An Overview of File Server Group in Distributed Systems

Muhammad Zakarya, Izaz Ur Rahman, and Imtiaz Ullah

***Abstract*—The file server is a key factor to accomplish the data sharing essential in distributed systems. The file server is perhaps the most heavily used resource of the distributed systems and as an outcome; its performance is vital and critical to the victory of the system. The explosive growth of the Web contents and Internet users has led to increasing concentration on two major challenges in these systems: scalability and high availability of network file system. A simple load-sharing mechanism for the NFS client to switch to a lightly-load server based on the number of NFS client's RPC requests for a period of time, making these systems more efficient and scalable. The replication techniques are used for improving the availability of Distributed systems in FSG. We also discussed distributed application in terms of java programming language. In this paper we have proposed a new architecture for Distributed Systems that provides better security. Our proposed solution is more secure & efficient in the sense that a client can not search the whole group for a specific file, because the group header maintains a huge database of all the group members along with their specific services they offers. Also load balancing mechanism is improved where the group header keeps a record of connected users to a specific server. Every group member is not required to advertise their state.**

***Index Terms*—Sun network file system (NFS), file server group (FSG), andrew file system (AFS), group header (GH), secure file system (SFS), log-structured file system (LFS), serverless file system (XFS / xFS).**

I. Introduction

Distributed file systems present remote access to shared file storage in a shared and networked environment. In FSG system, it improved the reliability of file system through replication to handle the effects of failures. An efficient consistency control protocol is previously proposed to ensure the consistency among replicas.

High Availability is required in today distributed shared environments, because if services are not available any time to its customer, then there is the possibility that customers will find some substitute service providers.

The implementation of the file server group, FSG is based on NFS and interacts by underlying IP multicasting. In designing system, the collection of replicated servers is treated as a group. Each group is assigned a group IP address. The IP address will be used by the underlying multicast protocol to deliver messages to all servers in this group. With multicast communication it is possible to implement distributed systems without any explicit need to know the precise location of data. Instead, peers find each other by communicating over agreed upon communication channels. To find a particular data item, it is sufficient to make a request for the data on the agreed upon multicast channel and any node that holds a replica of the data item may respond to the request. This property makes multicast communication an excellent choice for building a system that replicates data.

Manuscript received September 28, 2012; revised November 2, 2012.

Muhammad Zakarya and Imtiaz Ullah are with Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan (e-mail: mohd.zakarya@awkum.edu.pk, ask4imtiaz@gmail.com).

Izaz Ur Rahman is with Department of Computing & Mathematics, Brunel University, London, UK (e-mail: izaz.rahman@brunel.ac.uk).

II. Related Work

NFS file handles, *fHandle*, are normally created by the server and used to identify uniquely a particular file or directory on the server. The client does not normally create file handles or have any knowledge of the contents of a *fHandle*. As shown in Fig. 1 below, it illustrates the system model of FSG. A user on the client machines uses the "mount" command to connect to the sever group as the general UNIX mount command. The only difference between a UNIX mount and the proposed “mount” command is that the "host:pathname" parameter is replaced by "multicast IP address: pathname".

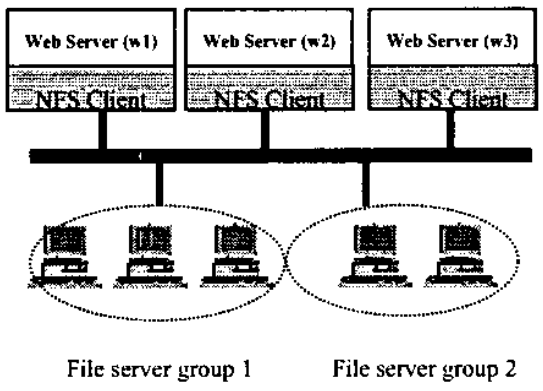


Fig. 1. System model for FSG

In Client Server Scenario the following steps are taking hold when a client wants to read data from a specific file server. This solution was proposed in [1], [3], [5].

1) The client generates an I_fHandle for mount point.
2) The client sends the mount request, carried I_fHandle to server group.
3) Each server in that same group creates a fHandle for the mount point.
4) The mountd process in server sends I_fHandle and fHandle to nfsd process.
5) The mountd process replies "ok" to client.
6) This I_fHandle is handed over to NFS client.
7) The client issue RPC calls to server/server group. On server side, the server transforms the I_fHandle into the real fHandle before executing the request.

