Improvement in Capacity and Efficiency of Network Storage by Configuring Hard Disk Drives on Nodes of a LAN

K.N.Honwadkar
Asst.Prof. (IT)
D.Y.Patil College of Engineering
Sector 29, PCNTDC, Akurdi
Pune 411 044, Maharashtra, India

Dr.T.R.Sontakke
Principal
Siddhant College of Engineering
Sudumbare
Pune 412 109, Maharashtra, India

ABSTRACT

Storage on Network has, always, been a key feature of the success of the network design. Various methods are in practice. Two techniques are suggested in this paper. Disk Clustering and a technique to have Uniform namespace KINDFS is a distributed file storage system designed to provide cost-effective storage service utilizing idle disk space on workstation clusters. The system responsibilities are evenly distributed across a group of collaborating workstations; the proposed architecture provides improved performance, reliability and scalability. Workstation uptime data varies from system to system. KINDFS prototype implementation and measurement of its performance is suggested. Preliminary results indicate that KINDFS performance is comparable to that of commonly used distributed file systems, such as NFS, Samba and Windows 2000 Server.

General Terms

Algorithms, Measurement, Performance, Design, Reliability.

Keywords

Distributed file system, metadata, SAN, NAS, NFS.

1. INTRODUCTION

Advancements in Computer Technology focus on the fulfillment of end users' needs. Increasing demands from the users have motivated the researchers to come up with excellent systems and components. Processor speed, Memory requirements, Storage etc. are few of the major concerns while developing computer systems. Data storage is one such area where pressing requirements have lead to very efficient storage devices and systems. We will focus on this very interesting aspect of any computer system.
During early days of computer system evolvement users had limited resources for data storage. Devices like punched cards, magnetic tapes and drums were a few popular system components. The bulky nature and complex working of these devices forced the developers to think about handy, faster and simpler schemes for data storage and handling. Floppy, Hard Disk, Compact Disk came to existence which served the purpose for long time. Up to 2000 few Gigabyte storage disks were in use.

Nowadays, storage capacity of 300+ GB is a common feature of latest desk- top as well as laptop computers. Data up to 700MB or 4GB can be stored on CDs and DVDs. Latest introduction of memory sticks or Pen drives and portable hard disks and Zip drives of capacity of the order of few Tens of Giga Bytes made it possible for the users to carry the data wherever and whenever needed.

The need of sharing of the information generated by users, perhaps the most important aspect of resource sharing, motivated the development of network systems.
Client-Server, Peer-to-Peer, Internet, VPNs exhibit different degrees of data handling capabilities, storage capacities and efficiency. New schemes were deployed to achieve large storage space with efficient access. Use of Redundant Array of Inexpensive Disks (RAID), Storage Area Networks (SANs) and Network Attached Storage (NAS) are the examples.

1.1 RAID:

A RAID [9] partitions a *stripe* of data into N-1 data blocks and parity block — the exclusive-OR of the corresponding bits of the data blocks. It stores each data and parity block on a different disk. The parallelism of a RAID's multiple disks provides high bandwidth, while its parity storage provides fault tolerance it can reconstruct the contents of a failed disk by taking the exclusive-OR of the remaining data blocks and the parity block [Patt88, Chen94]. RAIDs suffer from two limitations.

- Performance for small writes can be degraded by the overhead of parity management; if the system does not simultaneously overwrite all N-1 blocks of a stripe. Unfortunately, small writes are common in many environments [Bake91], and larger caches increase the percentage of writes in disk workload.
- Commercially available hardware RAID systems are significantly more expensive than non-RAID commodity disks because the commercial RAIDs add special-purpose hardware to compute parity.

1.2 SAN (Storage Area Network):

SANs are networked infrastructures designed to cater for the needs of an organization where large amount of data is to be stored and accessed from different locations within the organization. These infrastructures provide a flexible, high-performance and highly scalable storage environment. This is accomplished by enabling many direct connections between servers and storage devices such as disk storage systems. High-performance Fiber Channel switches and Fiber Channel network protocols ensure reliability and higher efficiency. One or more Fiber Channel switches provide the interconnectivity for the host servers and storage devices in a meshed topology referred to as a "SAN fabric". Sample SAN network is shown in the figure 1. Variety of networks based on various operating environments can share information through these implementations.

