International Journal of Information Technology and Knowledge Management
July-December 2011, Volume 4, No. 2, pp. 659-667

CONFIGURING HARD DISK DRIVES ON NODES OF A LAN TO IMPROVE CAPACITY AND EFFICIENCY OF NETWORK STORAGE

K. N. Honwadkar[1] & T. R. Sontakke[2]

Storage on Network has, always, been a key feature of the success of the network design. Various methods are in practice. Three techniques are suggested in this paper. Disk Clustering, Universal Virtual Storage 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. The techniques suggested can be implemented on any of the currently used Operating systems with required modifications. They are tested on homogeneous environment on a LAN. They can be further developed for Heterogeneity.

Keywords: Distributed File System, Metadata, SAN, NAS, NFS, Algorithms, Measurement, Performance, Design, Reliability

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.

[1]D.Y.Patil College of Engineering, Akurdi, Pune 411 044, Maharashtra, India
[2]Principal, Siddhant College of Engineering, Sudumbare, Pune–412 109, Maharashtra, India
E-mail: [1]knhonwadkar@yahoo.co.in, [2]trsontakke@yahoo.com

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 SAN (Storage Area Network)

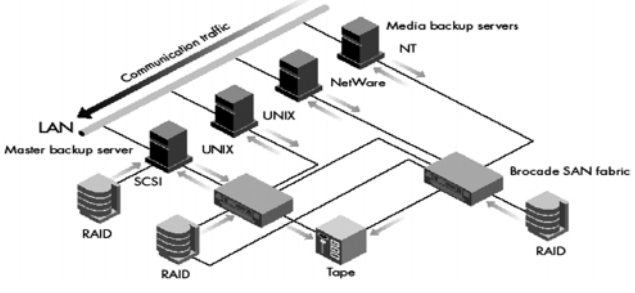

Fig. 1: Example Storage Area Network

SANs are networked infrastructures designed to provide a flexible, high-performance and highly scalable storage environment. SANs accomplish this by enabling many direct connections between servers and storage devices such as disk storage systems and tape libraries. High-performance Fiber Channel switches and Fiber Channel network protocols ensure that device connections are both reliable and efficient. These connections are based on either native Fiber Channel or SCSI (through a SCSI-to-Fiber Channel converter or gateway). One or more Fiber Channel switches—such as one of the Brocade® Silk Worm® family of switches—provides the interconnectivity for the host servers and storage devices in a meshed topology referred

to as a "SAN fabric". Because SANs are optimized to transfer large blocks of data between servers and storage devices, they are ideal for a wide variety of applications.

1.2 NAS (Network Attached Storage)

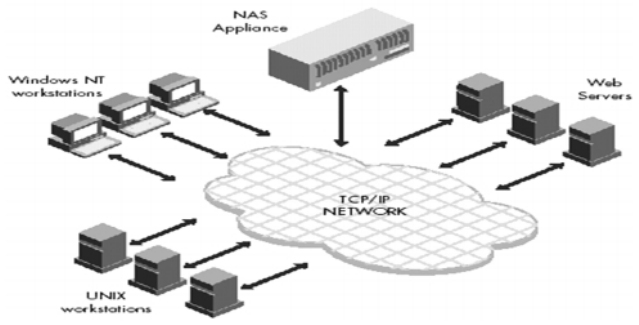

Fig. 2: Example Network Attached Storage System

NAS 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) []for file access. Most NAS connections reside between workstation clients and the NAS file-sharing facility. These connections rely on the underlying corporate network infrastructure to function properly. NAS enables organizations to quickly and easily add file storage capacity to their technology infrastructure. Because NAS focuses specifically on serving files while hiding many of the details of the actual file system implementation, NAS appliances are often self-contained and relatively easy to deploy. Typical interaction between a NAS client and an appliance involves data transfers of relatively short duration and volume. Today's small LAN/WAN network packet sizes force large transfers to be split into many small pieces. Additional processing is required at each end of the connection to break down and reassemble the data stream. As the number of packets involved in the transfer increases, so does the drain on the processor. Applications running on the same server might be adversely affected if packet processing consumes too many processor resources.

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.

