication. Overall, this article contributes to the research on optimizing job execution in Hadoop environments by introducing a framework that combines authorization and deduplication techniques. It offers insights into how Hadoop clusters can efficiently process data while ensuring data security and minimizing storage overhead [18].

3. Proposed System

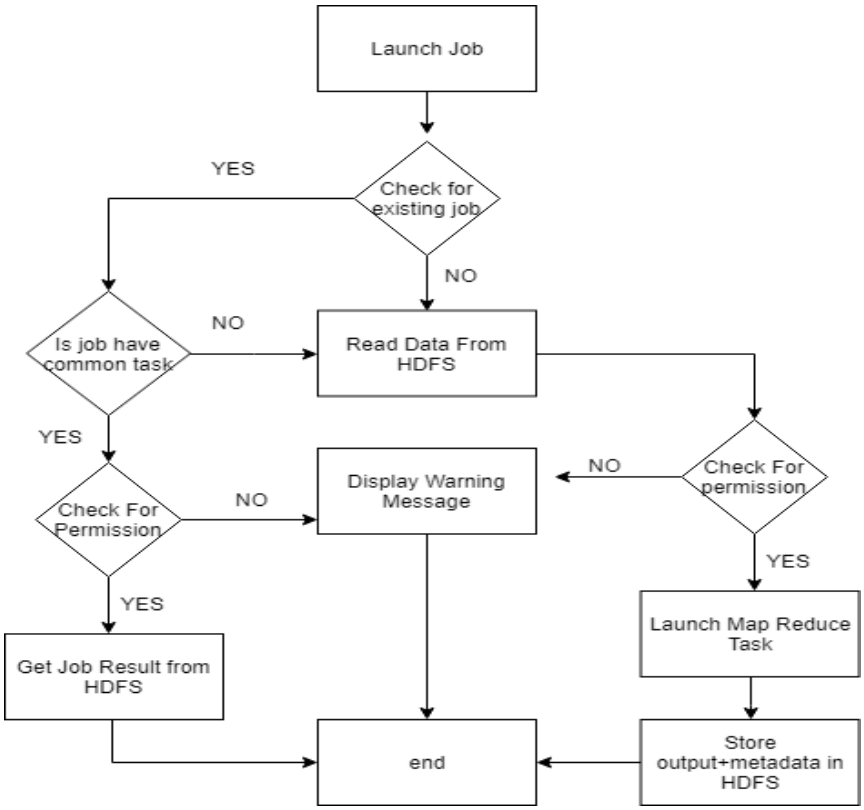

Fig 1: Proposed Task Deduplication Approach

Authorization and Uploading of Data:

While effective task allocation and execution are carried out during job execution, data uploading involves the construction of data authorization and deduplication checking policies. A thorough explanation of these features can be found in the section that follows.

Name Node keeps a lookup table where it stores information about jobs that have already been completed. The Common Jobs Block Table (CJBT)[15] is the name of this table. The jobs and data blocks related information of these jobs are stored in this CJBT. CJBT has a dynamic nature.

When users input data into the prefilter to save it in the storage system, the file filter checks the data to see if it is a duplicate of an already-existing file from the metadata table. If the data is redundant, it will be eliminated from the file filter. If not, the data will be split up into many blocks and the file's hash value is updated in the metadata table simultaneously. Ultimately, the data blocks are sent to the data center.

The metadata server, postfilter, and data center come after the prefilter. Data storage is handled by the data center, and the metadata server is where all of the metadata is gathered. The primary function of the postfilter is to cause the same block that has been stocked in the storage area to be wiped away in order to prevent redundant blocks from being maintained there. Nonetheless, block management is the responsibility of the postfilter, which is the second tier of the split HDFS structure. Additionally, we look at the duplicate block using the hash function-calculated fingerprint.