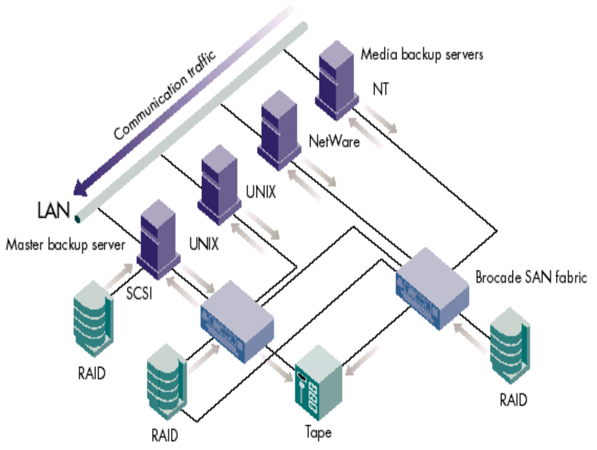

Fig 1 Example Storage Area Network

SANs are ideal for a wide variety of applications because they are optimized to transfer large blocks of data between servers and storage devices. Huge data storage requirements with considerably faster access demand these kinds of solutions. Organizations should strike a tradeoff between expenditure in implementation and availability and capacity of the data storage

1.3 NAS (Network Attached Storage):

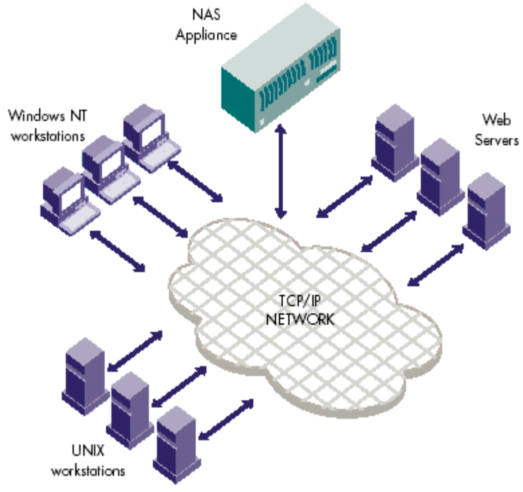

Fig 2 Example Network Attached Storage system

NAS [3] solutions are typically configured as file-serving appliances accessed by workstations and servers through a network protocol such as TCP/IP and applications such as Network File System (NFS) or Common Internet File System (CIFS) [10]for file access. Most NAS implementations mainly based around connections between workstation clients and the NAS file-sharing facility. NAS enables organizations to quickly and easily add file storage capacity to their technology infrastructure. NAS focuses specifically on serving files while hiding many of the details of the actual file system implementation, Small LAN/WAN network packet sizes force large transfers to be split into many small pieces. As the number of packets involved in the transfer increases the processor gets additional burden of splitting and reassembling of the data packets. This further degrades the overall system throughput as the processors cannot provide sufficient time for execution of multiple processes, already, running.

These implementations have advantages as well as some inherent disadvantages. On one side they are capable of storing huge data and allow the users to access the data in efficient way at the same time they are expensive and require additional hardware. Therefore, researchers started finding ways to make use of the hardware available with the computers / workstations connected in a network. Development of software solutions began with this motive. Software system developers also helped in efficiently accessing the data on a network. Depending the type of network, secure data handling is, looked upon as, the key to success of any new system introduced. This lead to what is known as ‘File System’ advancements and efforts were directed to handling of network distributed data.

Two different approaches were, primarily, thought over; Server Centric Storage (File Data) System and Server Intensive Computing. In the first way of implementations, Clusters of Servers are developed to cater for one or a few services to the network users and in the second, Server/s with sufficiently large storage are used for the execution of users’ applications. Linux Beowulf Cluster and LTSP or Ether boot programs are best examples of the two approaches respectively.

