KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI – 590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

(Approved By AICTE, New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Course Activity Report

DISTRIBUTED FILE SYSTEMS

Submitted in the partial fulfilment for the academic requirement of
6th semester
IN

DISTRIBUTED COMPUTING SYSTEM

SUBMITTED BY

Amar A K	2GI20IS004
Chinmay B W	2GI20IS008
Samarth Awati	2GI20IS034
Shubham P R	2GI20IS040

Under the Guidance of: **Prof. Rakesh Kadkol**

KARANATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI – 590008

(An Autonomous Institution under Visvesvaraya Technological University, Belgavi)

(Approved By AICTE, New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Amar A K, Chinmay B W, Samarth Awati, Shubham P R, of 6th semester and bearing USN 2GI20IS004, 2GI20IS008, 2GI20IS034, 2GI20IS040 has satisfactorily completed the course activity (Seminar) in Distributed computing systems . It can be considered as a bonafide work carried out in partial fulfilment for the academic requirement of 6th Semester B.E. prescribed by KLS Gogte Institute of Technology, Belagavi during the academic year 2022-23 The report has been approved as it satisfies the academic requirements in respect of Assignment (Course activity) prescribed for the said Degree.

Signature of the Faculty Member

Signature of the HOD

Date :

Marks allocation:

RUBRICS FOR COURSE SEMINAR

Sl No.	Rubric Head : Distributed file Systems	Marks	USN			
			2GI20IS004	2GI20IS008	2GI20IS034	2GI20IS040
1	Abstract and awareness related to the topic	5				
2	Presentation	5				
3	Queries Answered	5				
4	Report	5				

*** 20 marks is converted to 10 marks for CGPA calculation**

Table of Contents

ABSTRACT

Distributed file systems have become integral components of modern computing infrastructure, enabling efficient and reliable storage and retrieval of large amounts of data across multiple nodes in a network. This abstract provides a concise overview of the key concepts and characteristics of distributed file systems.First, we explore the motivation behind distributed file systems, highlighting the need for scalable, fault-tolerant, and highly available storage solutions in the face of ever-increasing data volumes. We discuss the challenges posed by centralized file systems and how distributed file systems address these challenges through their distributed nature.

Moreover, we shed light on notable distributed file system architectures and implementations, such as the Google File System (GFS), Hadoop Distributed File System (HDFS), and Ceph. We highlight their distinctive features, including fault tolerance mechanisms, data replication strategies, and the utilization of distributed metadata management.Lastly, we examine emerging trends and challenges in the field of distributed file systems. We discuss the impact of cloud computing, the rise of object storage systems, and the integration of distributed file systems with big data frameworks. We also touch upon security considerations and the importance of data integrity and confidentiality in distributed environments.In summary, this abstract provides a concise overview of distributed file systems, exploring their motivation, fundamental principles, techniques, architectures, and emerging trends. It serves as a foundation for further exploration and understanding of this critical component of modern data storage and processing systems.

INTRODUCTION

A distributed file system (DFS) is a network-based file storage solution that allows for the management and access of files across multiple machines or servers in a distributed computing environment. Unlike traditional file systems that are centralized, where files are stored on a single server, a distributed file system distributes file storage and management tasks across multiple nodes, providing several advantages in terms of scalability, fault tolerance, and performance.

The primary goal of a distributed file system is to provide a transparent and unified view of file storage to users and applications. It abstracts the complexities of file distribution and replication, making it appear as if all files are stored on a single, cohesive file system. Users can access and manipulate files using familiar file system interfaces and APIs, regardless of the physical location of the files.

Distributed file systems are designed to address the challenges posed by modern computing environments, which involve massive data volumes, distributed processing, and the need for high availability.

By distributing file storage and management across multiple nodes, they offer the following benefits:

1. Scalability: Distributed file systems can scale horizontally by adding more storage servers to accommodate increasing data volumes and user demands. They can handle large-scale data storage requirements and support a large number of concurrent users and applications.

