Comparative Analysis of Distributed File Systems: A Focus on Hadoop Distributed File System (HDFS)

Priya Shah, Dhruv Patel, Meet Shah, and Kenil vaghasiya

Computer Science and Engineering Department

California State University Long Beach

Long Beach,CA-90840,USA

ABSTRACT

In the era of burgeoning data volumes, the imperative for efficient and scalable storage solutions has become increasingly crucial. This research paper conducts a meticulous examination of the Hadoop Distributed File System (HDFS), with a specific emphasis on its security measures and load balancing algorithms. The study navigates into the foundational principles that govern distributed file systems, elucidating their architecture, data distribution mechanisms, and the strategies employed for ensuring data security and load distribution.

The research employs a comparative analysis framework to evaluate the robustness of HDFS in terms of security and load balancing. Through an amalgamation of extensive literature review the paper aims to shed light on the unique features that position HDFS as a leading choice in the domain of distributed storage, emphasizing considerations of data reliability, scalability, fault tolerance, security, and adaptability to diverse workloads.

Special attention is devoted to investigating the security protocols implemented within HDFS, examining encryption techniques, access controls, and authentication mechanisms. Additionally, the study explores the load balancing algorithms employed by HDFS to ensure equitable distribution of data across the distributed cluster, optimizing resource utilization and performance.

Real-world use cases and applications where HDFS excels in terms of security and load balancing are explored, providing insights into its practical implementations across different industries. The findings of this research contribute to a nuanced understanding of HDFS, offering valuable insights for system architects, data engineers, and researchers seeking optimal solutions for managing large-scale data in distributed environments, with a heightened focus on security and load balancing considerations.

Keywords: Distributed file systems, Hadoop Distributed File System (HDFS), data storage, scalability, fault tolerance, security, load balancing, comparative analysis, big data, distributed computing

1. INTRODUCTION

In the era of unprecedented data growth, the demand for robust and scalable storage solutions has become a critical concern for organizations across various domains. Distributed file systems, designed to manage large volumes of data across multiple nodes, have emerged as key components in addressing these challenges. This research endeavors to explore the intricacies of distributed file systems, focusing particularly on the Hadoop Distributed File System (HDFS), and aims to provide a comprehensive understanding of their architecture, functionality, and practical applications.

1.1 Background:

The escalating volume of data generated in diverse fields such as scientific research, business analytics, and internet applications necessitates storage systems capable of handling massive datasets efficiently.

1.2 Motivation:

The motivation behind this research lies in the need to evaluate and compare distributed file systems to ascertain their suitability for contemporary data-intensive applications. The focus on Hadoop Distributed File System (HDFS) stems from its widespread adoption in the realm of big data processing and analytics.

1.3 Objectives:

• Examine the foundational principles that underpin distributed file systems. • Investigate the architecture and design choices of prominent distributed file systems. • Assess the scalability, fault tolerance, and performance characteristics of distributed file systems. • Specifically analyze the Hadoop Distributed File System (HDFS) in comparison to other distributed file systems.

1.4 Scope and Significance:

This research contributes to the existing body of knowledge by providing a comprehensive analysis of distributed file systems, aiding practitioners, system architects, and researchers in making informed decisions about storage solutions in dis-

tributed computing environments. In navigating through the complexities of distributed file systems, this research sets the stage for a deeper exploration into their role in the evolving landscape of data storage and processing. As we delve into the subsequent sections, the paper will unfold with an in-depth literature review, methodological approach, comparative analysis, and practical implications, culminating in a nuanced understanding of distributed file systems and their specific manifestation in the form of Hadoop Distributed File System (HDFS).

2. RELATED WORK

In the contemporary landscape of computing, the role of distributed file systems (DFS) has evolved into an indispensable component, addressing a myriad of use cases within traditional, cloud-based, and hybrid environments. Foundational structures such as the Network File System (NFS) and Andrew File System (AFS) have long been pivotal in establishing distributed storage frameworks, seamlessly facilitating file access across interconnected computers. These solutions find widespread application in contexts where data sharing and collaborative endeavors are of paramount importance, notably observed in academic institutions, research laboratories, and enterprises.

The advent of cloud computing has heralded a transformative shift in data storage and management paradigms. Cloud-based distributed file systems, exemplified by stalwarts like Amazon S3, Google Cloud Storage, and Microsoft Azure Blob Storage, have risen to prominence due to their scalability, flexibility, and accessibility. Organizations strategically leverage these cloud-based DFS solutions to house extensive datasets, catalyze data analytics, and ensure high availability for teams dispersed across geographical locations.