Table 1 SAN vs. NAS

Feature	SAN	NAS
Protocol	-Fiber Channel -Fiber Channel-to-SCSI	TCP / IP
Applications	-Mission-Critical transaction based database application processing -Centralized data backup -Disaster Memory Operations -Storage consolidation	-File sharing in NFS and CIFS -Small block data transfer over long distances -Limited read-only database access
Advantages	-High availability -Data transfer reliability -Traffic on primary network reduced -Configuration flexibility -High performance -High scalability -Centralized management -Multiple vendor offerings	-Few distance limitations -Simplified addition of file storage capacity -Easy deployment and maintenance

Information sharing is largely achieved on Internet, primarily by the use of web servers; but the requirements for sharing within local networks and intranets lead to different type of system. Recent design advances for distributed file systems have exploited the higher bandwidth connectivity of switched local networks and new modes of data organization on disk to achieve very high-performance, fault-tolerant and highly scalable file systems. The primary goal of these systems is to emulate the functionality of a non-distributed file system for client programs and data on a persistent storage scheme.

In many computer labs, there is large number of computers, connected in high bandwidth (e.g. 100Mbps) network, with unused drive space [1,2]. There may also be relatively large quantities of data to be stored. The users of the network system may have to store the data generated by their applications either on a portable storage device and carry it with them or use the same computer till the development is complete or the unused area on every hard drive may be used for storage of common required files. Sharing of these files can be achieved using some of the file system solutions discussed earlier. It is very much possible that the solution implemented may become an overhead expensive in execution time, availability, security of the data of individual user as well as performance against stand alone system, with respect to the user application. Therefore, it is necessary to have some system which will allow the users to make use of the unused space on every hard disk of the networked computer to store the data and be made available on any of the member computer of the local network.

It is, therefore, proposed to develop algorithms to improve the efficiency of existing system to achieve better performance as well as the capacity of the network storage. These algorithms should work concurrently with the system functions and implementation of any solution or environment. The currently available algorithms may also be modified to meet the requirements. One more way of achieving the same is by developing light weight system (set of algorithms). This system must also be compatible to any version of the operating system. The goal of this system for storing data distributed over many computers, with enough redundancy so that data can still be recovered if several of the machines are unavailable (due to inevitable hardware failure).A new approach of building a high performance file system is presented. The system is aimed at providing a simple and convenient way to achieve high aggregate I/O bandwidth. The system uses a hyper structure to integrate multiple file system services. It provides multiple data layouts to effectively distribute file data among network nodes. Performance evaluations demonstrate that the system has very good data access bandwidth with near perfect scalability, while still maintains an acceptable Meta data throughput.

Research is still on for finding solutions for the short comings of the currently available implementations of systems for sharing of network distributed data. Clusters are built for large data, normally, involved in database systems; distributed file systems are developed to take care of data handling in distributed environments. Internet has become a common place for storage of personal data which need not be carried everywhere one goes.

2. THE CONCEPT

It is proposed to develop a system (a set of algorithms) and to configure the Hard Disk Drives of client machines in a network into a cluster to enhance the capacity and efficiency of the network storage. Great amount of work has also been done for systems where data is distributed on a local network. Here, one such scheme is suggested for the data available on local network.

2.1 Disk Clustering :

The System provides a simple form of redundancy for data through a process called mirroring. This form typically requires two individual drives of similar capacity. One drive is the active drive and the secondary drive is the mirror. The local area network configured in disk clustering may be represented as shown in the fig 3 below. It is clear from the figure that the clustering and mirroring can be achieved with 'even' number of nodes in the network; although a few member nodes may be down at run time, due to any avoidable problem like hardware failure.