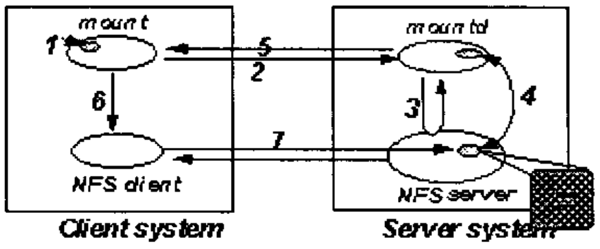


Fig. 2. Scenario of proposed mount procedure

Replication is a technique that allows improving the quality of distributed services. But still some major issues in this design are [2]

1) How do we select and estimate the metrics for taking replication decisions?
2) When do we replicate a given document?
3) Where do we place the replicas of a given document?
4) How do we ensure consistency of all replicas of the same document?
5) How do we route client requests to appropriate replicas?

In this solution scalability & efficiency is achieved through load balancing technique, which enables NFS client to switch to a lightly-load server based on the number of NFS client's RPC requests for a period of time, making these systems more efficient and scalable. In this case every server is responsible for advertising its load state to all clients. Using these information NFS clients decides their connection scenario.

III. Anatomy of Distributed Applications

A distributed application is built upon numerous layers. At the lowest stage, a network connects a group of host systems collectively so that they can speak to each other. Network protocols like TCP/IP permit the systems send data to each other over the network by providing the facility to wrap up and address data for delivery to another machine. Higher-level services are defined on top of the network protocol, for example directory services and security protocols. To finish, the distributed application runs on top of all these layers, using the mid-level services and network protocols in addition to the computer operating systems to carry out coordinated tasks across the network. A distributed application can be divided into Processes, threads, objects and agents [12].

- Process is created by describing a progression of steps in a programming language, compiling the program into an executable form, and run it in the operating system.
- Every process has at least one thread of control. Some operating systems hold up the creation of multiple threads of control inside a single process.
- Programs written in object-oriented programming languages are made up of objects.
- Agent is a higher-level system component, defined around a particular function or utility.

In [12], [13], a java programming language is used for distributed computing. It is because that java provides some functionalities that other object oriented programming languages can not. Java is concerned with simplicity, reliability and architecture neutrality. Such functionalities includes

Java is a pure OO language, having support for abstract interfaces, platform independence, fault tolerance through exception handling, network communication, better security using two dimensions i.e. secure runtime environment and secure remote transactions. Java also has the capability of multi threading. So it means that java is a better approach to develop distributed applications.

IV. Existing Systems

NFS was developed at a time when we weren't able to share our drives like we are able to today in the Windows environment. It offers the ability to share the hard disk space of a big server with many smaller clients. Again, this is a client/server environment. While this seems like a standard service to offer, it was not always like this. In the past, clients and servers were unable to share their disk space.

AFS is another way to move files around the Internet. It is a distributed file system that allows hosts to share files across local area networks--and bigger networks, such as the Internet. AFS, also known as "Andrew File System," was originally developed at the Information Technology Center at Carnegie-Mellon University.

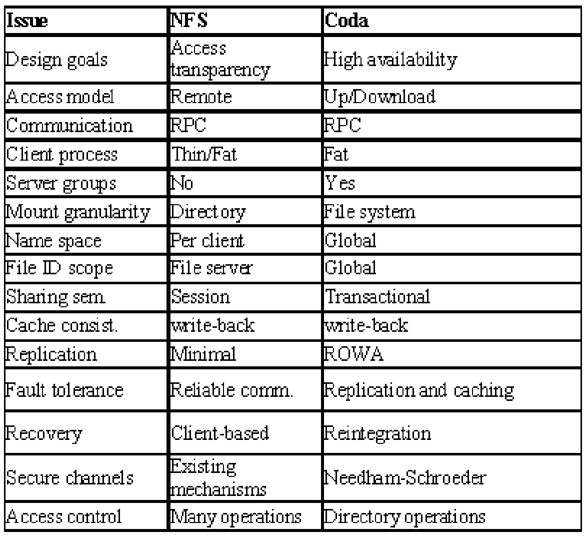
Coda is a distributed file system with its origin in AFS2. It has many features that are very desirable for network file systems. Currently, Coda has several features not found elsewhere.

- disconnected operation for mobile computing
- is freely available under a liberal license
- high performance through client side persistent caching
- server replication
- security model for authentication, encryption and access control
- continued operation during partial network failures in server network
- network bandwidth adaptation
- good scalability
- well defined semantics of sharing, even in the presence of network failures

We study different distributed or in other words network file systems, like NFS, XFS, LFS, SFS and Coda file system. A comparison made in [7] is given in Fig. 3 below.

