United-FS: A Logical File System Providing a Single Image of Multiple Physical File Systems on NFS Server

Huan Chen[1,2], Yi Zhao[1,2], Jin Xiong[1], Jie Ma[1], and Ninghui Sun[1]

[1]National Research Center For Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, China
{huanchen, zhaoyi, xj, majie, snh}@ncic.ac.cn

[2]Graduate School of the Chinese Academy of Sciences, Beijing, 100039, China

Abstract

NFS is considered to be the bottleneck in cluster computing environment because of its limited resources and centralized data management. With the development of hardware, NFS server has more than one I/O channel, more storage space and more powerful CPU. In this paper, we describe the design and the implementation of a new logical file system called United-FS[*]*. It can make storage devices connected to multiple I/O channels work concurrently and cooperatively. It can be exported by NFS server to provide a single file system image to clients by hiding a variety of native file systems built on different type of storage devices. This paper also compares the United-FS with the Software RAID system both from theoretical analysis and experiments. The results show that United-FS is much more flexible and its performance is better than Software RAID in most cases.*

Key：file system, software RAID, I/O channel, NFS, Workload

1. Introduction

Earlier NFS server has limited hardware resources, so it becomes the bottleneck when larger amount of clients access the server simultaneously in cluster environment. Parallel file systems try to solve such problem by aggregating the I/O ability of multiple servers. Most parallel file systems[2,4,5] do their best to achieve good scalability and unlimited high aggregated I/O performance. However, in reality, the scale of cluster machine for most scientific computing applications is not so large. In many cases, middle scale cluster which contains tens to hundreds of nodes is preferable for applications such as oil employment computing, auto machine design and simulation, gene sequence test, etc. Large scale parallel file system is not suitable for middle and small cluster environment in that such file system needs more than one server, the management is complex, and the cost is high. Most important, as the number of server increases, the failure rate of the node increases too. The worst case is the whole file system may not be accessible if one node goes down.

NFS[1] is still considered to be the best choice in small and medium sized LANs. It emerged in 1990 and was designed around a central server model. The advantages of NFS are that users can access files just like using local disks and the files can be shared among users. But for I/O intensive applications, when more clients are added and files are accessed concurrently, the server immediately becomes a performance bottleneck because it is limited by its resources including CPU, storage and network.

With the development of computer hardware, the performance of CPU, network and disk increases greatly, the mainboard of the computer can support much more component devices and the speed of

[*] This work is supported by the Innovation Research Project of Chinese Academy of Sciences under the grant No. KGCX2-SW-116, Key Technologies for New Generation Clusters

1-4244-0910-1/07/$20.00 ©2007 IEEE.

system bus connected to different devices also increases. Because all the I/O channels and the network channels on the server are extended, the physical I/O bandwidth can be improved if all the channels work concurrently. For such hardware system, how to make good use of multiple channels and physical file systems to provide high NFS I/O bandwidth becomes an issue.

To solve this issue, we developed a logical file system called United-FS. It is a logical file system because it locates on the top of many physical file systems. The advantages of United-FS are:

- It can make all I/O channels work concurrently and cooperatively.
- It can make good use of the native file systems to manage the different type of storage devices
- It can provide the NFS clients with a single view of exported file system which manages multiple native file systems on NFS server.

In this paper, we present the design and implementation of United-FS, and we compare United-FS with software RAID both from the characteristic and performance aspects. The rest of this paper is organized as follows: Section 2 describes the architecture of our United-FS system. Section 3 discusses implementation details of our prototype. Section 4 analyzes the differences between United-FS and Software RAID techniques. Section 5 shows the experimental results of the performance comparison between United-FS and Software RAID.

2. Architecture

2.1. NFS Server Hardware Platform

In order to exploit the maximal performance of each component, especially to extend the I/O transport capability, we customize the mainboard of the server to support multiple I/O channels. There are three HT PCI-X Tunnels link to the dual core CPU, and each PCI-X bus has two PCI slots, so there are total twelve PCI slots. A PCI slot can support a dual channel SCSI card or a dual port gigabyte network interface card.

