Storage Aggregation over a Network

Anuya Welling Ankit Parekh
MIT College of Engineering
Pune,Maharashtra,India.

Pallak Jhawar,Pratik Munot
MIT College of Engineering
Pune,Maharashtra,India.

Amit Savyanavar
MIT College of Engineering
Pune,Maharashtra,India.

ABSTRACT

The most valuable asset of any organization is data. Should this data be lost, it can be a potential threat to any business. Any enterprise/ company cannot afford such business interruptions. Such companies normally use the client server model to store their data. In the client-server model, all the data is stored on a server and the clients access it. This system has its own disadvantages. The fundamental one being that the space on the client machines is not utilized to the fullest. In order to overcome this limitation, we propose a model in which server utilizes the available space on the client machines which can be used for backup. Thus we describe a novel method which stores the data on the clients rather than on the server. Hence the cost, risk of losing the data and dependency on the server is reduced as well as the client storage space is fully utilized. Thus optimum usage of storage space on the clients to improve the efficiency and reliability of data storage on the network is the main aim of the project.

Keywords

Client Server Model, Distributed Storage Network, Efficiency, Reliability, NFS, Mirroring

1.INTRODUCTION

The most valuable asset of any organization is data. Loss of data results in a significant loss of revenue. While the importance of data is based on its value to an organization, effort spent to create it, the costs involved and often the significance of the data/information is not realized until it is temporarily or permanently unavailable. Should this data be lost, it can be very hard to replace and can be potentially dangerous to your business. Small and medium enterprises cannot afford such business interruptions.

In this paper, we deal with the data storage techniques employed in most companies and discuss the limitations of these techniques. In section III we present a novel technique of data storage which involves use of existing infrastructure/resources for data storage.

1.1 CLIENT SERVER MODEL

Client/server systems [1] are constructed so that the data can reside on a central computer, known as a server, and be shared among several users. Users access the server through a client or server application .Client-server software architecture [2] is versatile and flexible in today's fast-changing IT landscape. It is modular in structure and relies on messaging services for communication between components. They were designed to improve flexibility, usability, scalability, and interoperability.

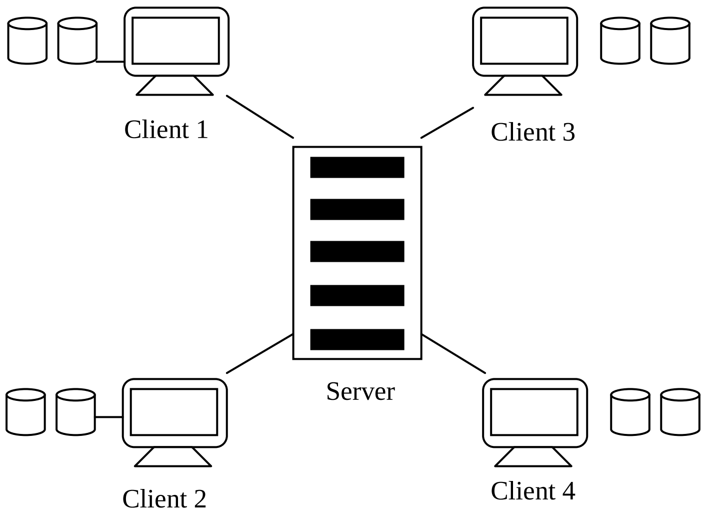

Figure 1 : Traditional Client-Server Model

Even though it is so beneficial, the client/server system has its own disadvantages.

DISADVANTAGES:

1. **Expensive**
Typically, the central server computer must be powerful enough to satisfy the storage requirements of all the client machines. This entails a substantial cost. Server costs [3] range from around 149$ to around 600$ if you want to really take advantage of their multiple processors and RAID configurations.

2. **Dependence**
The client-server network model relies on a functioning and available centralized server. If the centralized server is removed from the system or goes down due to problems, the entire network cannot function. Unfortunately if that does happen, all the data is lost. Thus there is high dependency on the central server.

3. **Wastage of space on the clients**
Let us take an example of an office environment where each client connected to a central server has disk space of around 500GB. Since the data is stored on the server, there may be a case where all of the 500 GB is never used. Only 100-200GB may be used. The remaining may be left unutilized.