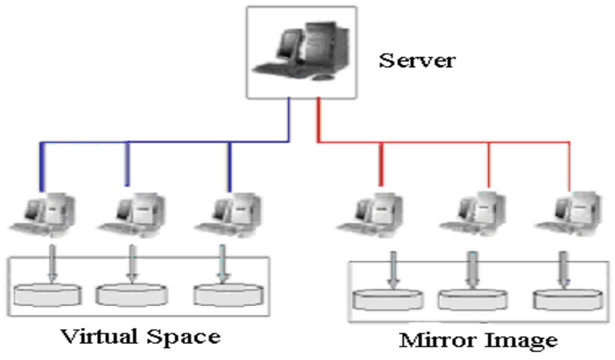

Fig 3 Disk Clustering Configuration

When data is written to the active drive, the same data is written to the mirror drive. The following is an example of how the data is written in a Disk Clustering implementation. Each row in the chart represents a physical block on the drive and each column is the individual drive. The numbers in the table represent the data blocks. Duplicate numbers indicate a duplicated data block.

	Drive 1	**Drive 2**
Block 1	Data1	Data1
Block 2	Data2	Data2
Block 3	Data3	Data3

This provides a full level of redundancy for the data on the system. If one of the drives fails, the other drive still has all the data that existed in the system. The big drawback of course is that the capacity of the RAID will only be as big as the smaller of the two drives, effectively halving the amount of storage capacity if the two drives were used independently. If stripping and mirroring is used, it then combines the methods of mirroring and striping to provide the performance and redundancy. The first set of drives will be active and have the data striped across them while the second set of drives will be a mirror of the data on the first two.

For example let us see how data is written in the implementation. Each row in the chart represents a physical block on the drive and each column is the individual drive. The numbers in the table represent the data blocks. Duplicate numbers indicate a duplicated data block.

	Drive 1	**Drive 2**	**Drive 3**	**Drive 4**
Block 1	Data1	Data2	Data 1	Data 2
Block 2	Data 3	Data 4	Data 3	Data 4
Block 3	Data 5	Data 6	Data 5	Data 6

In this case, the data blocks will be striped across the drives within each of the two sets while it is mirrors between the sets. This gives the increased performance of because it takes the drive half the time to write the data compared to a single drive and it provides redundancy. The major drawback of course is the cost. This implementation requires a minimum of 4 hard drives.

The main objective is to provide users with a comprehensive Data Management Service tool that fulfils the needs for Data network and Load management as well as a monitoring tool for prompting Crash system. With this objective in mind, the user is provided with a full featured package which includes RAID service, Service Analyzer as well as Load Management and shedding load through Clustering of Disk.

Table2: Upload and Download Time Measurement

Sr. No.	No. of Client Nodes on Cluster	File Size/ Type	Upload Time of file on Cluster	Download Time of file from Cluster
1.	4	1 MB (.dat file)	21 sec	2 sec
2.	3	5 MB (.mp3 file)	1 min, 45 sec	12 sec
3.	3	10 MB (.zip file)	3 min, 30 sec	25 sec
4.	3	20 MB (.exe file)	7min, 46sec	51sec

Advantages:
1. It requires no special storage devices it make use of disk space of client
2. Highly compatible with variety of file system (Linux/Windows)
3. Greater overall capacity and flexibility of a unified storage system and more efficient management of storage
4. Enhanced speed of data Access.
5. Greater efficiency in recovering from a disk failure (Disaster level recovery).
6. Redundancy of data Increases.
7. Providing Lower Cost Solution.
8. Increased Performance of the System

Disadvantages:
1. Suitable for small size networks
2. Database handling becomes tedious as number of files stored on the cluster increases
3. Files having smaller size also are split; increasing the overhead
4. Storage of Audio and Graphics files takes considerable time
5. Mirroring is very difficult in case variable size disk space is available on member nodes

There are instances where network storage is not required on regular basis. For example, small organizations may have a few stand alone computers connected in a network. The users store their data on local storage system. In case, some additional space is required a user may free some space from the storage system by moving some files to portable storage device or may delete some files. Disk Clustering would be very heavy solution. Therefore, some facility should be provided to have some additional space as and when required. The files can be stored and retrieved from this area. This space must be restored if the need be. This may be thought of as "storage space on demand".

