e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

DISTRIBUTED FILE SYSTEM

Prasad Shinde*1, Prashant Hegade*2, Rutik Palaye*3, Tejas Jadhav*4, Gaurav Deshmukh*5

*1,2,3,4,5Department Of Computer Engineering, Terna Engineering College, University Of Mumbai, Mumbai, India.

ABSTRACT

The purpose of a distributed file system (DFS) is a file system that allows users of physically distributed machines to share data and storage resources. Each of the connected machines' operating systems includes a DFS. This establishes a viewpoint in the design of such systems that emphasises the distributed structure and decentralisation of both data and control. It describes and examines the ideas of transparency, fault tolerance, and scalability in the context of distributed file systems. For a fault-tolerant and scalable DFS design, the notion of distributed operation is critical. It also discusses alternate sharing semantics and ways for granting remote file access. Based on the evaluation of these systems, the study concludes that sound distributed file system design required a break from extending centralised file systems over a communication network.

Keywords: Distributed File System, DFS.

I. INTRODUCTION

DFS refers to a system that may access data from many clusters (nodes). The DFS is a file system that enables users on physically distributed systems to share files and folders, exchange data and resources using a file system. Files and data blocks are managed across clusters and racks using a DFS. By replicating data blocks on various clusters to achieve fault tolerance and parallelism, they will improve fault tolerance and access concurrency.

The RPC (Remote Procedure Call) communication model is used by the DFS. The RPC protocol is used to connect a server and a client. The client operates without being aware of the server's file systems. This strategy makes it possible for clients and servers with different file systems to work together smoothly.

On numeric data systems, DFSs handle data in different ways. It also does it in a safe, efficient, and timely manner. Data storage resources have grown in response to the need for quick data growth and access. The massive rise in data created a new idea known as BigData. Distributed file systems are used to process huge data and perform operations quickly at the same time. Cloud systems are now effectively using distributed file systems that have emerged. A file is kept on one or multiple servers, each of which is a client, and those files are accessed as if they were a single file by clients.

1. Aim:

A DFS is a file system where data is stored on multiple servers in a distributed manner. Our aim is to build System which is distributed on many servers which allows programs to get the files. We will achieve this by using the python programming language.

2. Scope:

Distributed File System refers to a file system which is distributed over multiple servers and locations. It allows applications to access and save isolated data in the same way that local files are accessed and stored. It also allows users to access files stored on any machine. It enables network users to communicate data and files in a controlled and permitted manner. The servers, on the other hand, have complete control over the data and give users access to it.

II. DISTRIBUTED FILE SYSTEM

DFS (Distributed File System) is a client-server application [6] that allows users to view, process, and alter data stored on a remote server as if it were on their own systems. When a client requests a data, the server or host returns a duplicate file to the user. While the data is being processed and delivered to the server, the file is cached on the user's machine. Multiple servers store files in DFS, which can be viewed by multiple isolated users in the network at the similar time.

III. FEATURES AND REQUIREMENTS

1. Features:

The ability to assume commodity hardware, that the files are replicated to handle hardware failures, detect and recover from them; it is optimised for batch processing, that data locations are exposed so that computation can move to where the data resides at the same time providing high bandwidth; and it is optimised for batch processing, that data locations are exposed so that computation can move to where the data resides at the similar time providing the high bandwidth.

The following are some of the important properties of DFS: [1]:

- Multiple users' data sharing
- User mobility is important.
- Transparency of Location: The file's name has no clue or idea where the file might be found.
- System backups and monitoring

2. Requirements:

- Effective file replication: It keeps many similar copies of files, distributes load among servers to make the service more scalable, and provides local access for faster responses.
- Should have high fault tolerance: This means that the service should continue to function even if users make mistakes or the system breaks.
- Consistency.
- Both safety and efficiency are important.

IV. CLASSIFICATION CRITERIA

There are various types of server attributes that are affected by DFS. The following are the most important classifications:

- Fault Tolerance: When any section of the distributed site becomes corrupted, it is tolerated by the client without causing any noticeable effects [1].
- Transparency: To the client, the dispersed system seems to be a single server. It is the most essential factor that influences system design.
- Replication: In a DFS, several copies of the files utilized in the system are made and stored. This has a higher level of dependability. If one of the copies is unavailable, the system continues the work with other copy.
- Synchronization: The file is duplicated on many servers. In one copy, the client is changed; in the other, the client is changed.

