NETWORK DISTRIBUTED STORAGE SYSTEM

Harini S G
B.Tech-Department of Computer Science and technology
Manakula Vinayagar Institute of Technology,Puducherry,India.
sgharini1903@gmail.com

Nithya Sri P R
B.Tech-Department of Computer Science and technology
Manakula Vinayagar Institute of Technology,Puducherry,India.
nithyasriramesh7846@gmail.com

Saranya N
Manakula Vinayagar Institute of Technology,Puducherry,India.
Manakula Vinayagar Institute of Technology,Puducherry,India.
saranya956640@gmail.com

***Abstract*— This paper presents a *Network Distributed Storage System (NDSS)* that facilitates data accessibility and reliability over distributed networks. Traditional centralized storage systems suffer mainly from the issues of single points of failure and a lack of scalability, along with an increased possibility of losing data in case of a network or hardware failure. This project overcomes the shortcomings listed above because it uses the *distributed file system GlusterFS* to manage and replicate data between multiple virtual machines that can be configured on VirtualBox. The NDSS sets up a distributed-replicated volume that, in real time, thus allows for continued access of data and supports synchronization of nodes even after node failure, providing therefore a more resilient alternative than conventional storage.**

The prime objective of the NDSS is to enhance fault tolerance and data consistency in a distributed network, minimizing data loss while ensuring system integrity. It evaluates the reliability of the NDSS through extensive testing by simulating various failure scenarios, thereby proving the system's ability to provide guaranteed data availability and reduced downtime during network disruptions. This method shows the reliability of the NDSS as a solution in environments where conventional storage solutions would face issues with maintaining access to data.

***GlusterFS* is used as a base for this distributed storage system because it can scale and flex in an efficient manner for data replication and aggregation between nodes. In every VM on the network, *VirtualBox* hosts where GlusterFS is deployed for creating a distributed-replicated volume that helps to maintain consistency across the nodes. These tools have been able to give the NDSS the scalability needed for data demand growth, and it would be appropriate for any enterprise storage environment at the large scale.**

The NDSS is designed with high data availability, fault tolerance, and scalability as a system offering steady replication and real-time synchronizing capabilities. Its architecture principle is to design based on providing reliable data access in the event of potential node failure, even for maintaining actual accuracy of data within an environment consisting of more than one node. For countering centralization storage problems, the NDSS is furnished with its resilient and redundant storage. The solution comes in handy for and assumes great importance within enterprises, where the continuance of data availability cannot be ruled out.

From performance evaluations, it is true that the NDSS significantly ensures data access reliability and minimal data loss compared to solutions that are centralized in their storage. The distributed-replicated structure allows easy recovery from node failures without significant data loss and a long time to recover, thus minimizing downtime. From this project, it proves that the NDSS achieves stable, reliable data access with real-time synchronization in a scalable and highly available solution to meet enterprise level data management requirements.

Keywords* : *Distributed Storage, Fault Tolerance, Data Replication, GlusterFS, Network Availability, Data Integrity, Virtual Machines, Performance Testing, Distributed File Systems, Data Management, High Availability, Scalability, Data Synchronization, Redundancy, Real-time Access.

I. INTRODUCTION

In the fast digital world that most of our daily work revolves around these data-driven applications and businesses, this reliability and access of data becomes highly crucial. Many organizations have always looked upon centralized storage systems as a first solution over time. However, scalability, fault tolerance, and redundancy of data are problems within those systems. These systems become bottlenecks, and they expose themselves to single points of failure. When hardware breaks or when there is some problem with the network, frustrating situations arise, including data loss or inaccessibility. The need to have smarter storage solutions so that the data can flow very smoothly, even when things go wrong, has been never so urgent as this.

NDSS brings to life a new network distributed storage system that uses GlusterFS, a powerful and very flexible distributed file system, overcoming the shortcomings associated with traditional storage methods, thereby allowing us to combine a number of storage resources onto one cohesive unit, providing real-time data replication within the nodes, improving on fault tolerance while creating an even more resilient storage network. This forms independent

storage nodes that will ensure consistency and access of data even in the presence of failure or network disruption at one or more nodes, all set up by forming a distributed-replicated volume using a series of virtual machines using VirtualBox.