The fingerprint of each block in the postfilter will be determined using the hash function once the data received from the prefilter has been split up into many blocks. Next, we search the metadata server for every hash value to verify if the blocks are duplicates or not. The duplicate block is removed if more than one block has the same hash value. If not, it is sent to the data center, where the blocks are stored in Data Nodes. Afterwards, the duplicate blocks will be produced as HDFS in order to provide data dependability. Lastly, different deduplication selections will be made based on the storage space situation as it is.

The user has the option to define file access rights after uploading the file. Permission to read, write, and execute are among the rights. Users have the option to grant group rights or to a single user with regard to file permissions. The user can work on the specified files based on the permissions that have been allocated to them.[16], [17], [18]

Job Execution w.r.t Work Performance:

The assigned task in the Hadoop framework is carried out by a number of workers known as data nodes. The map reduce work technique is used to complete the task. This processing assignment is assigned to the Hadoop framework's name node. The name node is in charge of giving the data nodes tasks, or jobs. We refer to this procedure as mapping. The data node sends the execution result back to the main name node after receiving the task and processing the data. Once all of the worker results have been combined, the main node creates a copy of the final result. We call this method the reduce process. Using the map reduce technique on the Hadoop framework, the repeating, same job allocation requires the same amount of memory and execution time as previously. The processing history is not preserved by the name node or data node. A novel solution is suggested in order to get around this drawback of the current Hadoop framework. The suggested approach is an addition to the current Hadoop framework map reduce workflow. This extension improves the time and memory efficiency of the system's work execution [19], [20], [21].

Efficient task allocation and execution are carried out in job execution. Using CJBT, a two-fold modified dynamic deduplication is created. It operates in two stages. In the first part, the entire file is taken into account, and a hash value is generated and compared. In second part, the file is split up into blocks, and hash values are created and verified for each block.[16], [17], [18]

Deduplication filtering process:

The input of the algorithm is a text file. The HDFS will receive this text file and put it in a database. This algorithm produces no output other than the data being split up into blocks or, in the event that the same file already exists in the database, producing no output at all. The metadata table will store information from various blocks. The file's metadata property will be checked first when it is submitted for storage in a database. There won't be any processing necessary if these properties match the metadata table; instead, the process will skip ahead and return to the outside process. The input data will be divided into many blocks and the attributes of the new data will be kept in a metadata table in case an attribute is not present in the meta data. When Name Node notifies the data node to store the data, the process will be finished.[17]

Algorithm: Two-fold modified dynamic deduplication with CJBT
Part 1: Deduplication Check for File as a whole
3 Text file input
4 Blocks of data separated into output
or, in the event that a match is found, no output
5 Block details are stored in a metadata table.
6 Procedure

Once the file has been submitted,
review its properties.
If these properties correspond to the metadata table,
there is no need to proceed;
go with the skippable step.
If not, divide the data into blocks.
Add new file attributes to the metadata table;
instruct the data node where to store the data;
and end

Part2: Deduplication Check for blocks after dividing file into blocks

The next step in preventing duplication is block level duplication checks. We checked for file duplication in the first section; however, in this section, we are checking for duplication for the nodes that are formed after the file data is divided into blocks.

- Text file separated into blocks with their respective details entered as an input
- Blocks of data separated into output or, in the event that a match is found, no output
- Procedure

Block hash values are stored in a metadata database along with block details.
When the file level computation output is received,
the hash function computes it;
if a match is discovered, the operation is skipped.
If not, tell Data Node to store additional blocks.
Finish

3.3 User Rights Specification:

The user has the option to define file access rights after uploading the file. Permission to read, write, and execute are among the rights. Users have the option to grant group rights or to a single user with regard to file permissions. The user can work on the specified files based on the permissions that have been allocated to them.
The Access Rights Specific Common Job Blocks (ARS-CJB) table structure is maintained by the system in order to carry out this operation. It consists of:

1. Title of job
2. User;
3. Access specification;
4. Name of the data node
5. Details of the dataset
6. Location of the result

In order to begin deduplication checking, name node uses the data found in this table structure. The name node only gathers the results from the corresponding data nodes if deduplication is detected.

