SIFA: A Scalable File System with Intelligent File Allocation

Hsiao-Ping Tsai, Jiun-Long Huang, Cheng-Ming Chao,
Ming-Syan Chen and Cheen Liao[†]
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC
email: mschen@cc.ee.ntu.edu.tw

Synology Co. Ltd.[†]
Taipei, Taiwan, ROC
email: cheen@synology.com

Abstract

In this paper, we propose and evaluate a new NAS based system which is a consolidated file system in a network environment. The consolidated file system utilizes storage boxes cooperating as peers to provide all file system services, and presents itself to its NFS clients as a unified storage server. We have prototyped several key operations in this system. To provide more performance insights, we build a comprehensive system simulator and conduct extensive performance studies for the system via the simulator. This simulator has been run against a workload of a real Web server, and the system behavior of this workload is studied. In light of the experimental results, a consolidated file system is being implemented in full scale.

1. Introduction

With the explosive growth of Internet and data, there is a rapidly increasing demand for the storage. Unfortunately, these jobs to increase storage space for traditional file systems are done manually by system administrators, which is rather costly in comparing with the purchase price of the storage hardware [3]. In addition, when the partition process of allocating heavily used files and directories to a single server will unavoidably result in hotspots, which will in turn cause performance bottleneck. To achieve better performance and reliability, file migration and file replication issues have also been addressed in [8]. However, both migration and file replication increase the cost and the management complexity.

Most server storage will be external and networked by 2005 [3]. What makes this paradigm possible is the rapid increase of network bandwidth. In addition, the total cost causes enterprise customers to out-source their storage need to Internet storage service providers. Much effort has been elaborated on the improvement of a distributed storage to provide better performance and scalability. A network-attached disk introduces a new scheme to reduce the workload of a file server by transferring data directly between the clients and the network storage system [5][9][10]. Many systems such as Zebra [6], xFS [4][14], and SAN [15] are based on the remote storage concept. Specifically, Zebra is a log-structured file system with a remote storage on the network. Zebra buffers the write data and writes complete segments to the remote storage [6]. xFS extends Zebra design to achieve better scalability by dynamically clustering disks into stripe groups [4][14]. SAN uses dedicated fibre channel adaptors to provide physical connectivity among servers, remote disk arrays and tape libraries. Data is transferred via serial I/O rather than network protocols [15]. However, xFS continues using LFS. The dynamically changing of data and metadata locations increases the overhead of reads. In addition, the overhead of the cleaning process that creates empty segments is large as the workload is heavy. As for SAN, it is not suitable for large network due to the expensive connection media and the difficult installation. Network-Attached Storage (NAS) is a new distributed storage architecture that dispatches storage functions from a traditional file server to other file servers optimized for file I/O activity on the network. NAS storage servers equipped with disk arrays or tape device are dedicated for network file services. With NAS, server I/O bottlenecks are reduced, reliability and data availability are increased, allocation and use of resources are more efficient, the total cost are lower, and so on [1][2][12].

A consolidated file system based on the concept of NAS is proposed and implemented in this study. The consolidated file system we design is referred to as SIFA (standing for Scalable File System with Intelligent File Allocation). SIFA further groups file servers together to provide better scalability, availability, flexibility, and to reduce administration cost. In SIFA, file servers cooperate in peer

Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02)
0730-3157/02 $17.00 © 2002 IEEE