Such powerful server platform benefits us a lot. First, much more disks and network interface cards can be added to the system because of the extended system bus. Moreover, the cost of such server is much lower than multiple low-end servers while the performance is same.

2.2. United-FS — a Logical File System

As mentioned in the above section, the I/O server's hardware has multiple I/O channels which can support many I/O devices. But how to make good use of multiple channels and physical file systems to provide high NFS I/O bandwidth becomes the issue.

To solve this issue we developed a logical file system—United-FS, which manages the underlying physical file systems instead of developing a volume manager which manages the disk storage directly. The reason we develop the logical file system is that the complex disk block information can be managed by the physical file system, and different physical file system has its own characteristics of managing blocks. United-FS will decide where the file data should be stored so all the file system can work concurrently and cooperatively. By exporting United-FS, NFS clients can only see a single mount point, so all the physical file systems on NFS server is transparent to clients

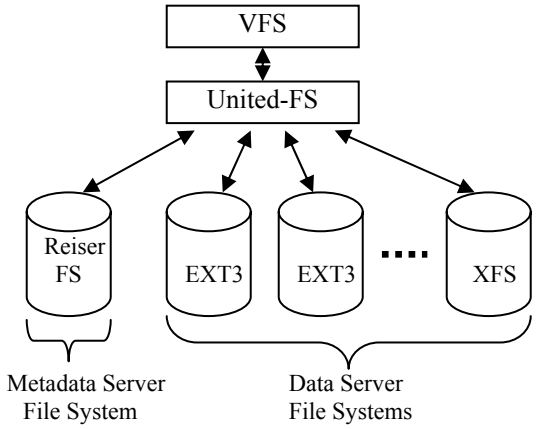

Fig. 1. United-FS Architectures

The architecture of the United-FS system is as shown in figure 1. On each device, we build a native file system, and on top of it is a logical file system. It manages the underlying native file systems and is exported by the NFS server. We classify the data to be stored in United-FS into two kinds: metadata and file data. We store metadata and file data in separate physical file system. The file system storing metadata is called Metadata Server File System, while other file systems storing real file data is called Data Server File System. The Metadata Server keeps the mapping information that guides the system to redirect the read/write requests to the right Data Server to be processed. By exporting the United-FS, users will see a single NFS mount point without knowing there are several file systems in service. Then all the Metadata Server and Data Server are transparent to NFS clients.

3. Implementation

In this section, we will describe the implementation of the metadata management and the real file data placement policies in details.

3.1. Metadata

Metadata contains the information that describes the location and extensive attributes of the file data. Metadata is important to the consistency and performance of the whole file system because before accessing the data, metadata information should be gotten first. If there is a mistake in metadata, the whole file data or even whole file system may not be accessible because one can not get the correct information to find where the data is stored. If accessing metadata is slow, it will affect the whole file system I/O performance severely for operations relative to accessing metadata take large scale in overall operations.

In United-FS, we use a single physical file system which we called MSFS (Metadata Server File System) to store the metadata of each file. MSFS manages the whole directory tree which is identical to the traditional file system. If it is a file, the file data is replaced with information about which data server to store the real file data and the strategies of storing the data (for example single mode or stripe mode). All the real file data are stored in Data Server File System.

The stability and reliability of centralized metadata management can be promised by some mature technologies. For example, metadata server can be constructed on top of hardware RAID which provides high data reliability.

The performance of processing metadata should also be taken into account. All the metadata about file data location information is stored in a small file that only needs dozens of bytes in the MSFS. According to such characteristics, we chose Reiser file system as our MSFS. The first reason is that Reiser file system is a journal file system, so metadata consistency can be maintained by itself. Second, based on our test, Reiserfs has a good performance in processing small files. Third, there is no pre-allocated inode number limitation in Reiser file system.

3.2. Data Placement Policy

Logical file system can have a global view of all the underlying physical file systems. The file system that stores the real file data is called DSFS (Date Server File System). Then, what file system should be used as DSFS? Where the data should be stored so as to make good use of the characteristics of these file systems?