De-duplicating tasks and comparing data similarities:

The data node stores the history of job and data execution while carrying out the assigned task over data. The name node determines whether or not data has already been processed using the same job. If so, it retrieves the data node's findings without repeating the previous task's execution by looking up the data node's history. Task de-duplication is the process of avoiding redundant effort in result generation.
Here, the following characteristics are found to be common across various jobs:
Block id,
Job name,
User,
Name of Data Node,
of Dataset.

The name node determines whether or not data has already been processed with the same job. If so, it checks the data node history and obtains the results from the data node without re-executing the same task. The similar blocks that are common to the two different tasks will not be processed again, resulting in reprocessing of similar blocks being reduced. Task de-duplication avoids the duplication work of result generation.

4. Results and Discussion:

Nomenclature used for results:

- **FS:** File Size

- **BS:** Block Size
- **TGT:** Tag Generation Time
- **FDT:** File duplication check time
- **BDT:** Block duplication check time
- **UT:** Upload Time
- **TPT:** Total Processing Time
- **DP:** Duplication Percentage
- **JDT:** Job Duplication Check Time
- **JT:** Job Execution Time

Table 1 Checking Deduplication at Job Level(Keeping Block Size 100 KB)

FS (MB)	JDT (Seconds)	JT (Seconds)	TPT (Seconds)
1	0.067	17.712	17.867
2	0.014	29.474	29.835
3	0.015	41.131	41.222
4	0.01	56.047	56.141

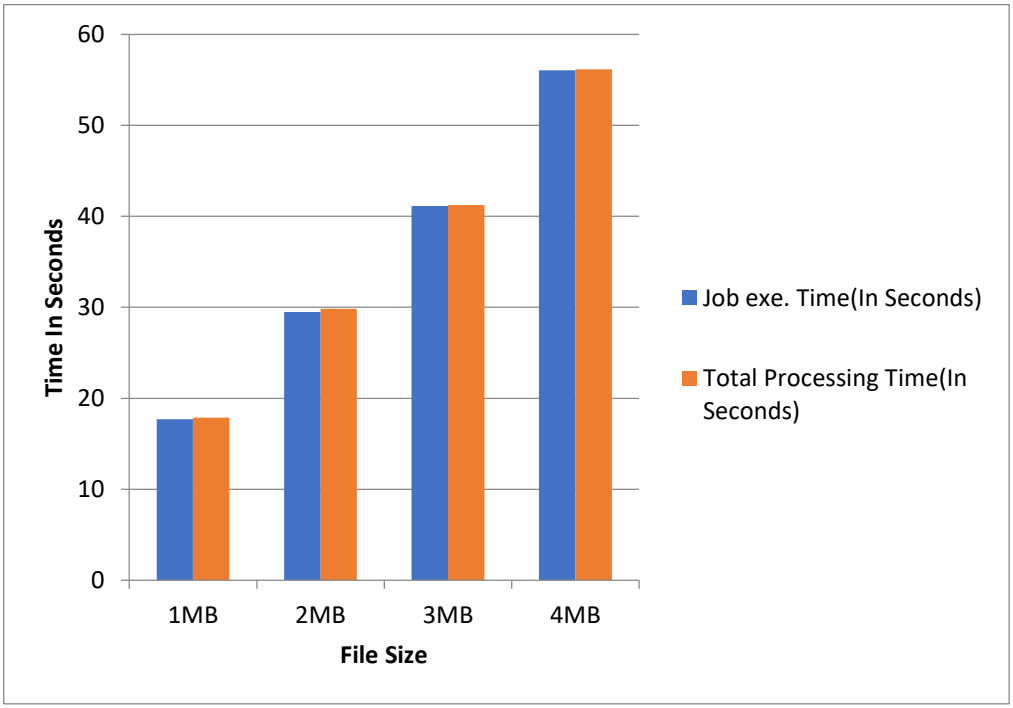

Table 2 Checking Deduplication at Job Level(Keeping Block Size 50 MB**)**

FS (MB)	JDT (Seconds)	JT (Seconds)	TPT (Seconds)
100	0.019	1788.6	1807.92
200	0.016	2993.46	3002.58
300	0.014	4160.4	4198.2
400	0.017	5604.6	5641.8