The major issue in distributed systems is of security. In [8], [10], the authors have briefly described these issues. In [7], [11], the authors have discussed file services types, replication, consistency semantics for file sharing, multicasting, replica management, load balancing i.e. load management, load distribution, and group management.

Another major issue in distributed systems that makes these systems a little bit unpractical is inconsistency when replication techniques are used for achieving high availability and Quality of Service. A file that is shared over the distributed system, is opened by more than one user for editing, or if a file is updated recently, is there any surety that a recent update is visible to another user? So replication provides for HA but it has its own shortcomings, which we have not included in this paper. In next section we talk about our proposed scheme



Issue	NFS	Coda
Design goals	Access transparency	High availability
Access model	Remote	Up/Download
Communication	RPC	RPC
Client process	Thin/Fat	Fat
Server groups	No	Yes
Mount granularity	Directory	File system
Name space	Per client	Global
File ID scope	File server	Global
Sharing sem.	Session	Transactional
Cache consist.	write-back	write-back
Replication	Minimal	ROWA
Fault tolerance	Reliable comm.	Replication and caching
Recovery	Client-based	Reintegration
Secure channels	Existing mechanisms	Needham-Schroeder
Access control	Many operations	Directory operations

Fig. 3. Comparison between NFS & CODA

V. Proposed Systems

In this paper we have proposed a new idea for achieving efficiency, scalability and high availability in distributed systems. Below diagram in Fig. 4 shows our System Architecture.

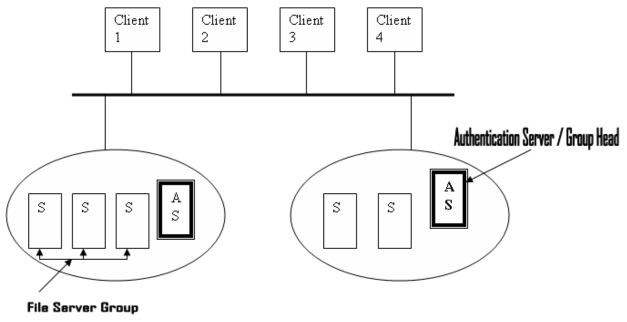


Fig. 4. System architecture

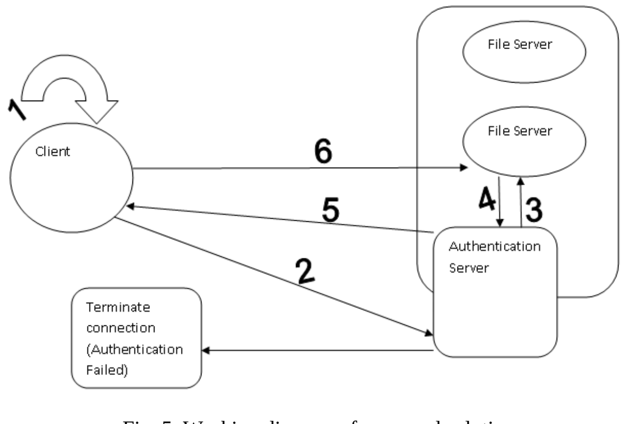


Fig. 5. Working diagram of proposed solution

In the proposed idea, a group header is responsible for authentication process i.e. makes the design more secure and it also works as a controller for the specified group. It means that before accessing a specific file server, AS will authenticate the client and will check for the file requested by the client. If the requested file is there on any file server, mounting process as defined in [1], [6], will take hold, otherwise connection terminate.

The idea works fine where high security is required. The system is efficient because a client will not try invaluable searching. The group header maintains record of all replication servers and files they have in their secondary storage. Cache mechanism on Group Header will improve our idea, in case a file server is down, but still client may access these from GH. Our proposed system runs in following steps.

1) The client generates an I_fHandle for mount point plus it's ID for authentication.
2) The client sends the mount plus authentication request, carried I_fHandle, to Group Header/AS.
3) AS authenticate the client, check its requested file in record database, If the file is in its own cache the request is handled, otherwise select the file server having the requested file. The mountd process in server sends I_fHandle to file server.
4) The mountd process replies "OK" to AS.
5) This I_fHandle is handed over by AS to NFS client, with a request to make RPC call to specific file server.
6) The client issue RPC calls to file server. The connection is established.

For load balancing technique the AS is responsible to count the total number of connected clients to a file server. So there is no need that all file servers in the group must advertise their state periodically as in [1]. Our proposed idea is efficient reducing network bandwidth for such advertisements. When a client request comes to AS, AS will decide keeping connection states of all file servers, and will response to the clients, to request another server group.