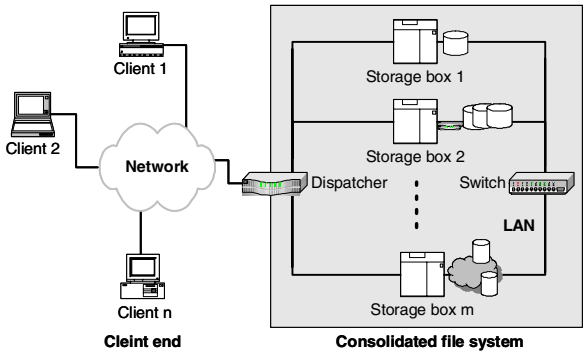

Figure 1: Block diagram of the consolidated file system

and each of them can provide full file services. With the two proposed schemes including FBM (standing for File-Based Mapping) and DFM (standing for Directory-Based Mapping), these storage boxes cooperate to accomplish a virtual and unified storage system to NFS clients. The proposed architecture includes one dispatcher, multiple peer storage boxes, and a back-end private LAN, as shown in Figure 1. The dispatcher is used to dispatch requests among file servers. The storage box is a file server which is optimized for file I/O activity and equipped with storage capacity by techniques such as SAS, RAID, SAN, or even another NAS. They are where the files stored. All control messages and the metadata are transferred via the back-end private LAN to prevent front-end network congestions. In SIFA, scalability is achieved by attaching a new storage box to the back-end LAN. This is done dynamically without disturbing the normal operations. After a new storage box is added, creation requests can be forwarded to the newly added storage box. A file migration procedure may also automatically be invoked to balance the storage space or workload, thus achieving the scalability and the flexibility. In this design, the file placement mechanism may be dynamic adapted, according to upper layer applications such as WEB, proxy and multimedia applications, to achieve better performance. In this paper, a simple file placement mechanism is also proposed in Section 3. In addition, some of the old file servers may cooperate together to continue file services by appropriate upgrade of the operating systems [7][13]. This can reduce cost greatly in system upgrade.

We have prototyped several key operations of the proposed system in FreeBSD 4.1 kernel. We design a stackable layer to FreeBSD 4.1 file system. Some stackable layering techniques are discussed in [7][11]. The layer is stacked upon the original vnode operation layer. In the layer, we implement vnode operations for the proposed system.

To provide more performance insights, we implement a trace-driven simulator against a workload of a real Web server. The simulation results capture several key factors that influence the system performance. The mapping mechanism, the issue of caching, the number of storage boxes, and the bandwidth of the back-end network influence the system performance prominently. Several simulation results with different configurations of storage boxes are also analyzed. In light of the experimental results, a new consolidated file system with good scalability, availability, and flexibility is being implemented in full scale for commercial purposes.

The rest of this paper is organized as follows. Section 2 provides an overview of the proposed system. Section 3 describes the naming schemes, FBM and DBM, and a placement policy. We describe the system simulator and discuss the system performance from the experimental results in Section 4. Section 5 concludes this paper.

2. Description of SIFA

The block diagram of SIFA is shown in Figure 1. This system includes one dispatcher, multiple storage boxes, and a private LAN.

2.1. Dispatcher

The dispatcher is the portal for clients to access this storage system. More than NAT, it also routes the inbound packets to a suitable storage box. Note that the dispatcher is not essential to the proposed system. Each storage box can accept client requests directly. However, with the dispatcher, client requests can be dispersed to storage boxes, leading to better availability and balance.

2.2. Storage Box

Files and directories are distributed and stored in storage boxes. Each storage box in the system is a file server and equipped with the storage capacity. These storage boxes work together to accomplish a virtual and concentrated storage system which is presented as a tree-like unified file system to NFS clients. In this design, the file placement mechanism may be dynamically adapted, according to upper layer applications such as WEB, proxy and multimedia applications, to achieve better performance. A storage box may forward creation requests to other storage boxes or dynamically invoke a file migration procedure to move local files and directories to other storage boxes automatically.

As shown in Figure 2, each storage box comprises four components, namely Naming Translator (NT), Interpreter (INT), Inter-box Communicator (COMM), and Operation Executor (OP), which will be described in details below.