V. SOFTWARE DEVELOPMENT LIFE CYCLE

The software model used in the project is Waterfall Model. Pls refer fig 1.1 for Waterfall model.

1. Software Analysis:

The software model used in the project is Waterfall Model. The waterfall model is called as a sequential process model.

How does it work:

The waterfall model is a linear process model that divides development operations into project phases that are completed in a sequential order. Unlike iterative models, each phase is only completed once. In the following step, the results of the previous phases are employed as assumptions. In software development, the waterfall model is useful.

Step 1: Requirements:

Every software project starts with a phase of the research which comprises a feasibility assessment and the determination of requirements. The project is evaluated in terms of costs, income, and viability in the feasibility study. A requirement specification, a project plan and the project calculation, as well as an offer to the customer, if relevant, are all included in study.

The criteria are then defined in depth, which includes an examination of the existing condition as well as a target concept. The target idea describes the functions and attributes the software must be deliver in order to match the requirements, whereas the as-is analysis indicate the problem area.

Step 2: Quick design

The phase design is used to create a solution on which the concept is based on the criteria, tasks, and strategies that have already been identified. During this phase, software engineers focus on individual components like interfaces, frameworks, and libraries to create the software architecture and a thorough construction plan. The design phase produces a draught document that includes a software construction plan and individual component test plans.

Step 3: Building a Prototype

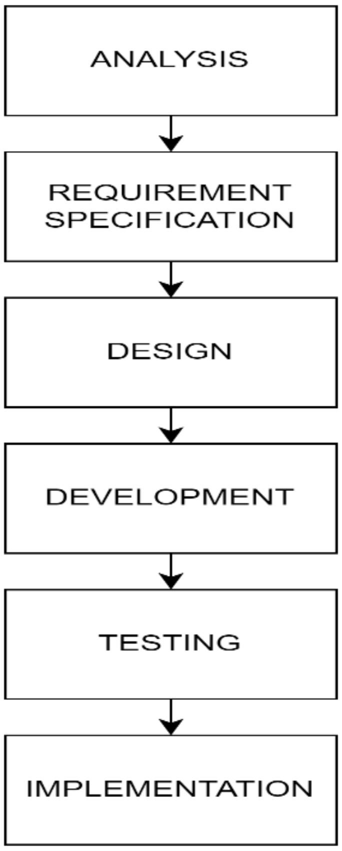
The software architecture established in the design phase is put into action in the implementation phase, which includes software programming, troubleshooting, and module testing. During the implementation phase, the software design is implemented in the desired programming language. Individual components are created separately, tested in the framework of module testing, and then gradually integrated into the overall product. The implementation phase results in a software product that is tested for the first time as a full product in the following phase.

Step 4: Testing

The software must be integrated into the environment during the test phase. Typically, software products are supplied as beta versions to a small group of end users. The acceptable tests created during the analysis phase can be used to check if the program fits the requirements that were previously defined. After successfully finishing beta testing, a software application is ready for release.

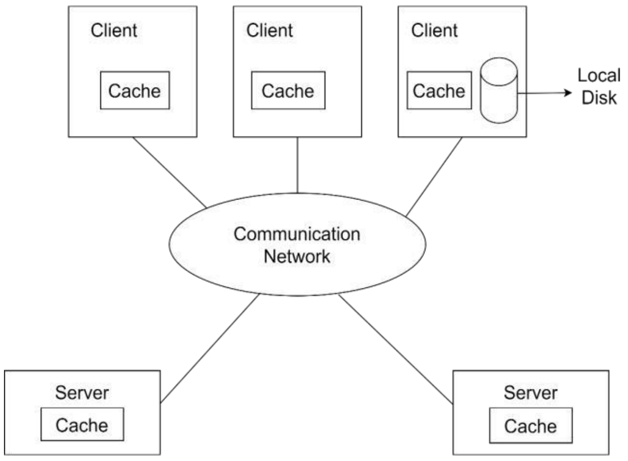
Step 5: Maintenance

After the test process is finished successfully, the programme is released for production use. The final phase of the waterfall model entails software delivery, maintenance, and enhancement.