Table 1
SAN Vs. NAS

	SAN	NAS
Protocol	• Fibre Channel • Fibre Channel-to-SCSI	• TCP/IP
Applications	• Mission-critical transaction-based database application processing • Centralized data backup • Disaster recovery 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 • Reduced traffic on the primary network • Configuration flexibility • High performance • High scalability • Centralized management • Multiple vendor offerings	• Few distance limitations • Simplified addition of file sharing capacity • Easy deployment and maintenance

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.

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.

2. PRESENT WORK

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, such schemes are suggested for the data available on local network.

Any network data storage system should have following functionality satisfying the basic needs of the users.

- Files or data should be smoothly uploaded and downloaded.
- Meta data of a file should be written once and followed later.
- Security of the data to be provided.
- Data or file must be made available as and when demanded by user.
- File of any size and type should be handled.

The functionality mentioned above leads to characterization of storage system as...

- Communication abilities
- Efficiency
- Security
- Availability
- Capacity

These characteristics can be described as.....

- Ability of the storage system to communicate with servers of peer nodes depends on the physical as well as logical connectivity and bandwidth.
- Meta data handling corresponds to setting of file attributes and its accessibility. Server centric systems are relieved from this burden as the meta data is handled by the server. On the other hand they are likely to get affected by the load on the storage servers. The system suggested tries to achieve good tradeoff between meta data handling and high throughput.
- Security of the file stored on the network storage is very important issue, since the file is visible to any network user, it must be protected against unauthorized handling.
- Basic idea of designing a network file storage system is to make the user file available to the same user or group of users from any member node. Factors affecting this are due to member node failures or heavily loaded node. Redundancy should be implemented by having more than one copies of the file stored.
- The users of the network might need to store files of any size and or any type. So far as the file type is not executable, other data files should be stored and retrieved from the storage system. Available storage area should be monitored for usable size and if file of greater size is to be uploaded then it should be loaded on other member node or nodes as the case may be.

2.1 Efficiency Consideration

Network storage system efficiency mainly depends upon the bandwidth available for communication, load on the network nodes involved and response of participating nodes for execution of the algorithms implemented as well as the access time of the storage device.

Nowadays, bandwidth used for LANs is normally 100 mbps. This bandwidth minimizes the communication time. The operating systems developed for multi-user, multitasking functionalities allow the users to execute multiple tasks at reasonably faster speeds. The hardware of the storage device used is reliable at the same time their access times are in the range of few microseconds or even few nanoseconds. Efficiency of network storage then solely depends on the efficiency of the algorithms implemented and configuration of the storage system. Configuration of the storage system should be such that it should not hamper the existing system implementations at the same time they will add to the ease of operation and better time complexity.

Algorithms implemented for the improvement in the efficiency of storage system should be executed concurrently with the existing system without adding to overheads of the system. These algorithms should take into account the modified configuration, if any as well as modifications in the operating system made to suit the improvement considerations.

2.2 Capacity Consideration

Capacity of network storage system corresponds to the space that can be made available for access from any member node of the network. Since network storage is being considered the effective storage space will be the addition of the shared space made available on all member nodes.

2.3 Efficiency Measurement

As discussed earlier efficiency of network storage system depends on bandwidth, load on individual node and execution time of algorithms along with the hardware specification of the nodes involved.

For a LAN setup the bandwidth available is more or less the same in all communications. Administrator can use the hardware with similar specifications of any two nodes.

Then the efficiency of the storage system will be proportional to the execution time and inversely proportional to load on member nodes.

Practical measurement of execution time of algorithms as well as load on a node is very difficult. Therefore, the time factor will be measured for complete transfer of a file under different load conditions. Lesser the time for file transfer more efficient the algorithms will be. The basic file operations can be considered for measurement of efficiency. The operations on files transferred to and from the network are 'upload', 'download' and 'delete'.

2.4 Data Security

Distributing multiple replicas of a file protects not only against accidental failure but also against malicious attack. To destroy a file, an adversary must compromise all machines that hold replicas of that file. To prevent an adversary from coercing the system into placing all replicas of a given file on a small set of machines under the adversary's control, the system must use secure methods for selecting machines to house file replicas. Since the system implementation is suggested on Linux, security features inherent to the operating system are used; without any additional algorithm for security.