2.2 KINDFS:(K(c)luster IntegratedNetwork Distributed File Storage (System)) :

2.2.1 Introduction to KINDFS

Many organizations have deployed desk top personal computers connected in a network for their regular operations and sharing of resources. Similarly, with the Internet exploding in size and reaching into every walk of life, digital data stored on-line are growing at an unprecedented rate. As a result, many organizations are under continuous pressure to expand their storage systems as demands for their services grow and their data sets swell relentlessly. Recent technological innovations ranging from faster peripheral channel, to dedicated storage area networks *(SAN)*, finally to aggressively specialized storage systems using specialized hardware and software have provided solution to the problem of providing large storage space. Higher costs of these highly specialized storage systems, more often than not, makes it difficult for many organizations to adjust budget on storage systems. At the same time, the computer industry has made significant advances in magnetic recording technology, with the cost of disk drives reduced due to mass production of disk drives. The standard disk capacity on mainstream computers is about 20-30GB as of mid-2001 and is growing continuously over time. However, most users prefer the network storage for various reasons:

a) Mobility - the users want to access the data through consistent interface from any place;

b) Quality of Service - normally the network storage is provided by high-performance highly-available storage systems with built-in redundancy and regular backup schedules;

c) Security - system security is much easier to maintain on a centralized storage system managed by professional administrators than on a decentralized system managed by individual users.

As a result of this, most of the local disk space on client workstations is only used for operating systems, application programs and temporary files, which in total take up only 2 to 4GB disk space. Douceur and Bolosky measured and analyzed a large set of client machines in a large commercial environment with more than 60,000 desktop personal computers. The measurement includes disk usage and content. The result shows that about 53% and 50% of the overall disk space of the studied environment is in use in September 1998 and August 1999, respectively. The disparity of space utilization ratios on storage servers and local machines is expected to deteriorate further over the time **as** the average disk size grows rapidly. This led to a design and deployment process of various implementations of utilizing idle disk space on workstation clusters. Using network attached storage (NAS) approach [3], NFS **[4]** by Sun Microsystems provide some way of sharing files on networked work stations**.**

The Network File System NFS can't provide enough bandwidth as it is based on single server functionality. In the applications like implementations on LAN the complexity of deployment and maintenance of Parallel file systems like GPFS or Lustre is too high. In such cases, a simple, yet powerful enough distributed file system is needed. A simple way of integrating several standard services together to build a high performance distributed file system with single namespace and aggregated data read and write bandwidth is presented.

This is **K**(c) luster **I**ntegrated **N**etwork **D**istributed **F**ile **S**torage (**S**ystem) and **KINDFS** for short.

The KINDFS is a kind of hyper file system. It stores its metadata and file data as files on the standard server and operates upon them on member nodes. Here the unused storage capacity of the member nodes is configured to add to the total storage space of the file system. This may lead to virtual storage volume available for file storage and redundancy of the storage over entire network. The system will be best suited for intranet applications on highly available data.

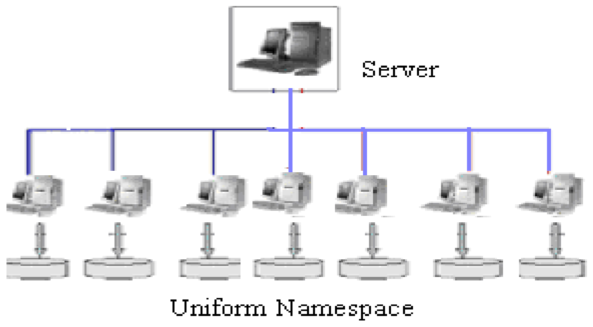

Fig. 4 : Global View of KINDFS

Fig. 4 shows a global view of server and clients. The configuration shown in the figure uses only one server and number of client nodes. Every member node has its mirrored counterpart in the same network. File data written to both the nodes is the same with the same attributes. When a node boots up and joins the network, it compares its KINDFS space with the mirror node and modifies the contents, if required. This leads to consistency in both the images of the file data. Since, a pair of nodes stores every user file on the common uniform namespace; at least, two copies of any file are available in the system at any working instance.