1.2 NFS (Network File System)

1.2.1 Introduction

The Network file system protocol facilitates user or client to remotely access to common files from different networks. Portability, which is the key feature, is taken under consideration while developing the protocol. So the protocol is accessible from various types of network architectures, transport protocols as well as various operating systems and machines. The protocol permits a user on client side to right to use files across a network in such a way that whole file system is attached to the client's local machine. The protocol has developed in order to achieve portability; hence it can be used in various kinds of operating systems and machine architecture. NFS is application layer protocol. It is created on top of eXternal Data Representation (XDR). It uses Remote Procedure Call (RPC) primitives so as to attain portability.

There are four different versions of NFS protocol.

1. The **first version** was developed by Sun Microsystems for the research purpose, so it was not available to use publicly.
2. **Second version** of NFS protocol was implemented over UDP. RFC 1094 describes the second version of the protocol. The protocol is designed in such a manner that server need not to maintain protocol state of its client. Server does not keep previous request information. Instead, a client maintains all information required to send request to the server.
3. The **third version** was introduced in 1995 which defined in RFC 1813. The improvements over the second version are:
   - Provide 64 bits file sizes as well as offsets to facilitate large size file handling.
   - WRITEs can be asynchronous with the purpose of getting better performance.
   - Added more file attributes in several replies, to pass up the re-fetch.
   - Several procedures were added for consistency and for performance improvement.

   The third version is developed to support TCP at transport layer as the TCP has been used widely at transport layer over UDP.
4. The latest **version 4** is introduced in the end of year 2000 and it has been modified in year 2003 defined in RFC 3530. The fourth version of protocol is stateful unlike the previous versions. Features like file locking, complex and multiple operations, emphasis on strong security etc have been added with the intention of performance improvement and make the protocol work well over the internet surroundings.

In our model we use NFS version 3 since it is a stateless protocol.

2.PROPOSED SYSTEM

The proposed system as shown in Figure 3 aims at reducing this dependency for data on the central server and utilizing the space available on the nodes efficiently. Hard disk on the nodes can be partitioned into two parts as shown in Figure 2

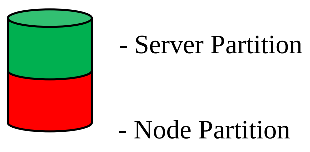

Figure 2 : Memory Partitioning on the Nodes

1. *Server partition*: This is the memory that can be used by the server
2. *Node partition*: This is the memory that can be used by the user which uses the node machine.

The user who uses the node machine does not have privileges to access the Server Partition. The server partitions from all the nodes are mounted on the server, so that the server now has access to these partitions. Server uses these mounted partitions for storage.

Whenever, server receives a file for storage, it will choose a location from all the available mounted partitions and store the file at that location. The server now needs to make an entry in its database, the location where the given file was stored. This database entry is used to retrieve the file whenever a request for the file is made.

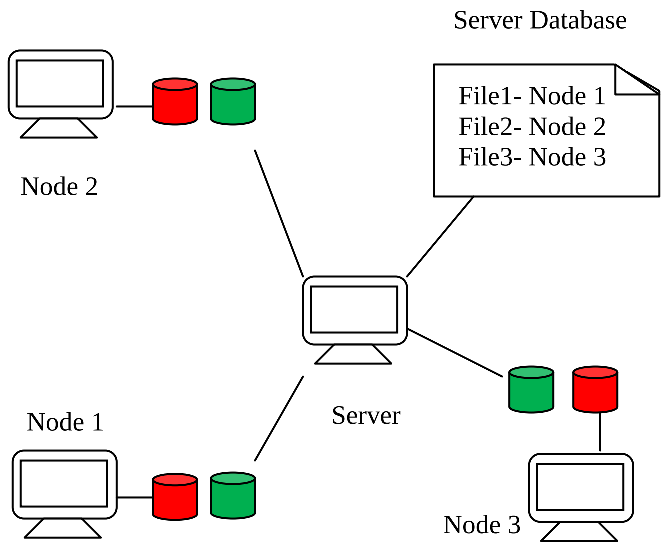

Figure 3 : Proposed System

For improving the system further, data can be mirrored at two locations i.e. data is stored at two locations instead of one location. By doing so, the data can be retrieved even if one of the nodes where the data is stored is unavailable. Also checksum of the file can be checked when storing and retrieving file to ensure data integrity. Thus the steps involved in storing and retrieving a file are as given below:

2.1 STORING A FILE

1. Start
2. File provided by the user for storage is copied to the server.
3. Checksum of the given file and file stored on the server are compared to ensure the data integrity of the file received.
4. Server searches for two available storage partitions. Available locations mean that the node machines are turned on and have enough space capacity to satisfy the storage requirements of the given file.
5. The file is copied from the server to each of the available locations.
6. Checksum of the file copied is validated to ensure that the file is stored correctly.

7. Server makes an entry in its database about the locations where a given file is stored. It also makes an entry of the checksum of the file.
8. An acknowledgement is given to the user about the storage of file and file is deleted from the server.
9. End

2.2 RETRIEVING A FILE

1. Start
2. The requested file is searched in the server database.
3. If file record is found in the database, then file is retrieved from any of the two available locations and provided to the user who requested the file.
4. End

3.ANALYSIS OF THE SYSTEM

The method explained above is cost-efficient as it uses the existing infrastructure, that is, the memory space on the client's. Also, even if the server crashes, data is not lost. Data is still present on the nodes. The locations where the files are actually stored are now unknown. Hence a backup of the database on the server needs to be taken regularly. Thus when server crashes, the backup of server database can be restored and smooth functioning can be ensured.

Table 1: Time to retrieve a File

File Size (Megabytes)	**Time Required in our model (seconds)**	**Time Required in client Server Model (seconds)**
100	38.55	29.11
200	60.50	52.81
300	90.59	79.68
400	122.51	100.23

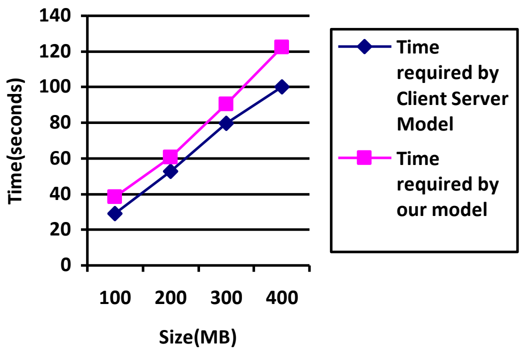

Figure 4: Comparison in time to retrieve file from server between client server model and our model

From Table 1 and Figure 4, we can observe the time to retrieve files of size 100MB, 200MB, 300MB and 400MB respectively. It can be seen that the time required to retrieve a file in our model is slightly more than the time required retrieving a file in a traditional client server model. However this is because the file is not retrieved directly from the server as in the case of the traditional client server model. It is retrieved from a third location specified by the server. Hence additional time is required. Also the time difference to retrieve the file between the two models is negligible.

Let us consider the scenario shown in figure 5. The maximum file size that we can fit in our case is 10 GB which is definitely a limitation.

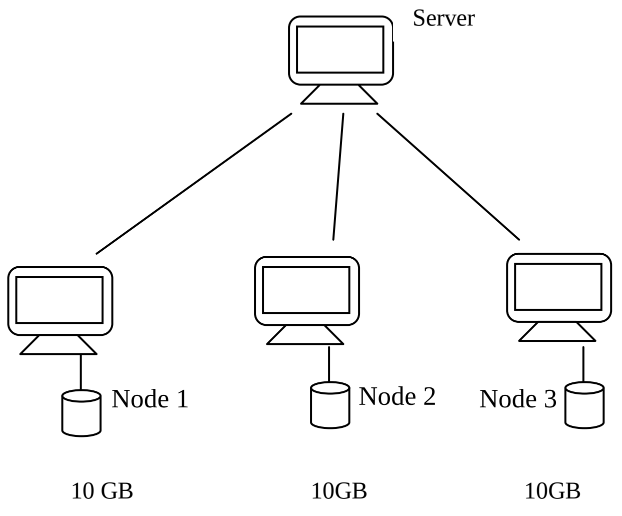

Figure 5: Limitation of the Proposed Model

4.OVERCOMING THE LIMITATIONS

To overcome this limitation, we can first group the available nodes into two groups: the nodes which actually store the data and the nodes which act as a backup to the first set of nodes. We then strip our file and store it on the nodes in each set. Thus in case a node from the first set is unavailable, the node from the second set completes the file. Figure 6 shows the enhanced model.