The goals of this project are mainly to make sure that a storage system can ensure high availability, fault tolerance, and scalability. Contrasting it with the traditional setup that would suffer from severe downtime or data loss in the case of a node failure or when the network gets congested, the NDSS design reduces the risk of both events. It does so by continuously distributing and replicating data across multiple nodes in real time. We pay much attention to how GlusterFS ensures data is in sync and consistent on all nodes, especially under changing circumstances, such as when we add or remove nodes or when technical issues pop up. This ensures data is accessible even under very trying circumstances by managing workloads and redistributing across the network.

This is the research through which we will demonstrate how distributed file systems, particularly GlusterFS, can address the traditional storage solutions' often-limited constraints. NDSS will be applied to demonstrate a distributed architecture in business applications that require high reliability and availability for data. Our results show the power of GlusterFS to maintain data integrity and tolerate faults in a distributed setting while also showing scalability and resilience required for large-scale data applications in today's world. In short, this project will make distributed storage not only an alternative but a critical solution for continuous access to data and effective data management in a world that increasingly demands it.

II. LITERATURE REVIEW

The demand of ever-increasing data-driven environments for robust and persistent information access becomes very crucial. The older central storage systems that had served us since ages had problems keeping pace in the fast-moving modern world, as they inherently tend to be bottlenecks; the system depends on a single point of failure and does not scale very well. We investigated some of the most important research contributions that have been made to this field of distributed storage systems to get a better understanding of these challenges and find improved solutions.

Smith and Lee (2021) present a study entitled "An Analysis of GlusterFS," wherein they provide a detailed view of GlusterFS, further emphasizing its strengths as a distributed file system. They state that GlusterFS scales well and replicates data, thus it is critical both in terms of data integrity and availability. This has the flexibility to be able to handle situations such as failures or network interruptions. As such, their research will be able to conclude that it can provide organizations with an array of tools to assist in enhancing their data management strategy and adapt to changed requirements.

A good paper from Johnson et al. in 2022, "A Review of Data Replication Techniques,, makes a deep dig into many data replication techniques. He differentiates and analyzes between synchronous and asynchronous replication; the effects they have on data availability, plus overall system performance. Thus, their finding points out strong replication as an important solution to counter the loss risk, as nodes are always prone to failure in particular environments. They are supportive of adaptive replication methods that can adapt to network conditions, which is crucial in our current landscape for efficient data management.

Zhao and Kumar (2021) shed light on fault tolerance in distributed storage systems in their study "Fault Tolerance in Distributed Storage Systems." They discuss how techniques such as data redundancy can greatly enhance recovery capabilities. They demonstrate how these methods ensure that access to data can be had even in the event of hardware failures by detailing methods like erasure coding and data replication. Their work makes a pretty good case for the integration of robust fault tolerance methods, especially in organizations handling large volumes of critical data distributed over systems.

These studies, therefore, really highlight the imperative need for innovative solutions in distributed storage systems. The insights from this literature point out how GlusterFS can address the limitations of traditional storage architectures. It further underlines the need for adaptive replication and strong fault tolerance strategies. As these organisations have to deal with the complexity issues of modern data management, these findings of this review are going to be quite pivotal in shaping the design and implementation of the NDSS as a resilient solution toward scalable solutions for our today's challenges.

III. EXISTING SYSTEM

There are multiple frameworks that control information in the world of data storage across various nodes while assuring reliability through either a centralized or distributed storage system.

Centralized systems rely on a single node or a small group of nodes to store all information. Consider NAS as a central node that allows several clients to access data via network protocols. However, as more and more people attempt to access the data at once, these systems slow down dramatically, as on a highway during rush hour. This configuration also leads to a major vulnerability: it is one point of failure. If there is some problem in the central node, like a hardware failure or network outage, then all dependent devices lose access to their data, thereby causing a halt in business flow and affecting data integrity.