In our system, we implemented two data placement policies: single policy, and stripe policy. Single policy means a file is stored in a single DSFS. Stripe policy means a file is stored in several DSFS in striped mode as RAID does. Users can choose different policy according to their workload.

(1) Single policy can be applied to the situation that many users access file system simultaneously. By dispatching the requests to different DSFS, the number of requests belonging to a single DSFS is decreased which will reduce the random movement of the disk head. All the file data are stored sequentially in a unique file system, so it does help to pre-fetch data of a file according to the pre-fetch algorithms. It is also good for the data reliability. When increasing the number of disks, the rate of disk failure will also be increased. If a DSFS breaks down, it will only affect the files on that particular file system while files on the other DSFS are still accessible.

Load balance problem should be paid attention to for the single policy because this policy will easily cause most active files to locate in the same DSFS. To solve this problem we implement two load balance methods. The first one is to dispatch the new file creation request on a neighbor DSFS of previous created file using Round Robin algorithm. The other method is to dispatch requests according to the user information of applications. Files belongings to the same user are placed in the same DSFS. The dispatch granularity of such case is not a single request but a batch of requests that relative to the same user.

(2) Stripe policy can be applied to the situation that the size of the requests is very large. A file is divided into N sub-files that reside on separate DSFS. Multiple DSFS then can process the request concurrently so as to improve a single file throughput.

For United-FS is designed for the NFS environment, we prefer single storage policy than stripe policy. Because from the result of our test, we found that the cost of stripe policy is higher than single policy. When heavy workload comes, the number of requests dispatched to the same disk does not reduce, so the disk head movement decreases the performance greatly. Another reason is the system kernel mechanism of writing dirty pages back causes that the concurrency of accessing a striped file is not as good as software RAID stripe mode.

4. Analysis Comparison between United-FS and Software RAID

The United-FS has several advantages over Software RAID.

First, United-FS interacts with physical file system interface, so it does not has to manage the location information of each disk block. All the work of managing block information is done by physical file system.

Second, it does not have to care about whether the device is SCSI device or IDE device or an array of RAID disks.

Third, United-FS has much more flexibility than Software RAID. Recently, there are many kinds of physical file systems and each of them has some particular characteristics suitable for some particular environment, so United-FS can take advantages of the characteristics of the underlying physical file system to improve the overall IO performance. Also, different users have different data reliability requirements. For example, for those who need high data reliability, United-FS can dispatch the files to the system build on the hardware RAID devices but Software RAID can not do this.

The last but not the least, Software RAID is at the level of device driver, so it can not view the information about the application request. All the information it knows is about data block. This limits the capability of scheduling the request in multi-application environment, while logical file system can get all the information about a request including file inode information, user id, offset and count etc. For this reason, it is much applicable to implement application request scheduling mechanism at the logical file system level. Table 1. shows the characteristic comparison between United-FS and Software RAID.

Table 1. Comparison between United-FS and Software RAID

	United-FS	Software RAID
Level	file system	driver in OS
Storage strategy	single、 stripe	RAID[0-5]
Granularity	file	stipe size (64k default)
CPU cost	lower	high
Scalability	good	good
Flexibility	good	no
Reliability	good	according to RAID level

5. Experimental comparison between United-FS and Software RAID

5.1. Data Placement Policy

All the experiments are conducted on a special NFS server running SUSE 10. The server has a dual-core CPU, 4 GB memory, 8 disks and 4 network cards. There are 128 commercial clients running SUSE 10 in our experimental system. The benchmark we choose is N-user IOzone. Each user writes its own file sequentially, and the filesize is 6GB.

In this section, we will compare the performance of United-FS with software RAID0 to show that United-FS performs better in multiple users system with heavy workload.

5.2. Disk Scalability

Figure 2 shows the IOzone sequential write performance comparison between United-FS and LVM driver configured with different physical file system when the number of disks increases. And figure 3 shows the read performance result. In this test, we use single data placement policy for United-FS while we use stripe mode for LVM. We don't use stripe data placement policy in United-FS because the system overhead is much higher than single data placement and the performance is not good in multiple users doing concurrent I/O operations environment.