- **NT Layer:**

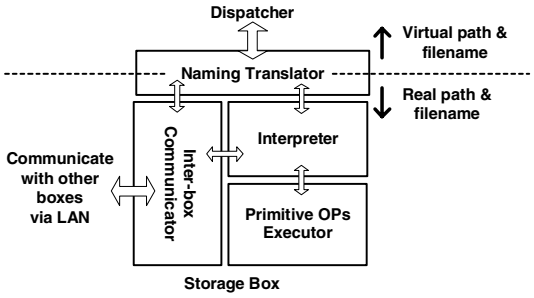

Figure 2: The components of storage boxes

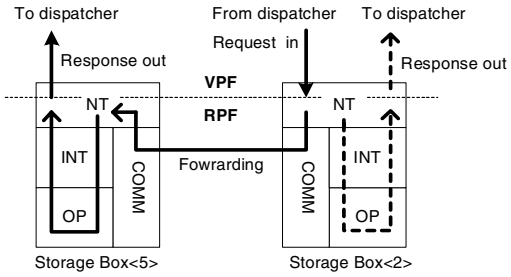

Figure 4: NT<2> forwards a request to NT<5> through COMM<2> and COMM<5>.

In SIFA, clients can access a file or a directory by a specified pathname seamlessly. The pathnames viewed by clients are referred to as VPFs (virtual pathname). In contrast, the information about file locations is referred to as RPFs (real pathname), including storage box id, path, and filename. In order to record the information about in which storage box a file is physically located, a mapping table is designed in the proposed system.

A mapping table containing the VPF-to-RPF mapping entries is maintained by NT. Accordingly, NT first translates the VPF to the corresponding RPF so that the system knows in which storage box the requested file is located. Then NT checks the RPF to see if the file exists locally. If yes, NT directly delivers the request to INT for further processing, as shown by the dashed line in Figure 4. Otherwise, NT hands the request over to COMM in order to forward it to the corresponding storage box. For example, a request is dispatched to storage box<2>, then NT<2> will forward the request to NT<5> through COMM<2> and COMM<5> since the RPF shows that the file is stored in storage box<5>, as shown by the solid line in Figure 4. After that, NT<5> will deliver the request to INT<5> for further processing. The virtually unified tree architecture and the physical tree architectures of this example is shown on Figure 3. The digit in < > stands for the box id.

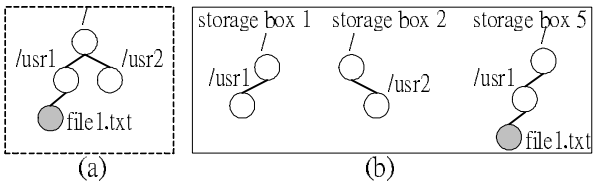

Figure 3: (a)The virtually unified tree architecture (b)The physical tree architectures of storage box 1, storage box 4 and storage box 5

- **INT Layer:**

INT is a layer that handles distributed file services. INT translates NFS Remote Process Calls (RPC) of client requests to the primitive vnode operations of the local file system. In most cases, INT translates RPC requests and then hands on the requests to OP for file system operations. However, there are two special cases for INT in the system. One occurs when INT receives requests containing the operations on directories such as “READDIR” or “RMDIR”. When the target directory is a cross-box directory, INT must duplicate several requests to the other corresponding storage boxes. The other special case occurs when INT receives a request for a file creation or a directory creation such as “CREATE” or “MKDIR”. In certain situations, such as insufficient disk space, INT needs to forward such requests to other storage boxes. We will go into details about this issue later.

- **COMM Layer:**

COMM transmits and receives information between storage boxes, including request forwarding, control signals propagation, and status messages collection.

- **OP Layer:**

OP implements local file services. For example, OP implements the vnode operation layer in FreeBSD including local file operations such as VOP_READ, VOP_WRITE, VOP_LOOKUP, and so on.