Within this expansive spectrum, the Hadoop Distributed File System (HDFS) occupies a distinct niche. Tailored explicitly for big data processing, HDFS stands out for its ability to handle large-scale data across clusters of commodity hardware. While it shares commonalities with cloud-based DFS paradigms in terms of scalability and fault tolerance, HDFS is uniquely optimized to meet the specific demands of the Hadoop ecosystem. Its prowess shines in scenarios demanding parallel processing of massive datasets, rendering it foundational for big data applications and analytics platforms. The versatility of HDFS is evident across diverse industries, including finance, healthcare, and e-commerce, where the efficient storage and processing of vast datasets are pivotal for deriving actionable insights. As organizations increasingly pivot towards data-driven decision-making, HDFS remains a linchpin in facilitating the storage and analysis of extensive volumes of both structured and unstructured data.

2.1 HDFS

2.1.1 Apache Hadoop :-

Apache Hadoop stands as a pioneering framework in the realm of distributed computing, offering a comprehensive ecosystem designed to address the challenges posed by the processing and analysis of vast datasets. At its core, Hadoop embraces a modular architecture that includes the Hadoop Distributed File System (HDFS) for scalable and fault-tolerant storage, and the MapReduce programming model for distributed processing. This open-source framework, managed by the Apache Software Foundation, has become synonymous with big data analytics due to its ability to distribute and parallelize computations across clusters of commodity hardware. Hadoop's significance lies not only in its capacity to handle massive datasets but also in its flexibility to accommodate a diverse range of data types, making it a go-to solution for organizations grappling with the complexities of unstructured and structured data. As the digital landscape continues to evolve, Apache Hadoop remains a cornerstone for enterprises seeking robust solutions to harness the power of distributed computing for large-scale data processing, storage, and analytics.

2.1.2 Hadoop Distributed File System (HDFS) Architecture:

The architecture of Hadoop Distributed File System (HDFS) is designed to provide a robust and scalable solution for handling large-scale data across distributed clusters. At its core, HDFS employs a master-slave architecture, comprising two primary components: the NameNode and DataNodes. The NameNode serves as the central metadata repository, storing information about the file system namespace, file metadata, and the block locations. On the other hand, DataNodes store the actual data blocks and report back to the NameNode periodically with block information. This separation of metadata and data storage facilitates scalability and fault tolerance. Furthermore, HDFS divides large files into fixed-size blocks, typically 128 MB or 256 MB, which are then distributed across DataNodes. This distributed storage model enables parallel processing of data across the cluster, a key factor in supporting Hadoop's parallelized data processing paradigm. Features of Hadoop Distributed File System (HDFS): HDFS incorporates several key features that contribute to its effectiveness in managing vast amounts of data in a distributed environment. One of its primary features is fault tolerance achieved through data replication. HDFS replicates each data block multiple times across different DataNodes, ensuring data durability and availability even in the face of node failures. Scalability is another pivotal feature, allowing organizations to seamlessly add more nodes to the cluster as data volumes grow. The append feature in HDFS allows data to be appended to existing files, making it conducive to scenarios where data is continually added, such as log files. Overall, the architecture and features of HDFS make it a cornerstone for distributed storage and processing in the Hadoop ecosystem, providing the foundation for scalable and fault-tolerant big data solutions.

2.2 LOAD BALANCING ALGO:

Block Placement Policies:.

HDFS utilizes block placement policies to determine the optimal locations for storing replicas of data blocks. The default policy, Rack-aware placement, strategically places replicas on different racks to enhance fault tolerance and reduce the risk of data loss during rack failures. This policy ensures that replicas are distributed across multiple nodes and racks, thereby improving data availability.

Heterogeneous Storage Balancer:.
The Heterogeneous Storage Balancer in HDFS is designed to address disparities in storage capacities among DataNodes. By considering the available space on each DataNode, this balancer intelligently redistributes data blocks to achieve a more equitable distribution. This dynamic approach optimizes storage utilization across the cluster, preventing certain nodes from becoming storage bottlenecks.

Balancing by MOVER Tool:.
Hadoop's MOVER tool is a utility that actively rebalances data across the cluster. It achieves this by moving data blocks from heavily loaded DataNodes to underutilized ones. Operating asynchronously, the MOVER tool continuously monitors the data distribution and dynamically adjusts it to ensure a balanced and efficient use of storage resources throughout the Hadoop cluster.

HDFS Federation:.
HDFS Federation is an architectural paradigm within Hadoop that introduces multiple independent namespaces. Each namespace has its own set of NameNodes and DataNodes, effectively creating separate file systems within a single Hadoop cluster. This approach inherently distributes the workload among different namespaces, contributing to a form of load balancing by allowing each namespace to manage a distinct portion of the overall data.

Cost-Based Optimizer:.
While not a traditional load balancing algorithm, the Cost-Based Optimizer in Hadoop focuses on optimizing query performance. It considers factors such as data distribution, node capabilities, and query complexity to generate an efficient execution plan. By optimizing query processing, this approach indirectly influences the distribution of the workload across nodes in the cluster.