2.5 Performance Analysis Methodology

To analyze the feasibility of deploying the proposed system on an existing computing infrastructure, usage data from desktop personal computers connected in network must be collected. Contents of file systems, availability of machines and load on machines are measured. Roughly half of all machines belong to technical developers; the other half is distributed approximately equally among the remaining categories.

2.6 File System Measurement

To determine the amount of disk space that is free and the amount of disk space that would be free if all duplicate files were eliminated the file systems should be analyzed and also the rate of cache misses, the size of local system caches, and the rate of file writes be estimated

2.6.1 Disk Usage Measurement

In September 1998, Microsoft employees run a scanning program on their Windows and Windows NT computers that collected directory information (file names, sizes & timestamps) from their file systems [7]. By this means, measurements of 10,568 file systems [8] can be obtained. In August 1999, study was conducted to remotely read the performance counters (free disk space, total disk space, and logon name of the primary user) of every Windows NT computer that could be accessed. By this means, measurements of 8669 file systems were obtained.

2.6.2 File Access Rate Estimation

The timestamps in the data set can be used to estimate the rate at which files are read and written. Each file's last read time determined the most recent timestamp (create, update, or access) on the file and its last write time as the most recent of the create and update timestamps. There are three aspects of file access rate: the miss rate of the local file cache, the size of the local file cache, and the amount of write traffic sent to file replicas on other machines. Only files that would be stored in the global file store were considered, excluding inherently local files, including the system paging file and those files in temporary directories or the Internet browser cache, all of which account for 2% of the bytes in the data set. Write traffic rate is a function of the lazy-update propagation lag. Write traffic rate is estimated as follows: For a propagation lag of n hours, files last written between 2n hours ago and n hours ago will have been propagated during the past n hours.

2.7 Machine Availability Measurement

To determine file availability in the proposed system, it is needed to know machine uptimes, whether these uptimes are consistent over time, and whether the uptimes of different machines are correlated. When the proposed system sees a machine go down, it needs to predict how long the machine will stay down. The reliability calculations require knowing machine lifetimes. Practically, no machine should be down for more than a few hours.

2.7.1 Machine Uptime Measurement

Patterson et al. analyzed the reliability of RAID disk arrays using queuing theory [15]. Similarly, the proposed architecture reliability depends on the reliability of each individual system whose idling disk space is utilized. We define the following notations for further discussion. For disks the Mean Time to Failure is denoted as Tmf and the Mean Time to Repair as Tmr. For disk arrays the Mean Time To Failure is Tmfg,. G is the number of data disks in a RAID group. C is the number of check disks in a group, where C is 1 in RAID level 1/4/5.

Therefore, Tmf is (G 4- c) * (G + c - 1) * TmRd With the latest disk manufacturing technologies, modem disk drives on client machines are very reliable. The main failing factor in this study would be the system availability of desktop workstations. Due to failures in operating system, or user's choice to reboot the system for whatever reasons, the mean time of client machine to come to up state is (Tmup) very small compared to TmMup, Mean up time of individual machine. Thus, it can be considered the TmMup of individual

disk drive infinitely large. However, when the system reboots, it normally can restore to working condition in 2 or 3 minutes, including grace time for all services to get started. This period of time for system to reboot is very small compared to the mean system uptime.

2.7.2 Machine Downtime Prediction Calculation

From the regular observation of the complete system, the administrator can find the time for which a machine is off in given span of time; which will help determine the machine down time for same as well as any period of time.

2.7.3 Machine Lifetime Measurement

Machines in the system can be periodically pinged and responses can be recorded. Scanning backwards through the data it can be determined the last time each machine responded to a ping. If machines have deterministic lifetimes, then the rate of attrition is constant and the count of remaining machines decays linearly. The expected machine lifetime (meaning the lifetime of the machine name, not the physical hardware) is the time until this count reaches zero. Since a LAN implementation is considered this time can be large enough to ignore the effect of machine downtime.