For sequential write/read operation, XFS file system performs better than EXT3 file system due to the different data block organization. As the number of disks increases, the CPU usage also increases. EXT3 spends much more CPU than XFS, so when there are more disks, the CPU becomes the bottleneck that causes the un-scalable performance.

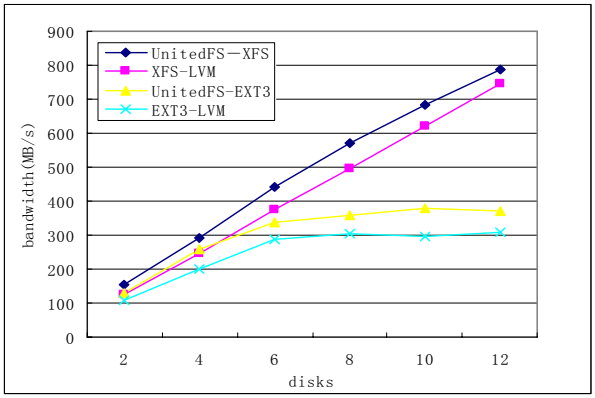

Fig. 2. Write Performance by Number of Disks

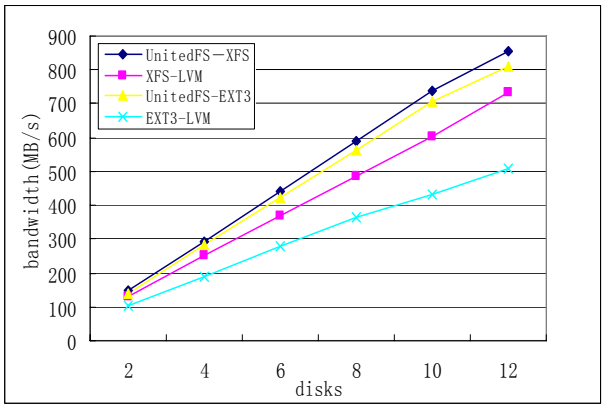

Fig. 3. Read Performance by Number of Disks

The result shows that United-FS performance scales linearly in the number of disks as LVM does and the performance is better. The reason is that the United-FS costs less CPU than LVM which needs CPU for each block address computation.

5.3. User Scalability

We measure the performance of the 8-disk I/O server varying the number of local users who concurrently conduct I/O operations. Figure 4 shows the write performance when the number of local users increases and figure 5 shows the read performance result. The write performance of United-FS is better than LVM because we distribute the requests to different disks while for LVM all user requests will be distributed to each disk. As to the United-FS, the reduced number of requests on a disk will make the disk head movement less random. But for read, as shown in figure 5, when there are 16 processes reading files, the performance of United-FS is not as good as LVM. The reason is that read is a synchronous operation and the workload is not heavy enough. So the parallel disk read of LVM performs better than United-FS using single data policy which does not fully exploit the disk parallelism when the workload is light. As the number of processes increases, the parallelism of disk access of United-FS is better.

Figure 6 shows the sequential write performance when the number of NFS clients increases and figure 7 shows the read result. We start 64 nfsds to process NFS requests on NFS server. The client number scales from 8 to 128. The United-FS write performance is better than LVM due to the writeback mechanism in Linux kernel which will gather write requests to increase the sequential write rate. We can see from figure 7 that the read performance of United-FS is not good as LVM. The reason is that read is synchronous operation and 64 nfsds influence the sequential read effect greatly. In such case, concurrent read access of LVM performs better.