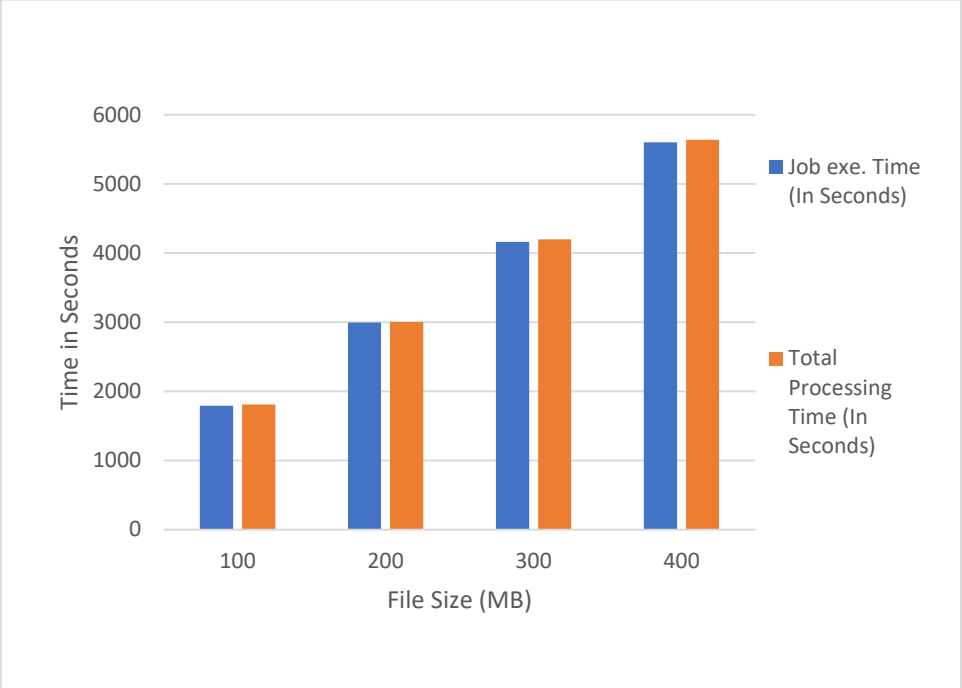

Fig: 3 Checking Deduplication at Job Level(Keeping Block Size 50 MB)

It is confirmed by tables 6, 7, and figures 9–10 that the number of blocks grows along with file size, which in turn increases job execution time. Regardless of file size, the job deduplication time is always the same.

Table 3 Checking Deduplication at Job Level(Keeping Block Size 100 MB)

DP	JDT (Seconds)	JT (Seconds)	TPT (Seconds)
0%	0.010	56.047	56.141
25%	0.013	38.63	38.648
50%	0.038	25.958	26.006
75%	0.035	14.834	14.88
100%	0.057	0	0.07