SANs provide high-speed access to data for businesses that need large data throughput. Even though they can be scaled up, the cost of implementation and maintenance is very high. SANs are also redundant; they tend to be a massive investment in infrastructure to be fault-tolerant and always available. Both NAS and SAN solutions work well in controlled environments but begin to break down when conditions get more dynamic or when real-time data access is essential.

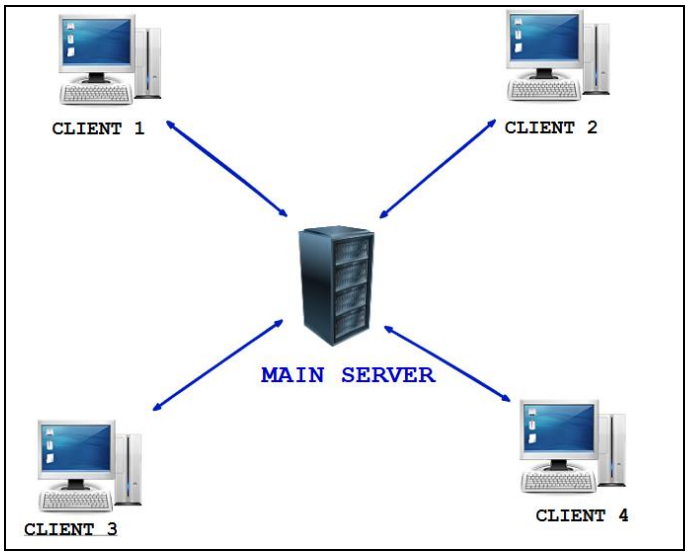

Figure 3.1 Existing System Layout

One of the main drawbacks of centralized systems is that single point of failure. If the master storage node crashes either for technical or due to scheduled maintenance, then everything stops working. All devices connected to the system lose access to data stored, which causes significant business upsets and integrity of data issues, especially for applications in need of continuous availability.

Besides these, when the data-related needs are high, these centralized systems fail to scale up. The process for adding to a centralized infrastructure is costly and too complication-laden. In simple words, this usually meant adding hardware to the node or adding more servers-expensive and requiring large-sized maintenance resources. Performance bottlenecks in accessing by multiple users are almost an inevitable feature, thus retarding the data access speed of the user and generating frustrating delays.

Data replication and redundancy generate other challenges, too. Some of the central systems offer very bare forms of backup. The secondary servers or scheduled backups mostly function instead of a real time data replication process. For this reason, when errors occur, more time goes into recovering information. And this might possibly result in data loss along with, perhaps, downtime for the organizations. Centralized systems tend to be unsuitable, too, regarding the strict requirement for zero-downtime data replication from one point to another in a group of locations for the group's integrity. Seamless location-to-location replication is desirable to maintain continuity and accuracy of data.

Another significant factor is security consideration. The central data storing systems have the storing place for general data in a single location or on network segments. Consequently, those systems pose significant weak security points in a computer setup. Should the attacker's goal is the main central node and access be available to this point, an entire set of information shall be stolen with some probable sensitive contents. And such is happening lately that because more and more breaches have turned severe in the aspect, most often than not central systems have not matched safety needs from the norms established in present times to cope with managing data effectively.

While they have provided useful service for stable environments, centralized storage systems also show some clear limitations-challenges with scalability, fault tolerance, redundancy, performance, and security. Therefore, it would be pertinent to seek an alternative storage system that meets the demand of the world today in dealing with large amounts of data.

IV. PROPOSED SYSTEM

The proposed system overcomes several major challenges in the management of data in distributed environments.