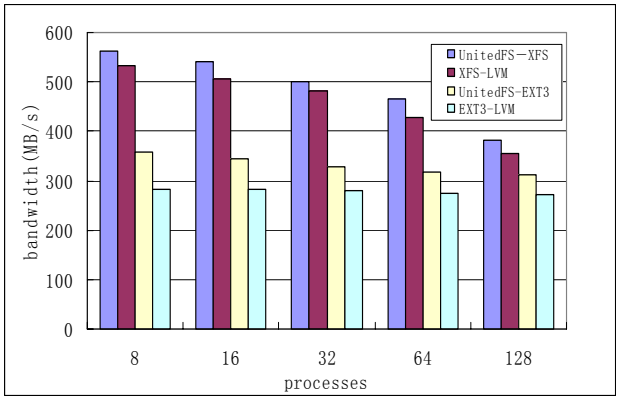

Fig. 4. Write Performance by Number of Processes

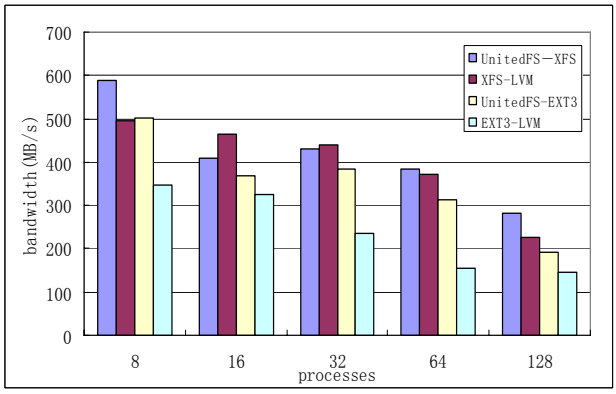

Fig. 5. Read Performance by Number of Processes

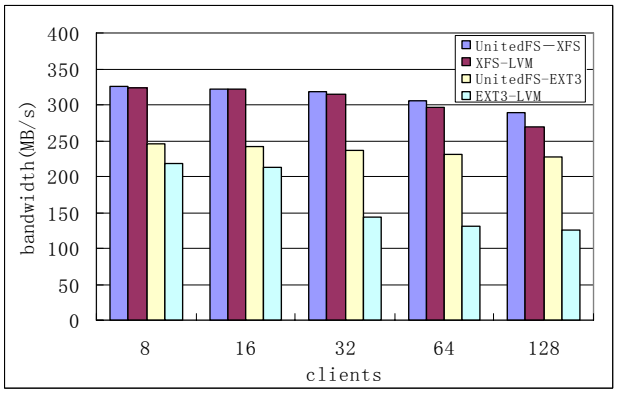

Fig. 6. NFS Write Performance by Number of Clients

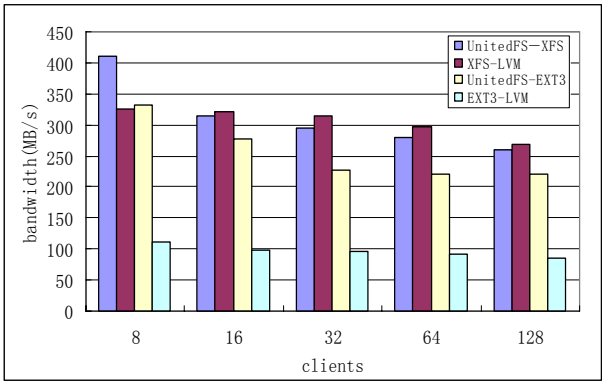

Fig. 7. NFS Read Performance by Number of Clients

6. Related Works

A single disk has the limitation in capability and physical data access speed. To maximize the bandwidth of local storage, many methods are exploited to make a collection of disks work in parallel. The method can be classified into three categories:

One is at the hardware level. The dedicated I/O server is configured with high-performance hardware RAID disks and presents to the host only a single disk for RAID array. In I/O intensive environments, performance is optimized by striping the large I/O request into several records distributed to different drives in the array. This solution works but it is quite expensive.

Another is at system driver level--software RAID. The Linux software RAID driver supports currently RAID levels 0,1,4,5 and linear mode. Such method occupies host system memory and consumes CPU cycles. The performance of a software-based array is directly dependent on server CPU performance and load. In contrast, hardware RAID occupies less host system memory, and it is operating system dependent. Linux software RAID can distribute data across ATA, SCSI, iSCSI, SAN, network or any other block device while hardware RAID cannot even span a single card.