Our proposed solution provides the following benefits and a single shortcoming i.e. what happens in case of failure of a specific Group Header (GH).

- Local Security Policy
- Little computation as compared to Global security policy
- User accesses AS / GH, for authenticated & authorization check
- Performance Scalability + QoS + load balancing
- No need for individual node to check the user identity
- No Single point of failure, affect some part of the Distributed System
- Local & Quick allocation of resources by AS / GH
- Headache of AS, that are required to inform all corresponding AS in case of new node to any group community

VI. Conclusion and Future Work

The file server is a key factor to accomplish the data sharing essential in distributed systems. The file server is perhaps the most heavily used resource of the distributed systems and as an outcome; its performance is vital and critical to the victory of the system. The explosive growth of the Web contents and Internet users has led to increasing attention on two major challenges in these systems: scalability and high availability of network file system. A simple load-sharing mechanism for the NFS client to switch to a lightly-load server based on the number of NFS client's RPC requests for a period of time, making these systems more efficient and scalable. The replication techniques are used for improving the availability of Distributed systems in FSG. Our proposed solution is more secure & efficient in the sense that a client can not search the whole group for a specific file, because the group header maintains a huge database of all the group members along with their specific services they offers. Also load balancing mechanism is improved where the group header keeps a record of connected users to a specific server. Every group member is not required to advertise their state.

Our future work is to solve the main issue concerning to this proposed idea i.e. what about if AS fails or is temporarily unavailable? In that case the whole network file system will be offline.

References

[1] F. J. Liu, C. S. Yang, and Y. K. Lee, “Achieving efficient pathname LOOKUP in file server group,” IEEE.

[2] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. V. Steen, “Replication for web hosting systems,” *ACM Journal.*

[3] M. M. Leboute and T. Weber, “A reliable distributed file system for UNIX based on NFS,” UFRGS, Brazil, *IFIP International Workshop on Dependable Computing and Its Applications (DCIA 98)* January 12 - 14, 1998, Johannesburg, South Africa.

[4] F. J. Liu and C. S.Yang, “THE DESIGN AND ANALYSIS OF A HIGHLY-AVAILABLE FILE SERVER GROUP,” *IEICE Transactions on Information and System*, vol. 86-E, no.11, pp. 2291-2299, 2003.

[5] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D.C.Steere, “Coda: A highly available file system for a distributed workstation environment,” *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 447-459, April 1990.

[6] C. S. Yang, S. S. B. Shi, and F. J. Liu, “The design and implementation of a reliable file server, *Newsletter of the Technical Committee on Distributed Processing*, summer 1997.

[7] A. S. Tanenbaum and M. V. Steen, *Distributed Systems Principles and Paradigms*, Prentice Hall of India New Delhi – 110001.

[8] E. Miller and D. Long, “Strong security for distributed file systems,” IEEE 2001, pp. 34 – 40.

[9] G. Couloris, J. Dollimore, and T. Kinberg, *Distributed Systems – Concepts and Design*, 4th Edition, Addison-Wesley, Pearson Education, UK, 2001.

[10] Distributed system security: issues, processes and solutions. [Online]. Available: http://www.researchandmarkets.com/reports/705910.

[11] F. J. Liu and C. S. Yang, “Design and Analysis of highly efficient file server group,” Dept of CS & Engg, National Sun Yat-Sen University.

[12] J. Farley. Java distributed computing. [Online]. Available: Catalog no http://oreilly.com/catalog/9781565922068.

[13] W. Stalling, “Data and computer communications,” Ch. 22, *Distributed Applications.*

Muhammad Zakary is working as lecturer in computer science department of Abdul Wali Khan University Mardan, Pakistan. He is a new researcher to the field of new emerging computing technologies like Grid, Cloud and Green Computing. He has done MS in Computer Science and is interested for a doctorate degree in computer engineering. Currently he is working on security issues in Grid and Cloud computing i.e. distributed systems. He is interested in implementing Cloud computing over bioinformatics. He is also working and interested to Green the Cloud, Energy Efficient Scheduling for Real-Time Systems and Smarter Power Grids.

Muazzam Ali Khattak is working as lecturer in computer science department of Abdul Wali Khan University Mardan, Pakistan. He is a new researcher to the field of distributed systems. He is given a fully funded overseas PhD scholarship from Abdul Wali Khan University. Currently he is pursuing his PhD from Brunel International University, London. He is also interested in Green Computing. He is the author of several research articles in high performance computing and bioinformatics. His research topic is Smart Grid technology.