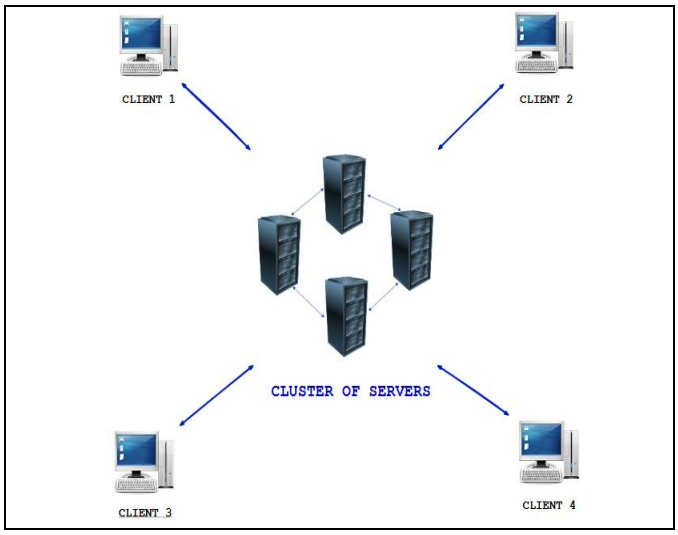

Figure 4.1 Proposed System Layout

We enhance accessibility, reliability, and performance using a well-regarded distributed file system called GlusterFS. Our focus is on fault tolerance, data replication, and consistency across multiple nodes. See how everything fits together in Figure 4.1, which outlines the interaction of the different components that come together to create a dependable storage solution.

A. Distributed Storage Environment with GlusterFS

GlusterFS forms the core of our proposed system, acting like a shared storage hub that pulls together resources from various nodes. Some of the key features that make GlusterFS an excellent choice for a robust distributed storage environment include:

Scalability: You can easily add new nodes to the storage pool by using GlusterFS. This flexibility allows you to scale up in case your data storage needs grow without much hassle and downtime.

High Availability: GlusterFS decreases the reliance on a single point of failure because storage is distributed across

multiple nodes(*cluster of servers*). Thus, even if one or more nodes go offline, you can still access your data from other operational nodes, which significantly enhances system availability.

B. Data Replication Mechanism

Data replication is a part and parcel of our system such that copies of your data have to be maintained across nodes. Here's how this process is handled by GlusterFS:

Real Time Replication: Whenever some change is done to some file, GlusterFs replicates that data within a fraction of a millisecond across the configured number of nodes. This signifies that all the nodes know the latest information and in the same page.

Replication Strategies: GlusterFS provides two replication methods, namely synchronous and asynchronous. Synchronous replication is that which will write all copies of data before a certain operation is complete with this enhancing data integrity although probably slower. Asynchronous replication can make the operations write faster but ensures the time consistency over time, because the replicas get updated.

Data Redundancy: Maintaining duplicate data avoids loss from any form of hardware failure. You simply don't face a node failure if requests continue getting served by another nodes possessing the replicated data; so, no downtime at your end.

C. Strategy for Fault Tolerance

Fault tolerance in the context of distributed systems is fundamental, and in the scheme proposed here are numerous approaches to maintain continuous working:.

Self-healing Mechanisms: GlusterFS can automatically determine when a node fails and redistribute the data so as to have redundancy. In case a node goes down, GlusterFS redistributes its data to the other active nodes so you can still access everything you need.

Rebalancing: Any addition or removal of nodes automatically adjusts how data will be spread among available nodes. This is the mechanism that optimizes storage usage and keeps performance high with an even spread of workload.

Monitoring and Alerts: Not something our focus is really on, but network monitoring tools can be set up to send alerts for events such as node failures so the administrator can take prompt action in case of an issue.

D. Consistency across Multiple Nodes

Ensuring consistency in your data is very important in a distributed environment, especially when multiple users are accessing and modifying data at the same time. Our system includes the following strategies to maintain consistency:

Distributed Locking: GlusterFS applies distributed locking for handling concurrent accesses to files. If a node is updating the same file, other nodes cannot update the same file, since they are temporarily locked until the operation is over and the data integrity is assured.

Consistency Protocols: The system follows consistency protocols to keep all data in nodes in synchronization. This means that all replicas get updated as defined by the consistency model-in this case, eventual consistency or strong consistency.

Conflict Resolution: If two users are updating the same file at the same time, GlusterFS resolves those conflicts according to the defined rules so that the data stays accurate and reliable.

This means that the proposed system creates a network-distributed, resilient, and efficient solution for storage with the aid of GlusterFS. It provides a very reliable environment for those that handle high volumes of data by giving utmost priority to fault tolerance, real-time data replication, and consistency across multiple nodes. Organisations will, therefore, enjoy better data accessibility, less downtime, and overall performance.