2. Fault Tolerance: By replicating data across multiple servers, distributed file systems provide fault tolerance and high availability. If a server fails or becomes unavailable, the system can continue to provide access to files by retrieving them from other replicas.

3. Performance: Distributed file systems can improve performance through parallelism and load balancing. File data can be distributed across multiple servers, allowing for parallel retrieval and storage operations. Load balancing mechanisms ensure that the system efficiently utilizes available resources.

4. Data Consistency: Maintaining consistency across distributed replicas is a crucial aspect of distributed file systems. They employ various techniques to synchronize updates and ensure that all replicas have consistent views of the data. Consistency protocols and distributed locking mechanisms prevent conflicts and data inconsistencies.

5. Data Security: Distributed file systems often incorporate security measures to protect data and ensure authorized access. Authentication mechanisms, access controls, and encryption techniques are used to enforce data privacy and integrity.

Examples of well-known distributed file systems include the Hadoop Distributed File System (HDFS), Google File System (GFS), Ceph File System (CephFS), and Lustre. Each of these systems has its unique architectural design and features, but they all aim to provide scalable, fault-tolerant, and high-performance file storage in distributed environments.

Objectives of the Report:

The objective of the report on distributed file systems is to provide a comprehensive understanding of the key concepts, architecture, applications, and benefits of distributed file systems. The report aims to fulfill the following objectives:

1. Explain the Concept: The report should clearly explain the concept of distributed file systems, highlighting the fundamental principles behind their design and operation. It should provide an overview of how distributed file systems differ from traditional centralized file systems and why they are essential in modern computing environments.

2. Describe the Architecture: The report should delve into the architectural aspects of distributed file systems. It should explain the components, data

organization, and communication mechanisms involved in the distributed file system architecture. Special emphasis should be given to aspects such as metadata management, data distribution, replication, and fault tolerance mechanisms.

3. Explore Application Areas: The report should explore the diverse application areas where distributed file systems are utilized. It should discuss real-world use cases and examples in domains such as big data processing, cloud storage, content delivery networks, scientific research, media streaming, high-performance computing, enterprise file sharing, and virtualized environments. Each application area should be explained in terms of the specific benefits and challenges addressed by distributed file systems.

4. Highlight Benefits and Challenges: The report should analyze the benefits and challenges associated with the adoption of distributed file systems. It should discuss the advantages of scalability, fault tolerance, performance, data consistency, and data security provided by distributed file systems.

5. Provide an Overview of Existing Systems: The report should provide an overview of prominent distributed file systems in use today, such as Hadoop Distributed File System (HDFS), Google File System (GFS), Ceph File System (CephFS), and Lustre. It should highlight their key features, architectural differences, and strengths in addressing specific requirements.

By achieving these objectives, the report on distributed file systems will provide readers with a comprehensive understanding of the subject matter and equip them with knowledge about the principles, architecture, applications, benefits, and challenges associated with distributed file systems.

OVERVIEW OF DISTRIBUTED FILE SYSTEMS

The overview of a distributed file system provides a high-level understanding of its purpose, functionality, and key characteristics. It introduces the concept of distributing file storage across multiple machines and describes how it addresses the challenges of scalability, reliability, and performance in large-scale computing environments. Here is an overview of a distributed file system:

A distributed file system is a network-based file storage solution designed to manage and store files across multiple servers or nodes in a distributed computing environment. It enables users and applications to access and manipulate files as if they were stored on a single centralized file system, even though the data is distributed across different machines.

Key characteristics of distributed file systems include:

1. Scalability: Distributed file systems can scale horizontally by adding more storage servers to accommodate growing amounts of data and increasing workload demands. This allows for efficient handling of large data volumes and the ability to support thousands or even millions of concurrent users.