KINDFS architecture is designed to offer a *file* interface with built-in fault-tolerance to achieve reliability and high availability. All the members of the network share responsibilities evenly distributed to them. KINDFS can sustain balanced and scalable performance.

The scheme presents the following contributions:

 KINDFS is one of the better approaches to integration of multiple services to achieve file system which enables the access of data distributed over the local network.

 It proposes some techniques for improvement in the storage efficiency and capacity of the uniform storage space.

 It also proposes processes for evaluation of the performance of KINDFS.

 The resultant KINDFS will, certainly, have good performance and scalability.

 The development and deployment of the KINDFS is easier and simpler for use with other such services.

2.2.2 KINDFS Architecture

Fig. 5 shows a general structure of the KINDFS. The KINDFS is a kind of hyper file system, which is built upon single NFS server. The KINDFS store its meta-data on every member node and file data on respective node/s. The KINDFS on the client node follows the VFS interface. It can be mounted to a directory like ordinary file systems. Virtualization in File storage can be realized by dynamically mounting the users network storage at the time of 'login'. This will enable the user to access the files stored by the user only. The files owned by other users can only be read. This will add to the security to the files stored on the network storage .

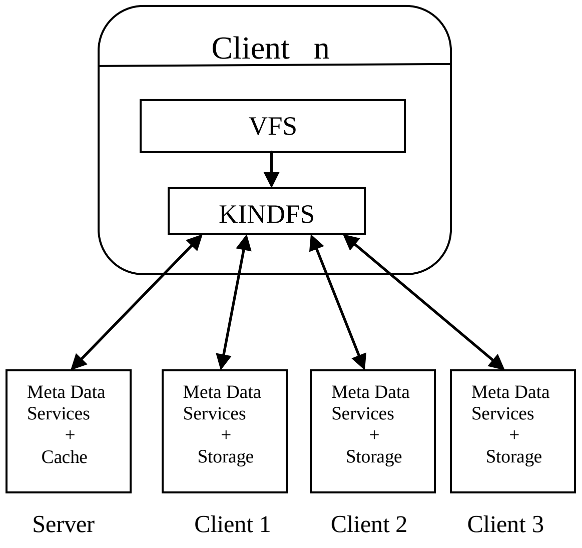
Client n
VFS
KINDFS
Meta Data Services + Cache
Meta Data Services + Storage
Meta Data Services + Storage
Meta Data Services + Storage
Server
Client 1
Client 2
Client 3

Fig 5. General structure of the Integrated Network Distributed File Storage -- KINDFS –

Both the KINDFS and its underlying meta file system have the identical directory and file name structure. As Figure shown below describes, for example if we have a file named file1 under the directory /subdir3, then we also have a file named file1 under the same position in the meta file system where the subdir3 is mounted as shown at /subdir3. The 'subdir3' in this example will be available on every member node and will have the same file system 'KINDFS' mounted on it. This will make the path for 'file1' as '/subdir3/file1' anywhere in the entire network. The file, if moved, from its location at the time of creation, to any member node of the KINDFS network to the same directory 'subdir3', the path remains the same. Here, 'subdir3' is the KINDFS area available on every member node and it is globally readable as well as writable. This is depicted in fig 6. The network administrator has to configure this space at the installation time. Every node in the network becomes client and server for this specific storage space.

Since the metadata is stored on the meta File server, and the meta File server hands over the same to the user at login time. When a file is uploaded the meta data of the file is decided at the node where the user has logged on and copied to the node where the file is stored. An entry is made in the meta file which is moved to the meta server at the time when the user logs out.