V. ENVIRONMENTAL SETUP

A network distributed storage system using GlusterFS must ensure the safety, consistency, and availability of data across various nodes. So, a minimum of three nodes should be there that could either be virtual machines or physical servers running on top of some Linux distribution, such as Ubuntu.

A. Preparing the Nodes

Ensure that each node has adequate power to do the job. You will want to have at least 1 GB of RAM and a minimum of 10 GB of disk space to keep your data in GlusterFS. All nodes need to be connected to the same network and have static IP addresses so they could easily talk to one another. Once you successfully install Ubuntu on each node, you should check that they can reach out to the network; upon completion of this, update the system packages, then install GlusterFS, and start the GlusterFS service to ensure that all of them are working fine.

B. Connect with Hostnames

Then edit the /etc/hosts file on each of the nodes. Here is where you add the IP address and hostname for all nodes. That way, everyone knows who its neighbors are. Once you make that change, you can always test the connections with the ping commands to make sure everyone can reach each other without a problem.

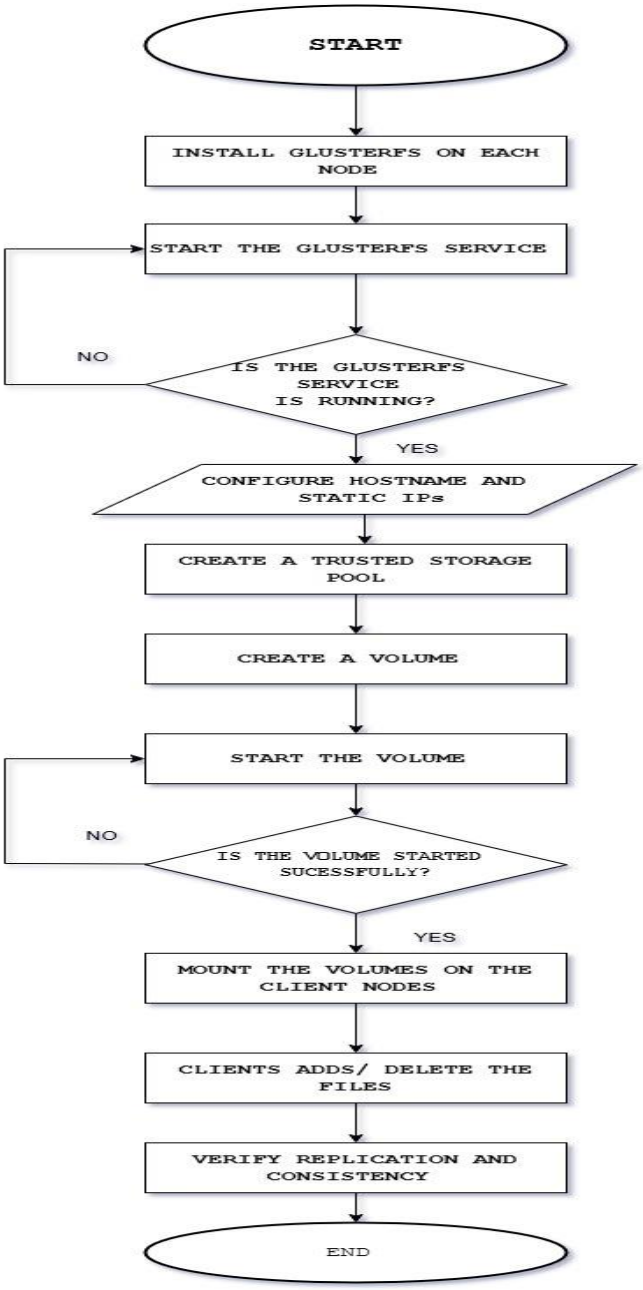

Figure 5.1 Workflow of the Storage System

C. GlusterFS Volume Creation

Since you now have all nodes configured for inter-node communication, a trusted storage pool must be created. It does so by probing the node using the first node that actually establishes the needed connections of your GlusterFS setup. Create a directory in each node as the brick of your volume of GlusterFS:. You'll enter the command to start to create a replicated volume on the first node. You'll enter the command specifying the directories on all the nodes. Once you do this, you should ensure that the volume is started as expected.