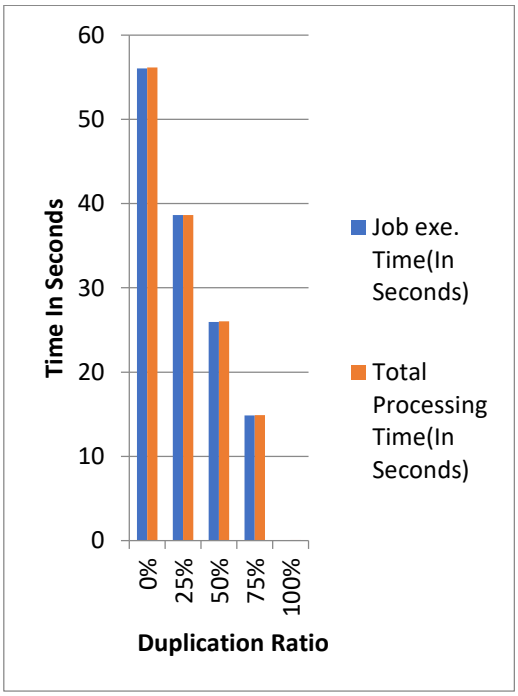

Fig: 4 Checking Deduplication at Job Level(Keeping Block Size 100 MB)

Table 4 Checking Deduplication at Job Level(Keeping File Size 100 MB)

DP	JDT (Seconds)	JT (Seconds)	TPT (Seconds)
0%	0.017	5604.6	5641.8
25%	0.013	4168.2	4195.2
50%	0.038	3001.26	3014.16
75%	0.035	1780.26	1795.86
100%	0.057	0	3.78

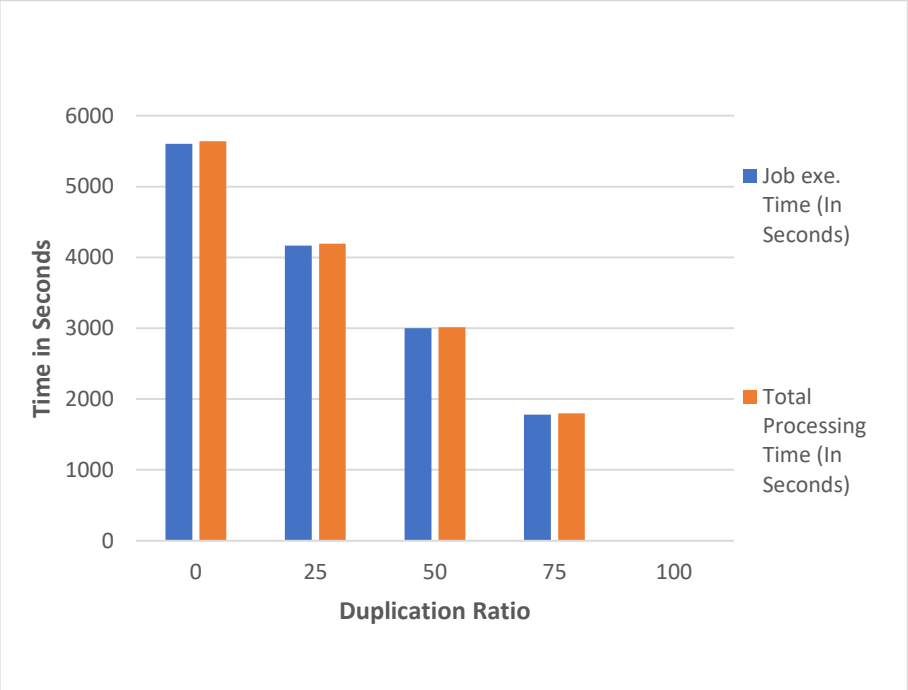

Fig: 5 Checking Deduplication at Job Level(Keeping File Size 100 MB)

It is evident from tables 8 and 11 as well as figure 9 and 12 that when the duplication ratio rises, job execution time falls. The right block size and duplication ratio must be carefully chosen.