2. Fault Tolerance: Distributed file systems employ redundancy and replication techniques to ensure high availability and fault tolerance. By storing multiple copies of data across different servers, they can continue to provide access to files even if some servers fail or become unavailable.

3. Data Distribution: Files in a distributed file system are typically divided into smaller units or blocks, which are distributed across multiple servers. This distribution improves data access performance by allowing parallel retrieval and storage operations.

4. Metadata Management: Distributed file systems rely on metadata to track the location, attributes, and permissions of files. Metadata management is crucial for

efficient file access, directory operations, and maintaining file consistency across distributed nodes.

5. Consistency and Coherency: Distributed file systems ensure data consistency by providing mechanisms to synchronize file updates and maintain coherency across multiple replicas. Consistency protocols and distributed locking mechanisms help prevent conflicts and ensure data integrity.

6. Network Transparency: Distributed file systems aim to provide a transparent interface to users and applications, hiding the complexities of data distribution and replication. Users can access files using familiar file system APIs, regardless of the actual physical location of the data.

Examples of popular distributed file systems include:

- Hadoop Distributed File System (HDFS): Designed for storing and processing large-scale datasets in a distributed computing environment, HDFS is widely used in big data analytics applications.

- Google File System (GFS): Developed by Google, GFS is a distributed file system optimized for handling large amounts of data across multiple servers in a highly reliable and scalable manner.

- Ceph File System (CephFS): Ceph is a distributed storage platform that provides a scalable and fault-tolerant file system. It is known for its ability to seamlessly integrate with object and block storage.

- Lustre: Lustre is an open-source parallel distributed file system designed for high-performance computing (HPC) environments. It is commonly used in research institutions and supercomputing centers.

This overview provides a glimpse into the fundamental aspects and characteristics of distributed file systems. Further exploration and study of specific distributed file system architectures and implementations will provide a more in-depth understanding of their inner workings and advanced features.

ARCHITECTURE OF DISTRIBUTED FILE SYSTEM

The architecture of a distributed file system describes its structural design and the components involved in storing, accessing, and managing files across multiple machines in a distributed computing environment. While specific architectures may vary between different distributed file systems, there are some common elements and concepts. Here is an overview of the architecture of a distributed file system:

1. Client Layer:

- The client layer includes the software components and libraries that interact with the distributed file system on behalf of users and applications.

- It provides a user-friendly interface and APIs for file access, metadata operations, and data manipulation.

- The client layer handles user authentication, file system mounting, and communication with other layers of the distributed file system.

2. Metadata Server:

- The metadata server manages the metadata of files stored in the distributed file system.

- It stores information such as file names, directory structure, file permissions, timestamps, and file locations.

- The metadata server maintains consistency and coherence of the file system by enforcing file system semantics and handling concurrency control.

- Clients communicate with the metadata server to perform file operations, retrieve file metadata, and resolve file locations.

3. Data Storage Layer:

- The data storage layer consists of multiple data servers that store the actual file data.

- Each data server is responsible for storing a portion of the file data or blocks.

- Data servers manage data replication, data placement, and data retrieval.

- Replication ensures fault tolerance and availability by storing multiple copies of data on different servers.

- Data servers handle read and write requests from clients and maintain data consistency across replicas.

4. Communication Layer:

- The communication layer facilitates communication and coordination between the client layer, metadata server, and data storage layer.

- It uses network protocols and messaging mechanisms to exchange requests, responses, and metadata updates.

5. Caching and Buffering:

- Distributed file systems often incorporate caching and buffering mechanisms to improve performance.

- Caching involves storing frequently accessed files or blocks in client-side or server-side caches to reduce network latency and improve response times.

- Buffering involves temporarily storing data in memory to optimize read and write operations.

6. Security and Access Control:

- Distributed file systems implement security measures to control access to files and ensure data privacy.

- Authentication mechanisms, access control lists (ACLs), and encryption techniques are employed to enforce security policies.