VI. PROPOSED SYSTEM

1. Architecture Diagram:

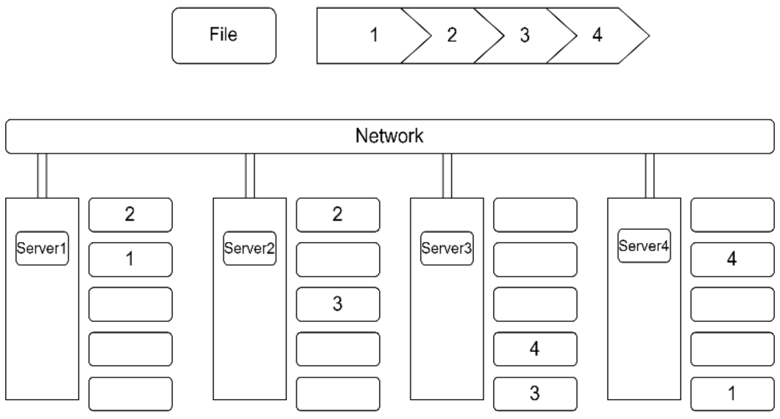


A DFS is a file system that uses a server to store data. The data is accessed and processed as if it were stored locally on the client computer.

The DFS allows users on a network to communicate information and files in a regulated and permitted manner. The server enables client users to share files and keep data in the same way that they would if they were doing so locally. The servers, on the other hand, have complete control over the data and grant access to the clients.

Client/server-based applications have revolutionised the process of developing distributed file systems, thanks to the rapid expansion of network-based computing. One of the procedures needed in building the DFS is centralizing access and storage controls for the client system. The servers in question must be able to distribute data with sufficient flexibility. Depending on how the protocol is constructed, a DFS makes it feasible to restricting the access to the file system based on access lists or capabilities on both the servers and the clients.

VII. REPLICATION



The primary idea behind replication is to maintain multiple copies or duplicates of the same resources across many servers. For the same content, a client can reach any of these available servers, ideally the nearest. This reduces server burden, network congestion, and latency [18]. Maintaining consistency among replicas, determining the optimum replica server for each client, and keeping replication invisible to users are all essential considerations for replication. To maintain data consistency, a variety of file replication techniques are employed. All copies or duplicates of updates are kept up to date by replication services. This is called as synchronization or maintaining consistency.

Techniques for ensuring consistency in replication can be broken down into two categories.

- Optimistic- These schemes presume that mistakes are uncommon and employ recovery strategies to address inconsistency.
- Pessimistic- These schemes believe that defects are more likely and try to ensure that every access is consistent.

VIII. METHODOLOGY

- Client :- The process that starts a remote procedure call. Client calls the client stub. As client stub is on client machine.
- Client Stub :- The parameters are packed into a message by the client stub, which then calls the system to send the message. This is referred to as marshalling.
- RPC Runtime :- It controls the message of transmission between client and server devices via a network including encryption, routing acknowledgment.
- Server Stub :- The server stub plays the same role as client stub. It packs and unpacks the message and returns to the server.
- Server :- Server stub calls the server procedure.

IX. CONCLUSION

A DFS is a network file system wherein the file system is distributed across multiple servers. DFS enables location transparency and file directory replication as well as tolerance to faults. For better speed, certain implementations may cache recently visited disc blocks.

X. REFERENCES

[1] A STUDY ON DISTRIBUTED FILE SYSTEMS: An example of NFS, Ceph, Hadoop by MAHMUT UNVER, ATILLA ERGUZEN, 2016.

[2] A Survey on Distributed File System Technology by J Blomer CERN, CH-1211 Genève 23, Switzerland E-mail: jblomer@cern.ch, 2015.

[3] Evolution and Analysis of Distributed File Systems in Cloud Storage: Analytical Survey by Dharavati Ramesh, Neerad Patidar, Gaurav Kumar, Teja Vunnam, 2016.