2.8 Machine Load Measurement

CPU and disk load is measured by running a program on some data-collection computers that iteratively selected a random machine and remotely read the performance counters (CPU load and disk load) of that machine. CPU load was measured as the fraction of cycles expended in processes other than the idle process. Disk load was measured as the number of disk operations performed per second.

2.9 Reliability Analysis

If machines always notify the system before being permanently decommissioned, overall file reliability is governed by the disk failure rate [16]. In particular, the likelihood of permanently losing a file is exponentiated by the replication factor. The mean time between loss of one machine full of replicas is equal to the mean machine lifetime, 'l'. A directory full of files will be permanently lost if all machines containing the other replicas of these files are decommissioned before the system recognizes the loss and creates new replicas. Given 'd' directories per machine, 'r' replicas and 'lag τ' between a machine's turnoff and the creation of replaced replicas.

KINDFS (K(c)luster Integrated Network Distributed File Storage (System)):

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 [14], NFS 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 Integrated NFS like Distributed File Storage (System)[KINDFS for short.]

The KINDFS is a kind of hyper file system. It stores its meta data and file data on the standard Node 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.

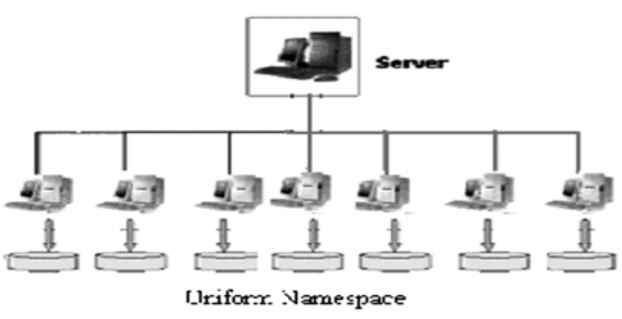

Fig. 3: Global View of the Server and Clients in KINDFS

Figure 3 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 storage 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.
- The resultant KINDFS certainly, has good performance and scalability.
- The development and deployment of the KINDFS is easier and simpler for use with other such services.

Global Name Space and Meta Data

Figure 4 shows a general structure of the KINDFS. The KINDFS is a kind of hyper file system, which is built upon single 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.

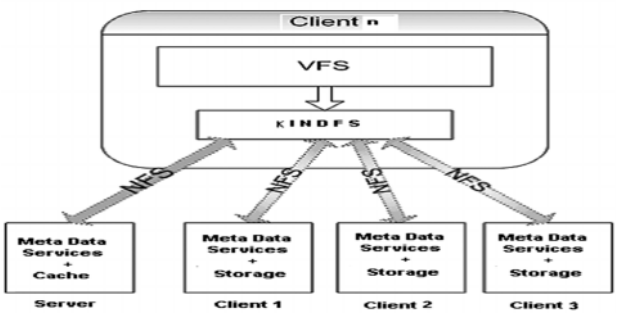

Fig. 4: General Structure of the K(c)luster Integrated Network Distributed File System

Both the KINDFS and its underlying file system have the identical directory and file name structure. As Figure 5 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.

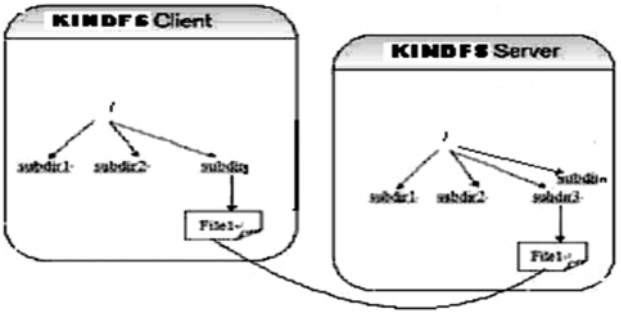

Fig. 5: KINDFS has Similar Directory Structure as its Meta File System

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. 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 meta data 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. 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 local flat file system of the member node, including file name, creation and modification time, uid, gid etc. Here, we present a typical meta data operation by the example of file creation.

When a file in the KINDFS is created and stored on the KINDFS space following process of creating a file is used.