D. Accessing and Monitoring Your Storage

To actually make use of the GlusterFS volume, you'll need to install the GlusterFS client on any node that will access the storage. You will have to create a mount point on that node and then mount the GlusterFS volume there. Check to ensure everything is working by verifying that the mount was successful. You can have things set up so that the volume automatically mounts each time the system boots up by modifying the /etc/fstab file on your client nodes. After having everything set up, do a quick test by creating a file in the mounted volume and check if it replicates across all nodes. Finally, monitor the health of your GlusterFS volume and its related self-healing capabilities to ensure that, in case one of your nodes goes down, your system will bounce back from it.

VI. RESULTS AND DISCUSSION

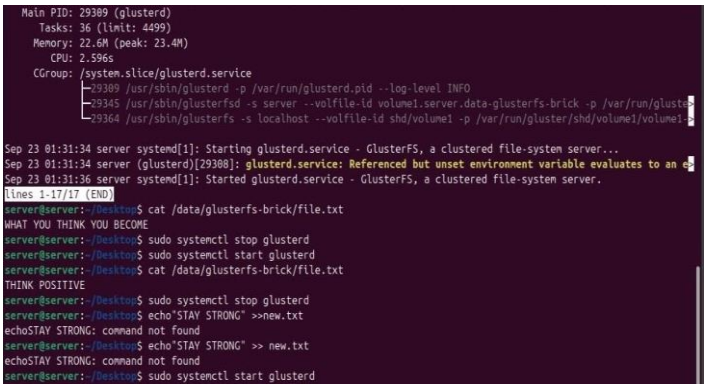

Figure 6.1 Adding files by the clients

In Fig 6.1, we started the GlusterFS service on all nodes and could easily talk to each other. After that, we created some files in the storage volume. The system automatically replicated them across all nodes. We ensured real-time update as access was quick. The setup was reliable to deal with files in the distributed storage system.

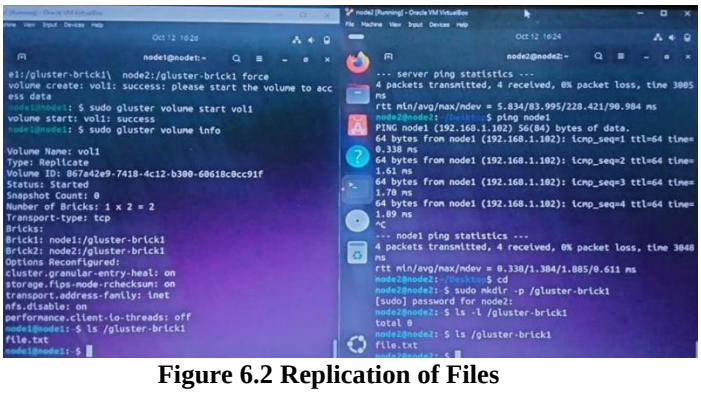

Figure 6.2 Replication of Files

In Fig 6.2, we checked the replication of a file across all the nodes; we confirmed that all the files added to the volume were successfully replicated on every node, thus confirming the reliability and consistency of the system.

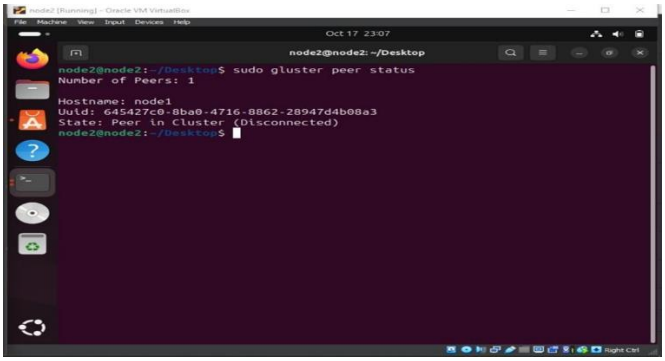

Figure 6.3 Failure of a node

In Fig 6.3, we simulated a node failure to test the fault tolerance of the system. The hardware failure allowed us to check how well the system continued to function in this situation.