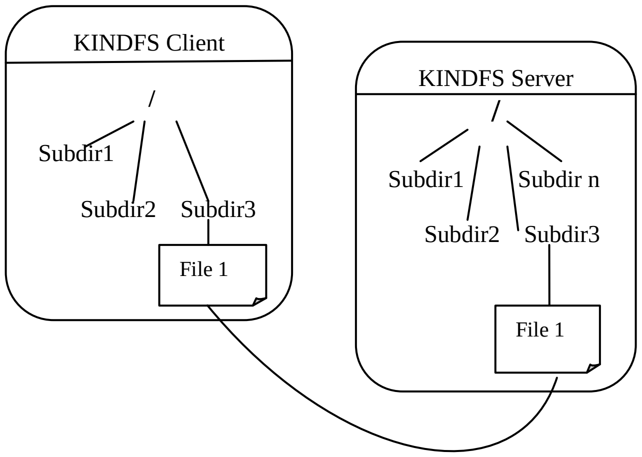

Fig 6: KINDFS has similar directory structure

Thus the single global namespace of KINDFS is maintained. The attributes of files in the KINDFS are stored as the attributes of the files in meta file system, including file name, creation and modification time, uid, gid etc. Here, we present a typical metadata operation by the example of file creation. When a file in the KINDFS is created and stored on the KINDFS space following process is used.

1. Create a file on the client node.
2. The file attributes and access rights are set.
3. Copy all the attributes from the inode of the meta file to the inode of KINDFS file, and associate these two inode structures by a pointer in KINDFS's meta file.
4. The file is stored on the local KINDFS space and to a node which is a pair node of it.
5. Finish other housekeeping work that a normal file system must do on creating a file.

From the previous process, we can see that the inode and dentry structure of the KINDFS is simply a wrapper of the according data structure of the meta file. This makes it easier for the server to move any file from any location to other location keeping the path same, when it is trying to balance the load or any other housekeeping operation. This clearly indicates that the KINDFS storage space on the network uses flat storage system i.e. no sub directory is supported. The virtualization is achieved by the meta file which maps the path of the file to correct KINDFS space on the respective node.

2.2.3 Bandwidth and capacity integration:

The KINDFS is designed to integrate both the bandwidth and capacity of its member nodes. This is done by distributing data among member nodes. Since the KINDFS is using the specific disk space of its member nodes to store the file data, it provides the user a file system with the summary capacity of all the member nodes. When data is distributed on multiple KINDFS servers the concurrent accesses to these data are also distributed on multiple KINDFS servers, so by this way, the network and disk bandwidth of all the KINDFS servers can be integrated.

2.2.4 Observations:

Following are the observations of the time taken by KINDFS to upload and download variety of file types. The network used has the specifications as follows…

1. Four nodes having different processor and hardware specification are connected in a switched network.
2. Network bandwidth is 100 Mbps

3 Linux operating system with different distributions but same version of kernel are installed on these member nodes with same '/data' as the KINDFS storage space name
4 Capacity of the uniform namespace viz. /data is different on every member node.
5 The system is implemented as an application running on the member nodes
6 Same users are created on all nodes; although different users login at the time of testing
7 Every member node acts as a server as well as client, so far as the network storage is concerned
8 Write operation (upload) takes longer time since two copies of the same file are stored on two different nodes
9 Read operation (download) requires less time , since the first available copy of the file is downloaded
10 The availability of the file stored is very high provided the failure rate of the member nodes is kept very low

Table 3 : Upload and Download Time Measurement

Sr. No	No. of Nodes in the system	Node Status	File Size/ Type	Upload Time of file on KINDFS	Download Time of file from KINDFS
1.	4	Running (2Appln)	1 MB (.dat file)	11 sec	2 sec
2.	3	Running (3Appln)	5 MB (mp3file)	45 sec	12 sec
3.	4	Running (1Appln)	10 MB (.zip file)	2 min, 20 sec	25 sec
4.	4	Running (2Appln)	20 MB (.pdf file)	3min, 46sec	51sec