Table 5 Checking Deduplication with Variable Block Size at Job Level(Keeping File Size 4 MB**)**

BS (KB)	JDT (Seconds)	JT (Seconds)	TPT (Seconds)
50	55	106873	106935
100	10	56047	56141
150	57	36498	36564
200	128	31149	31600

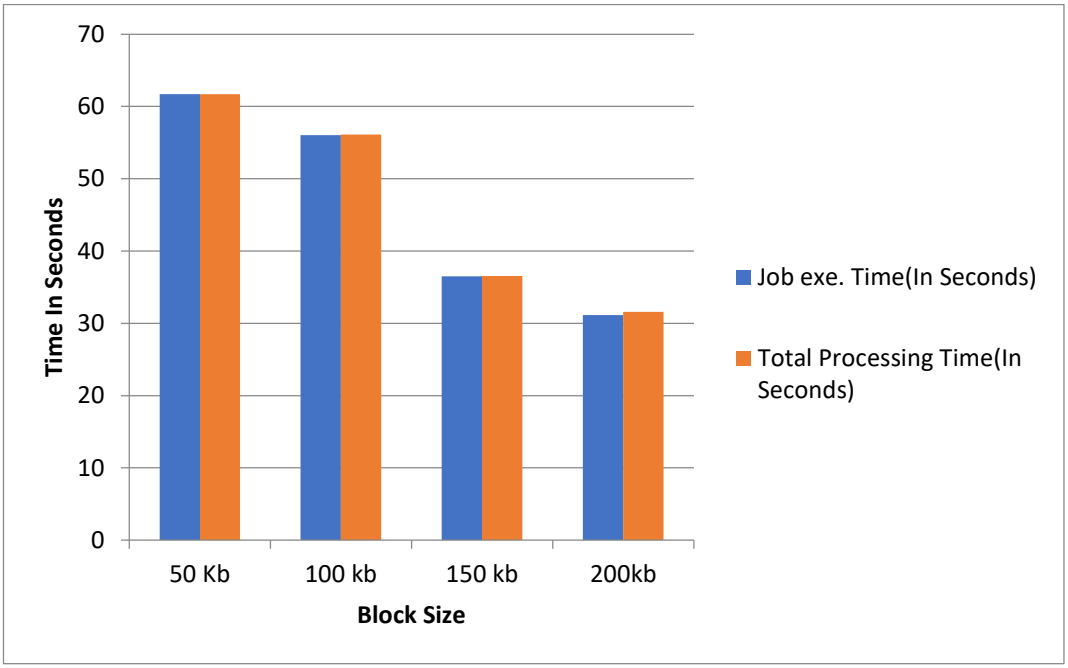

Fig: 6 Checking Deduplication with Variable Block Size at Job Level(Keeping File Size 4 MB)

It is confirmed by table 10 and figure 13 that processing time lowers as block size grows because fewer blocks are processed. Selecting a moderate block size is crucial.

4. Conclusion and Future Scope

The contributions of this study extend across algorithmic advancements, dynamic strategies, storage system integration, resource optimization, fault tolerance, compression-deduplication synergy, energy efficiency, real-time deduplication feasibility, security measures, and user experience design. These contributions collectively advance the understanding and practical implementation of deduplication in Hadoop environments, offering valuable insights for both researchers and practitioners in the field of big data processing.
The recommendations for future research highlight the importance of tailored algorithms, dynamic strategies, integration with diverse storage systems, resource optimization, fault tolerance, compression-deduplication synergy, energy efficiency, real time deduplication, security, privacy, and user experience. Addressing these facets collectively will contribute to the ongoing evolution of Hadoop systems, ensuring their adaptability to the ever-growing demands of big data processing while maintaining a focus on efficiency, reliability, and security.