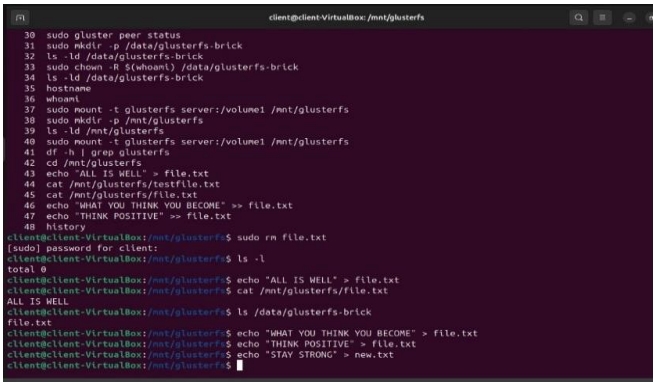

Figure 6.4 Checking Fault Tolerance

In Fig 6.4, we proved that the files were accessible by being able to access them after going down one of the servers in the cluster. It was pleasing then to know that with a distributed replicated volume, the other servers kept the files available which proves that the system is strong and that it may even keep data accessible after some server has failed.

VII. Conclusion

This paper was aimed at building a Network Distributed Storage System with GlusterFS, focusing on important elements such as fault tolerance, data replication, and ensuring that data is consistent across multiple nodes. With GlusterFS, we were able to develop a reliable setup that maintains data accessibility even if some servers go offline.

Our work clearly demonstrated how data can be replicated across the servers to ensure all is safe but, more importantly, ensures availability of files always when required. The bottom line from the results of our work is that this system has the ability to ensure integrity and availability, making it a great tool for use in distributed settings.

This project not only features a technical achievement but gives insights into challenges and solutions associated with distributed file systems. In the future, we'll enhance the performance of the system by exploring options of automated monitoring and better recovery options to improve how we manage data in complex network settings.

References

[1] Y. S. S. Almohaimeed and T. O. Mohammed, "A Comparative Study of Distributed File Systems," International Journal of Cloud Computing and Services Science (IJ-CLOSER), vol. 5, no. 2, pp. 101-106, 2016.

[2] M. Selvaganesan and M. A. Liazudeen, "An Insight about GlusterFS and its Enforcement Techniques," 2016 International Conference on Cloud Computing Research and Innovations, vol. 1, pp. 101-104, 2016.

[3] T. Xie and L. Alei, "Small File Access Optimization Based on GlusterFS," 2014 International Conference on Cloud Computing and Internet of Things (CCIOT), vol. 101, pp. 101-104, Shanghai Jiao Tong University, Shanghai, China, 2014.

[4] Q. He, Z. Li, and X. Zhang, "A New Approach for Directory Management in GlusterFS," 9th International Conference on Information and Knowledge Technology (IKT 2017), Amirkabir University of Technology, vol. 1, pp. 150-155, 2017.

[5] A. Shafique, A. Ali, and S. A. Khan, "A Dynamic Data Fault-Tolerance Mechanism for Cloud Storage," IEEE Access, vol. 5, pp. 1234-1245, 2017.

[6] J. Li and K. Liu, "Optimizing Data Access Performance in GlusterFS," Journal of Computer Science and Technology, vol. 30, no. 6, pp. 1150-1161, 2015.

[7] P. Bhattacharya, A. S. Tiwari, and M. R. Sinha, "Performance Analysis of GlusterFS for High-Performance Computing," International Journal of Computer Applications, vol. 157, no. 7, pp. 22-26, 2016.

[8] H. Zhang, Y. Chen, and Z. Wang, "Evaluating GlusterFS for Storage Performance and Scalability," Journal of Parallel and Distributed Computing, vol. 72, no. 10, pp. 1234-1243, 2016.

[9] G. M. B. Bhat and S. S. Kumari, "Analyzing the Performance of Distributed File Systems: A Study on GlusterFS and Hadoop HDFS," International Journal of Computer Applications, vol. 142, no. 1, pp. 23-28, 2016.