1. Create a file on the client node.
2. The file attributes and access rights are set.
3. The file is stored on the local KINDFS space and to a node which is a pair node of it.
4. Finish other housekeeping work that a normal file system must do on creating a 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. In short the

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.

3. RESULTS AND 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…

- Four nodes having different processor and hardware specification are connected in a switched network.
- Network bandwidth is 100 Mbps
- 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
- Capacity of the uniform namespace viz. '/data' is different on every member node.
- The system is implemented as an application running on the member nodes
- Same users are created on all nodes; although different users login at the time of testing.

Since, every member node in KINDFS is both server and client at any operating time the algorithms are developed in such a way that they don't burden the servers. Location of any owned file demanded by the user is determined at the client end only. There is no need of mounting/unmounting of directories from remote node thereby reducing the time, otherwise, would have been spent in doing mounting/unmounting, either statically or dynamically. In addition to this KINDFS stores files on the uniform namespace with one more copy of each file, at any given time the file will be available to the user from any member node. The two copies of any file stored on the KINDFS storage space are stored on the two member nodes which are addressed with 'Paired Addressing' Technique. This makes it have files stored uniformly on the entire network.

Reading a File from KINDFS Space

1. Get the file name specified by the user in the command line
2. Check whether the file is available in the KINDFS space locally
3. If YES, open the file in read mode
4. If NO, open the file containing the information of the specific user files in KINDFS
5. Get the first location and send request to the specified node for 'file sending'
6. If the first node is unavailable, get the second node from the same file
7. Send request to send the file
8. If successful in either case the file will be read
9. If unsuccessful in both the tries, report error to the user

Table 2
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 (2 Appln)	1 MB (.dat file)	11 sec	2 sec
2.	3	Running (3 Appln)	5 MB (.mp3 file)	45 sec	12 sec
3.	4	Running (1 Appln)	10 MB (.zip file)	2 min, 20 sec	25 sec
4.	4	Running (2 Appln)	20 MB (.pdf file)	3min, 46sec	51sec

Table 3
Network Storage Systems at a glance

System / Feature	Petal	LanStore	UVS	Disk Clustering	KINDFS
Platform Supported	Unix	Windows	Linux/Windows	Windows	Linux
Metadata Storage Location	Server	Server	Server	Server	Client Node
Underlying Network	ATM	(TCP/IP)	(TCP/IP)	(TCP/IP)	(TCP/IP)
Level of Action	Block Level	File Level	N/W Node Level	File Level	File Level
Design Purpose	Redundancy and Mirroring	Checking Redundant Data	Data Storage	Redundancy and Mirroring	Uniform Namespace
Load on Server	Thick Server	Thick Server	Thin Server	Thick Server	Thin Server
Performance	Delay, as designed for server	Delay, due to vote system	Optimized for LAN	Delay, due to file stripping	Optimized for LAN
Applicability	Important Server Data Repository	Safe and Confidential Data Storage	Back up Data Repository within LAN	Load Balanced Storage space on LAN	Uniform Storage on LAN with sufficient availability
Scalability	Limited Scope	Limited Scope	Restricted by Loaning	Moderate Scope	Highly Scalable
Cost Effectiveness	Expensive	Expensive	No Additional Cost	No Additional Cost	No Additional Cost

4. CONCLUSION

For efficient utilization of the free space on the nodes connected in a LAN can be achieved. This space may have been wasted, otherwise.

The three techniques make use of different configurations and respective set of algorithms. For example 'Disk Clustering' uses the shared drives on the member nodes to store small parts of user file (Stripping Algorithm), 'UVS' also uses space on member nodes, with additional configurability of only a part of the member node shared drive which is spare by the administrator (Loaning Algorithm) and 'KINDFS' makes use of the extra space spared on member node configured as a global namespace. All the three techniques use the original file system of the Operating system loaded on the individual node (desktop), but utilize extra space on each member node of the LAN to make it a 'Network Storage Space'. In effect the storage space seen by individual user will be the locally available storage PLUS the network storage space. At the same time these techniques make the files, stored on the network storage space, available to the user from any member node; thereby, reducing the use of external storage devices while providing mobility to the users within the network area and optimum use of spare storage space on all the member nodes without any extra cost.