3. RELATED WORK

The most relevant work is Microsoft Research's Farsite project [5**]**, which builds **a** serverless distributed file system with a large group of collaborative desktop computers.. Fault-tolerance of Farsite is achieved through replication of files assisted by Byzantine-fault-tolerant algorithm. The distributed RAID approach used in xFS [8] and Petal [7] was applied to build a virtual disk with block interface. On top the virtual block devices, higher level file systems and distributed file systems were built, like the metadata manager in xFS and Frangipani**.** KINDFS evenly distributes metadata and the management across all storage devices in the cluster. Metadata in KINDFS systems are stored at fixed locations while data can be stored anywhere, compared to the "Anything, Anywhere" file layout in xFS**.** KINDFS architecture bears close resemblance to Network-Attached Secure Disks (NASD) [6**],** which targets at high performance storage systems based on directly network-attached secure disks. The main difference is that in KINDFS file namespace consistency is maintained collaboratively by participating cluster members, different from NASD whose central file manager maintains the namespace consistency for the system.

4. CONCLUSIONS and FUTURE WORK

Efficient utilization of the free space on the nodes connected in a LAN can be achieved. This space may have been wasted, otherwise. Files with larger size can be stripped into as many as the number of member nodes in the network and can be evenly distributed over entire network where 'Storage Disks of the network will be Clustered' to have large storage space. This will facilitate to have uniform utilization of the network storage space. Splitting any file is a tedious task in itself and if we have to split the file in variable number of fragments it becomes an overhead for storing and handling the metadata into the database. Decision of size of the files to be split is added task which adds to the upload time. It cannot be implemented to have every file split before upload; since it becomes a huge overhead in case of files having smaller sizes. At the same time the configuration of 'disk clusters' allows the usage of the network drives to get a low cost space for distributed storage.

The requirement of uniform name space is achieved by the implementation of KINDFS. This cost effective solution provides network distributed storage space which is sum of all the free space available on all the member nodes (divided by two to take care of redundant copy of the file stored). The redundancy makes the file stored on this space highly available as the chances of both the disks where a file and it's copy are stored fail at the same time. The meta data handling shifted to the client end makes the file access faster. The elimination of any database implementation and handling makes the system more efficient as need of database maintenance is entirely removed from the usage of network distributed storage. It also extends the security features inherent to the Operating system implementations to the user files stored on the KINDFS storage space.

5. REFERENCES

[l] J. R. Douceur, and W. J. Bolosky. "A Large-Scale Study of File System Contents," in *Proceedings of the 1999 Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)***,** May 1999.

[2] W. J. Bolosky, J. R. Douceur, et al., "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PC**s,"** in *Proceedings of the 2000 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS),* June 2000.

[3] R. Gibson, and R. Van Meter, "Network Attached Storage Architecture," in *Communications of the ACM***,** vol. 13, Nov. 2000.

[4] Sun Microsystems Inc., NFS: Network File System

[5] Farsite project website: http://www.research.microsoft. com/research/sn/Farsite/

[6] G. A. Gibson, et al., "A Case for Network-Attached Secure Disks," TR-CMU-CS-96-142, Sept. 1996.

[7] E. Lee**,** and C. Thekkath, "Petal: Distributed virtual disks," in *Proceedings of the ACM 7th Intentational Conference on Architectural Support for Programming Languages and Operating Systems (ASPWS)***,** 1996.

[8] T. Anderson, M. Dahlin, et al.. "Serverless Network File Systems," *ACM Transactions on Computer Systems (TOSC),* Feb. 1995.

[9] D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Array of Inexpensive Disks (RAID)," in *Proceedings of the I988 ACM Conference on Management of Data (SIGMOD***).** Chicago, IL, June 1988.

[10] P. Leach, and D. Naik, "Common Intemet File System (CIFS/l .O) Protocol Preliminary Draft," Intemet-Draft Dec. 1997.

[11] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, R. Wang. Serverless Network File Systems. *15th SOSP*, p. 109-126, Dec 1995.

[12] Edward K. Lee and Chandrohan A. Thekkah. Petal: Distributed virtual discs. SIGPLAN Notices, 31(9):84-92,1-5 October 1996.