HDFS Router-based Load Balancing:.
HDFS Router-based Load Balancing leverages routers to intelligently route client requests to the appropriate DataNodes. This approach optimizes read and write operations by distributing requests evenly across the cluster. By balancing the load at the router level, this mechanism enhances the overall efficiency and performance of the Hadoop cluster.

2.3 SECURITY:

Security in traditional DFS is solved by using an authentication and a controlled access to files. All mentioned traditional DFS use a Kerberos system as an authentication protocol. The Kerberos is a network authentication protocol which allows the users to authenticate themselves to someone else. The Kerberos prevents nodes in DFS from an eavesdropping or a repeating authentication communication and guarantees a data integrity. It was primarily made for client-server model and provides a mutual authentication: both client and server verify each other identity.

The Kerberos is based on the Needham-Schroder's protocol and Key Distribution Centre (KDC). The Kerberos consists of two logical independent parts: an Authentication Server

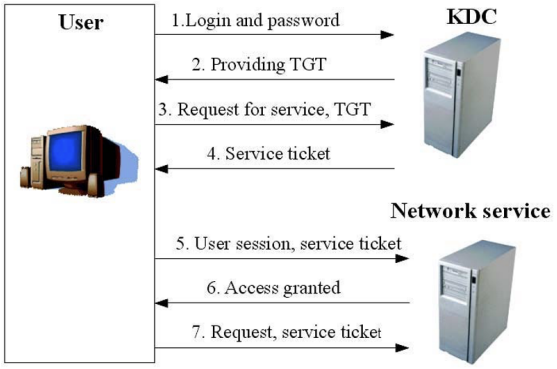

Figure 1: Kerberos authentication procedure

(AS) and a Ticket Granting Server (TGS). The Kerberos uses a ticket which serves for providing users' identity. KDC holds database of secret keys and every user also knows this secret key. This key is used to authenticate user's identity. For communication between two subjects KDC generates a session key for this communication .

Kerberos uses Single Sign-on Mechanism which allows logged in users using several services without the need of logging in to all these services again. After the user has logged in, the user becomes a Ticket Granting Ticket (TGT). When users want to access services, they use TGT to get a Service Ticket (ST). ST is always sent while user requests a service. ST has a valid time. The ticket must be refreshed after expiration [6]. Whole authentication procedure is depicted in Fig. 1.

Traditional DFS:-.
All traditional DFS use Access Control List (ACL) to control access to files. ACL defines a list of rights for all files and directories. This list assigns users and rights for the file or for the directory. Users can access files after they authenticate themselves and have adequate rights to these files. There can be some problems when using ACL (e.g. user must have same ID in the whole system). This problem solves for example OpenAFS by using extended rights for files, and separated user's management.

2.3.1 MODERN TRENDS IN RELIABILITY AND SECURITY

This section will describe modern trends in reliability and security in distributed file systems. Distributed File Systems are still being developed and new trends are described in many papers. This section will summarize these trends. These days, DFS are also used for storing data for mobile devices. Mobile devices have limited capabilities in storage capacity. Communication channel between mobile device and server is slow and unreliable. New trends described in this section were chosen with regards to these devices.

- **Reliability** :- For achieving reliability in distributed file

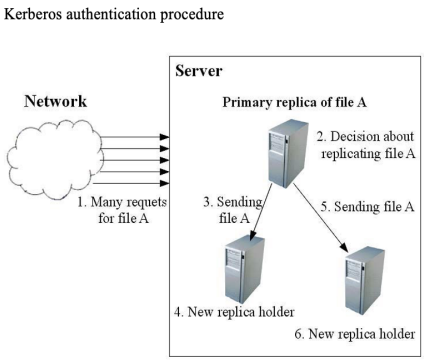

Figure 2: Data Replication

systems, the first step is choosing a suitable file system for storing data and/or metadata. Reliable file systems usually use journaling. The journal technique, can be also used for preventing whole DFS consistency. a DFS which uses a journal as a write-ahead commit log for changes to the file system that must remain unchanged. Every transaction made by a client is recorded in the journal, and the journal file is flushed and synchronized before the change is committed back to the client. The second step in increasing reliability is file replication. Recall that file replication in a traditional DFS is made administratively. A replication algorithm is depicted in below figure.
Dynamic file replication:- File replication for achieving reliability is also used in other DFS. CloudStore [12] typically uses 3-way file replication. If there is a need for replication (e.g. node outage), a metadata server can replicate a file chunk to another node. This conception of file replication for achieving reliability is derived from the Google File System [13]. Reliability in GlusterFS [14] can be achieved by using distribution over mirrors. This means, that each storage server is replicated to another storage server. GlusterFS also supports other ways of storing files (e.g. file distribution to one node or file stripping over nodes).