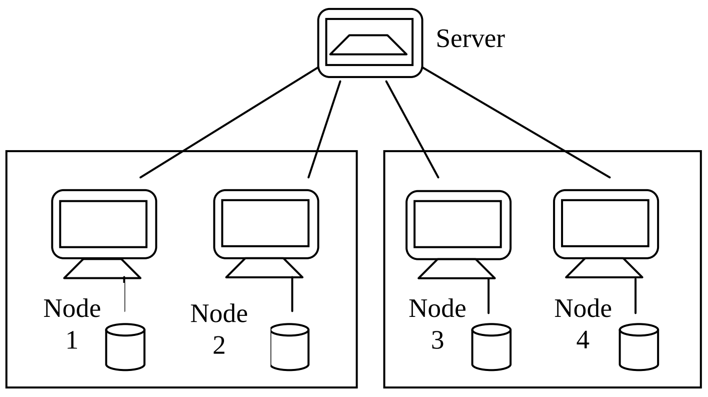

Figure 6: Enhanced Model

5.ADVANTAGES

The main advantage of the system includes the efficient utilization of resources. Also storage of data on multiple clients ensures uninterrupted availability of data; thus ensuring business continuity. At the same time mirroring reduces the dependency for data on a single node. Also the system can be implemented to work with heterogeneous operating systems such as UNIX, Windows, MAC, etc. In this way the storage solutions currently employed by enterprises can be replaced. Also in case of damage to the server, data is not lost. Only the database of which file is stored where is lost. This data must be backed up at regular intervals and can be restored in case of server failure.

6.DISADVANTAGES

The system may face performance issues if the server is not implemented properly. Server does the entire work of administration..Also there may be a case when all the nodes where data is stored are unavailable. In such a situation data cannot be provided to the user which requested the data. However the probability of occurrence of such a situation is very low.

7.CONCLUSION

The system if implemented properly has many features such as reliability and availability of data, low power consumption, no initial setup for a server, low maintenance and low cost. This proposed model for data storage can replace the enterprise storage solutions.

8.REFERENCES

[1] Microsoft technet. Chapter 7, client server architecture

[2] Client server architecture: the importance of flexibility in the modern world Author : Exforsys Inc. Published on: 30th Jun 2007The Basics if Reliable Distributed Storage Networks - Thomas C.Jepsen IT Pro May June 2004.

[3] Dedicated server school. How much does a dedicated server cost? http://www.serverschool.com/dedicated-servers/how-much-does-a-dedicated-server-cost/

[4] NFS version 3 design and implementation. Brian Pawlowski, Chet juszchak, peter staubach, carl smith, diane label, david hitz

[5] [Joy84a] Joy, Bill, “Sun Network File Protocol Design Considerations,” Internal Sun Microsystems technical note, January 1984. A description of the basic design principles in the NFS protocol, rationale and implementation, including the use of a reply cache for correctness

[6] Multiprotocol Data Access: NFS, CIFS, AND HTTP by Andy Watson, Paul Benn, and Alan G. Yoder, updated by H.T. Sun, Network Appliance, Inc. a technical report by netapp

[7] Juszczak, Chet, “Improving the Performance and Correctness of an NFS Server,” USENIX Conference Proceedings, USENIX Association, Berkeley, CA, January 1989, pages 53-63. Describes a server reply cache implementation for work avoidance. Listed as a side-effect,the reply cache avoids the destructive re-application of nonidempotent operations—improving correctness

[8] Prof. Richard Sinn,a technical report on ”Network File System Protocol”, San Jose State University,Network Architecture & Protocol,CMPE 208, Fall 2007.