5. References

1. P. R. Gawali, R. Talmale, R. Babu, and M. Tech, "Hadoop Security-A Review," *International Journal of Innovative Research in Computer and Communication Engineering (An ISO*, vol. 3297, 2007, doi: 10.15680/IJIRCCE.2015.0310017.
2. K. Zhang, S. Radia, R. Marti, and C. Harrell Yahoo, "Hadoop Security Design," 2009.
3. P. Revathy and R. Mukesh, "Analysis of big data security practices," in *Proceedings of the 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2017*, 2018. doi: 10.1109/ICATCCT.2017.8389145.
4. P. Johri, A. Kumar, S. Das, and S. Arora, "Security framework using Hadoop for big data," in *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, 2017. doi: 10.1109/CCAA.2017.8229813.
5. V. N. Inukollu, S. Arsi, and S. Rao Ravuri, "Security Issues Associated with Big Data in Cloud Computing," *International Journal of Network Security & Its Applications*, vol. 6, no. 3, pp. 45–56, May 2014, doi: 10.5121/ijnsa.2014.6304.
6. S. Suganya and S. Selvamuthukumaran, "Hadoop Distributed File System Security -A Review," in *Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018*, 2018. doi: 10.1109/ICCTCT.2018.8550957.
7. S. A. Salunkhe and A. B. Rajmane, "A Survey on Performance and Security of Hadoop," 2015. [Online]. Available: www.ijsr.net
8. P. Adluru, S. S. Datla, and X. Zhang, "Hadoop eco system for big data security and privacy," in *2015 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2015*, 2015. doi:

10.1109/LISAT.2015.7160211.
9. W. Rajeh, “Hadoop Distributed File System Security Challenges and Examination of Unauthorized Access Issue,” *Journal of Information Security*, vol. 13, no. 02, 2022, doi: 10.4236/jis.2022.132002.
10. [S. Osborn, R. Sandhu, and Q. Munawer, “Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies,” *ACM Transactions on Information and System Security*, vol. 3, no. 2, 2000, doi: 10.1145/354876.354878.
11. [Y. B. Reddy, “Access control mechanisms in big data processing,” in *Proceedings of the IASTED International Symposium on Software Engineering and Applications, SEA 2015*, 2015. doi: 10.2316/P.2015.829-006.
12. S. Ding, J. Cao, C. Li, K. Fan, and H. Li, “A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2905846.
13. M. Gupta, S. Bhatt, A. H. Alshehri, and R. Sandhu, “Authorization Frameworks for Smart and Connected Ecosystems,” in *Access Control Models and Architectures For IoT and Cyber Physical Systems*, 2022. doi: 10.1007/978-3-030-81089-4_3.
14. [N. Sirisha and K. V.D. Kiran, “Authorization of data in Hadoop using Apache Sentry,” *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 3.6 Special Issue 6, 2018, doi: 10.14419/ijet.v7i3.6.14978.
15. H. Alshammari, J. Lee, and H. Bajwa, “H2hadoop: Improving hadoop performance using the metadata of related jobs,” *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, 2018, doi: 10.1109/TCC.2016.2535261.
16. S. A. Thanekar, A. Bagwan, and K. Subrahmanyam, “Improving Hadoop performance by enhancing name node capabilities,” *Fronteiras*, vol. 6, no. 2, 2017, doi: 10.21664/2238-8869.2017v6i2.p1-8.
17. S. A. Thanekar, K. Subrahmanyam, and A. Bagwan, “Effective utilization of storage space by applying file level and block-level deduplication over HDFS,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 6, 2019.
18. T. Sachin Arun, K. Subrahmanyam, and A. B. Bagwan, “Effective Job Execution in Hadoop Over Authorized Deduplicated Data,” *Webology*, vol. 17, no. 2, 2020, doi: 10.14704/WEB/V17I2/WEB17043.
19. G. D. Puri and D. Haritha, “Survey big data analytics, applications and privacy concerns,” Indian J. Sci. Technol., vol. 9, no. 17, 2016, doi: 10.17485/ijst/2016/v9i17/93028
20. G. D. Puri and D. Haritha, “Improving Privacy Preservation Approach for Healthcare Data using Frequency Distribution of Delicate Information,” Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 9, pp. 82–90, 2022 .
21. G. D. Puri and D. Haritha, “Implementation of Big Data Privacy Preservation Technique for Electronic Health Records in Multivendor Environment,” Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 2, 2023, doi: 10.14569/IJACSA.2023.0140214 .