Example of file storage in the P2P overlay:- Files in this system are split into fragments, which are then stored on clients (peers). Links to these fragments are stored in a Distributed Hash Table. Every client in this system is responsible for files, whose fragment keys are near the peer ID. . A fragment key is made from a 160-bit random value and a combination of a path name and a fragment number as a domain.

While achieving reliability in DFS, data consistency is also very important. Data consistency can be achieved by using file locking. In the system there are usually two types of locks: **a shared lock** for file reading and **an exclusive lock** for file writing. A shared lock is used when the data can be read simultaneously by many clients, but cannot be written simultaneously. While a file is being read by clients, other clients cannot write into this file. An exclusive lock means that only one client can access a locked file and can write file content. In this system there are four kinds of agents. These agents are used for communication in a domain and also for communication between domains.

- **Security**
  Modern trends in achieving security in DFS can be divided into two parts: secure network communication and secure file storing. In this section, modern trends in these areas will be described.

Secure communication :- .
Secure network communication protects data against eavesdropping attacks. The communication is usually encrypted by using a symmetric cipher. This cipher needs a key, which will be used for data encrypting. The same key is also used for data decrypting. the session key is used for ciphering which is a unique key, which is established each time the session is created between server and client. there are many techniques which will prevent other kinds of threats like **user masquerading, man-in-the-middle attack and replay attack**. User masquerading is prevented by using unique credentials, which are selected randomly by the file server and established during a genuine user login session . Man-in-the-middle attack is prevented by checking the integrity of messages by applying message integrity codes. Replay attack is prevented by using a secure communication channel.

Another way of creating keys is Identity Based Encryption (IBE). This algorithm allows deriving a public key from some known aspects of user identity. This information can be, for example, IP address, email address, validation period, etc. IBE also needs a third party entity called a Private Key Generator. This entity provides a private key after authentication for decrypting data.

a DFS which is based on Java and Java Remote Method Invocation (RMI). RMI is used as a middleware mechanism for remote file communication . For secure communication, Java Crypto libraries are used. These libraries perform block level encryption of file content and also serve for communication with the server.

Secure file storing. File content which is stored on data nodes can be stored on a traditional file system or on a secured file system. Storing data on a secured file system allows one to protect data against theft. There are several Encrypting File Systems, such as TransCrypt, NCrypt, eCryptfs, dm-crypt, Microsoft EFS, etc. Some of the Encrypting File systems uses TransCrypt for data encrypting and decrypting. TransCrypt is an enterprise-level, kernel- space encrypting file system for Linux .

Another secure file storing which uses a symmetric cryptosystem and a per-file encryption key. These encryption keys are stored in Meta-Data Storage (MDS).

MDS is trustworthy for maintaining cryptographic keys. As the encryption algorithm, an Advanced Encryption Standard (AES) block cipher in Cipher Block Chaining (CBC) mode is used. Unauthorized file modification is prevented by using a cryptographic hash or "digest" of every file. As a hashing algorithm, Secure Hash algorithm (SHA) family of hash function is used and produced hash is stored in MDS.

3. CONCLUSION

A load-balancing algorithm is used to deal with the problem of load rebalancing in large-scale, dynamic, and distributed file systems in clouds has been presented and to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity. However as Hadoop's use and demand grew in the network, handle big data security became critical, So that authentication mechanism Kerberos is used. This approach has the advantage that one could continue to use the tokens to supplement a different primary authentication mechanism. The integration of secure communication and advanced encryption techniques demonstrates a commitment to ensuring the confidentiality and integrity of data in distributed file systems. Additionally, the exploration of reliability mechanisms like journaling and file replication underscores the ongoing efforts to enhance the dependability of these systems.

4. REFERENCES

- Thakur, Amey Satish, Mega Rizvi, Hasan. (2022). A Comparative Study on Distributed File Systems. 10.13140/RG.2.2.31450.82887.
- K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System," 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, 2010, pp. 1-10, doi: 10.1109/MSST.2010.5496972.
- H. T. Almansouri and Y. Masmoudi, "Hadoop Distributed File System for Big data analysis," 2019 4th World Conference on Complex Systems (WCCS), Ouarzazate, Morocco, 2019, pp. 1-5, doi: 10.1109/ICoCS.2019.8930804.
- Shvachko, Konstantin V., Hairong Kuang, Sanjay R. Radia and Robert J. Chansler. "The Hadoop Distributed File System." 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) (2010): 1-10.
- K. Dwivedi and S. K. Dubey, "Analytical review on Hadoop Distributed file system," 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence), Noida, India, 2014, pp. 174-181, doi: 10.1109/CONFLUENCE.2014.6949336.
- De, Suman Panjwani, Megha. (2021). A Comparative Study on Distributed File Systems.10.1007/978-3-030-68291-0_5.