- Distributed file systems may integrate with existing authentication and authorization systems, such as LDAP or Kerberos, to authenticate users and authorize file operations.

APPLICATIONS OF DISTRIBUTED FILE SYSTEMS

Distributed file systems (DFS) find numerous applications in various domains where scalable and reliable file storage and access are crucial. Here are some common applications of distributed file systems:

1. Big Data Processing: Distributed file systems are extensively used in big data processing frameworks like Apache Hadoop. They provide a scalable and fault-tolerant storage infrastructure for storing and processing massive volumes of data across distributed clusters. Distributed file systems enable efficient data storage and parallel processing, supporting distributed data analytics, batch processing, and real-time streaming applications.

2. Cloud Storage: Distributed file systems form the backbone of cloud storage platforms. They allow cloud service providers to offer scalable and highly available storage to their customers. Distributed file systems enable users to store and access their data from anywhere, providing features such as data redundancy, fault tolerance, and elastic storage capacity.

3. Content Delivery Networks (CDNs): CDNs rely on distributed file systems to store and distribute content across multiple edge servers located in different geographical regions. By replicating content and distributing it closer to end-users, CDNs enhance content delivery speed and improve user experience. Distributed file systems ensure efficient content distribution, cache management, and load balancing within CDNs.

4. Scientific Research and Data Repositories: Distributed file systems are used in scientific research environments to manage large-scale datasets generated from experiments, simulations, and observations. They provide reliable storage and

access to scientific data, allowing researchers to collaborate, analyze, and share data across distributed research communities.

5. Media Streaming and Video-on-Demand (VoD): Streaming services and VoD platforms rely on distributed file systems to store and deliver multimedia content to users. Distributed file systems enable efficient storage and retrieval of video files, support concurrent streaming sessions, and ensure seamless playback across various devices.These applications can improve efficiency, safety, and sustainability in a variety of industries.

6. High-Performance Computing (HPC): Distributed file systems play a crucial role in HPC clusters, where large-scale scientific computations and simulations are performed. They provide shared storage solutions that allow multiple compute nodes to access data simultaneously, enabling efficient data exchange and coordination among compute nodes.

7. Enterprise File Sharing and Collaboration: Distributed file systems offer scalable and secure file sharing and collaboration platforms for enterprises. They allow employees to access and share files across distributed teams and locations, ensuring data consistency, version control, and access control.

In summary, distributed file systems have diverse applications in areas such as big data processing, cloud storage, content delivery, scientific research, media streaming, high-performance computing, enterprise file sharing, and virtualized environments. They provide scalable, reliable, and efficient file storage solutions that cater to the demands of modern distributed computing environments.

CONCLUSION

The DFSs are the most important and widely used forms of shared permanent storage. DSFs are the principle storage solution used by supercomputers, clusters and data centers. Architecture, naming, synchronization, availability, heterogeneity and support for databases will be key issues that are to be taken into consideration while designing the DFS. NFS was the most popular, open, and widely used Distributed File System for many days. It is compatible with multiple operating systems and platforms. As NFS Server is stateless easy crash recovery is the most important advantage of NFS. AFS employs aggressive client side caching with proactive cache-invalidation. This is one of the strengths of AFS. This is the important feature of DFS. But it has some weaknesses like it requires heavyduty, non-standardized cluster management system. and it is compatible only with Linux platform. The negative side of GFS is that it is not a open source version. It is especially designed for Google Applications and can be used only by the people of Google. As DFS has more advantages over GFS it is more reliable to use Distributed file systems.

REFERENCE

1. Sunita Mahajan ”*Distributed Computing*”, Oxford University Press
2. ACM Digital Library
3. Konstantin Shvachko,” *The Hadoop Distributed File System*”, Yahoo-Inc.com.
4. Ghemawat, S., Gobioff, H., Leung, S.T., “The Google file system”, ACM SIGOPS Operating Systems Review, Volume 37