REFERENCES

[1] Eunsung Kim Hyeong S. Kim Heon Y. Yeom, "GiSK: Making Secure, Reliable and Scalable VO Repository Virtualizing Generic Disks in the Grid", In Eighth IEEE International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), Protocol Specification, RFC 1094. March 1989.

[2] H. Howie Huang, John F. Karpovich, and Andrew S. Grimshaw, "A Feasibility Study of a Virtual Storage System for Large Organizations", Second International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006).

[3] I. Clarke et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System", Workshop on Design Issues in Anonymity and Unobservability, 2000, pp. 46-66.

[4] B. Cohen, "Incentives Build Robustness in BitTorrent", Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, May 2003.

[5] Paul massiqlia and frank bunn, "Virtual Storage Redefined: Technology and Applications for Storage Virtualization", 2nd ed., VERITAS, 2000.

[6] FreeNet. http://freenet.sourceforge. net.

[7] D. Hitz, J. Lau, and M. Malcolm, "File Systems Design for an NFS File Server Appliance", In USENIX Winter 1994 Technical Conference Proceedings, Jan. 1994.

[8] R.Card, T. Ts'o, and S. Tweedie, "Design and Implementation of the Second Extended File System", In Proceedings of the First Dutch International Symposium on Linux, 1986.

[9] E.Riedel, "Storage Systems - not Just a Bunch of Disks Anymore", QUEUE, pp. 32–41, 2003.

[10] Vazhkudai, S., et al., "FreeLoader:Scavenging Desktop Storage Resources for Scientific Data", In Supercomputing 2005 (SC'05): Intel Conference on High Performance Computing, Networking and Storage. 2005.

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

[12] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead", 2003.

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

[14] J. Kubiatowicz et al, "OceanStore: An Architecture for Global-Scale Persistent Storage", ACM ASPLOS, 2000.

[15] 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.

[16] Seagate Technology, Inc. http://seagate.com.

[17] D. Hitz, J. Lau, and M. Malcolm, "File Systems Design for an NFS File Server Appliance", In USENIX Winter 1994 Technical Conference Proceedings, Jan. 1994.

[18] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-Peer File System", OSDI 2002.

[19] M. McKusick, et al., "A Fast File System for UNIX,"ACM Transaction of Computer Systems (TOSC). Aug. 1984.

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

[21] W. J. Bolosky, S. Corbin, D. Goebel, J. R. Douceur., "Single Instance Storage in Windows 2000", To Appear in 4th Usenix Windows System Symposium, Aug 2000.

[22] D. Solomon., "Inside Windows NT Second Edition, Microsoft Press, 1998.

[23] W. Vogels., "File System Usage in Windows NT 4.0. 17th SOSP", p. 93-109, Dec 1999.

[24] Farsite Project Website: http://www.research.microsoft.com/research/sn/Farsite/

[25] A. Rowstron and P. Druschel, "Storage Management And Caching In PAST, A Large-Scale, Persistent Peer-To-Peer Storage Utility", ACM SOSP, 2001.

[26] Seagate Technology, Inc. http://seagate.com.

[27] E. Lee, and C. Thekkath, "Petal: Distributed Virtual Disks", In Proceedings of the ACM 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPWS), 1996.

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

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

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

[31] Douceur, J.R. and W.J. Bolosky, "A Large-Scale Study of File-System Contents", In Proceedings of the International Conference on Measurement and Modeling of Computer Systems, 1999.

[32] J. Wylie, M. Bigrigg, J. Strunk, G. Ganger, H. Kiliccote, and P. Khosla, "Survivable Information Storage Systems", IEEE Computer, 33(8):61–68,August 2000.

[33] B. Callaghan, NFS Illustrated Addison Wesley, 2000.

[34] R. Card, T. Ts'o, and S. Tweedie, "Design and Implementation of the Second Extended File System", In Proceedings of the First Dutch International Symposium on Linux, 1986.

[35] 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.

[36] A. J. Peters, P. Saiz, and P. Buncic, "AliEnFS - A Linux File System for the AliEn Grid Services", 2003. in CHEP03.

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