The third method is at file system level. The benefit gained through this method is that it has much more flexibility. It can implement aggregated I/O bandwidth regardless of the type of hardware or the underlying file system. RAIF[10] is a fan-out stackable file system that implements RAID layout in file system level. United-FS differs with RAIF in several aspects. United-FS has a specific metadata file system that stores the location and relative metadata information. At this point, United-FS is a linear stackable file system because the dentry in United-FS maps only one dentry in metadata server file system. However, in RAIF, the dentry in RAIF manages multiple dentries in sub file systems. Thus, our data server file system layout is different from the layout of the sub file systems of RAIF. We can construct the data server file system directory tree layout that matches the characteristic of the file system best.

7. Conclusions and Future Work

United-FS is designed for a special hardware I/O Server which has multiple I/O channels allowing multiple I/O requests to be processed in a parallel pattern. United-FS is implemented on the top of several physical file systems, so it has the flexibility in distributing the request to the proper physical file system which is most efficient in processing the request. United-FS can be exported by NFS server to provide single NFS file system image to clients in the middle scale multi-user cluster environment. This paper also shows the characteristic and performance comparison between United-FS and software RAID. In future, we will add an intelligent scheduling mechanism to schedule the I/O requests in a much more adaptive mode to increase the performance in NFS environment and we will try to improve the metadata process efficiency.

8. References

[1] Sun Microsystems, Inc. “RFC 1813 - NFS: Network File System Version 3 Protocol Specification.” IETF Network Working Group. June 1995.

[2] Lustre: A Scalable, High Performance File System. Cluster File System, Inc. 2003.

[3] P. Schwan. Lustre : Building a file system for 1,000-node clusters. In *Proceedings of the Linux Symposium*, Ottawa, July 2003.

[4] Philip H. Carns, Walter B. Ligon III, Robert B. Ross, Rajeev Thakur, “PVFS: A Parallel File System for Linux Clusters”, In *Proceedings of the 4th Annual Linux Showcase and Conference*, October 2000

[5] Lombard P., Denneulin Y. nfsp: A distributed nfs server for clusters of workstations. In *Proceedings of International Parallel and Distributed Processing Symposium*, 2002

[6] D. Roselli, J. Lorch, and T. Anderson. A comparison of file system workloads. In *Proceedings of the 2000 USENIX Annual Technical Conference*, San Diego, CA, June 2000.

[7] Muntz, D., Building a Single Distributed File System from Many NFS Servers. Technical Report HPL-2001-176, 2001

[8] H. Tang, A. Gulbeden, J. Zhou, W. Strathearn, T. Yang, and L. Chu, A self-organizing storage cluster for parallel data-intensive applications. In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC '04)*, ittsburgh, PA, Nov. 2004.

[9] D. Patterson, G. Gibson, and R. Katz, A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the ACM SIGMOD*, June 1988

[10] Joukov N., Rai A., Zadok E., Increasing distributed storage survivability with a stackable raid-like file system. In *Proceedings of the 2005 IEEE/ACM Workshop on Cluster Security in conjunction with the 5th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*,Cardiff, UK, 2005.

[11] Wright C. P., Dave J., Gupta, P., Krishnan H., Quigley D. P., Zadok, E., Zubair M. N., Versatility and unix semantics in namespace unification. *ACM Transactions on Storage*, ACM, Vol. 1, No. 4, Novemeber 2005.

[12] Erez Zadok, Rakesh Iyer , Nikolai Joukov , Gopalan Sivathanu, Charles P. Wright , On Incremental File System Development. *ACM Transactions on Storage (TOS)*， Volume 2 , Issue 2 , May 2006

[13] G.Gibson and R.Meter, "Network Attached Storage Architecture". *Communications of the ACM*, vol.43, 2000

[14] http://sources.redhat.com/lvm2/

[15] Norcott, W. Iozone filesystem benchmark. URL: http://www.iozone.org/, 1998