3. File-Based Mapping (FBM) and Directory-Based Mapping (DBM)

A key challenge to distributed storage is to couple multiple file servers to provide a unified view to its clients. All file systems have to provide some schemes to manage metadata that is used in looking up data blocks with a specified pathname. In SIFA, NT handles all metadata management

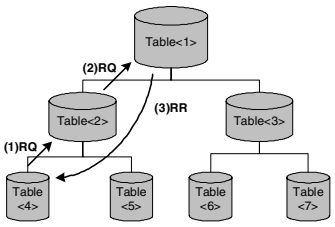

Figure 5: The tree structure of the file-based mapping table

for naming. In this paper, the metadata for naming is designed to be a mapping table that records the mapping relations between VPFs and RPFs. There is a trade-off between network and space overhead. On one hand, while the mapping table is dispersed, VPF-to-RPF transmission may cause a lot of network traffic. On the other hand, the disk space required for the whole system mapping entries is relatively huge to a single storage box, implying the necessity of striking a compromise between these two factors. In the paper, we propose two schemes, file-based mapping in tree (referred to as FBM) and directory-based mapping (referred to as DBM), for naming and metadata management. With the proposed schemes, a single volume can be spanned on multiple file servers. Both the two schemes also improve the ability and flexibility to adapt variant file placement policies for applications such as web, proxy, or multimedia to achieve better performance. As will be shown in the experimental studies later, the directory-based mapping scheme outperforms the file-based mapping one in performance whereas the latter processes better flexibility in file migration.

3.1. File-based Mapping in Tree (FBM)

In FBM, the mapping table in FBM records the VPF-to-RPF relationship to the level of file. Files in the same directory can be stored separately at different storage boxes. Figure 5 shows the tree structure of the mapping tables in FBM. Each node stands for a mapping table in a storage box. Each mapping table on a leaf node only stores the mapping entries of the objects within its local storage box. Each mapping table on a parent node stores not only mapping entries within its local storage box but those in its child nodes. Consider the example in Figure 5 where mapping table<2> stores all mapping entries in storage boxes 2, 4, and 5. The mapping table on root (i.e., on box 1) contains the whole system mapping entries. Depending on the position in the tree structure, NT queries the local mapping table first in the lookup procedure, and a Remote Query (referred to as RQ) will not be sent to parent NT until the query fails. According to the result, the client request can be served locally or be forwarded to the storage box that physically contains the file or the directory. In this design, the maximum number of RQ for a client request is $\log_2^N$ where N stands for the number of storage boxes.

Additionally, we also consider the caching of mapping table in order to enhance the performance of queries and reduce the amount of NT communications. The cache in NT is called type-1 cache. Another placement of the cache in SIFA is in the dispatcher. The dispatcher cache records the id of the storage box that the last request with the same VPF was dispatched. The cache in the dispatcher is called type-2 cache.

3.2. Directory-based Mapping (DBM)

In the directory-based mapping scheme, the mapping table records the VPF-to-RPF relationship to the level of directory. Each NT contains a copy of the complete mapping table. However, the flexibility for data migration is limited since all files in the same directory should be moved concurrently. Also, the scenario for storage boxes to be busy in file migration may happen in DBM. This scenario occurs when all storage boxes have nearly the same free space but will be full soon after a migration.

3.3. Data Placement Policy

The creation of a file or a directory is another important consideration in the design of a distributed file system since it involves the file locality and the capacity balancing issues among storage boxes. In SIFA, INT also takes charge of choosing a storage box to create a file or a directory based mapping on the status of the ensemble file system.

INT in each storage box maintains a capacity table about the ensemble storage volume information and periodically sends messages of the latest system capacity information of itself to the other storage boxes. For example, Table 1 shows the capacity table in each INT in our system composed of four storage boxes. The value of *Average Usage* stands for the average volume consumed of all storage boxes. We define a *Dynamic Threshold (DT<bid>)* for each storage box according to *Total Volume<bid>* and *Average Usage*. We assume that the *Total Volume* of each storage box is fixed, so that the value of *DT<bid>* increases as the value of *Average Usage* increases. The value of *DT* allows INTs to ascertain the capacity status of each storage box. The capacity status of a storage box *Over-threshold (*respectively, *Under-threshold)* means that the volume consumed in that storage box is relatively high (respectively, low).

In DBM, the flexibility for file locality and capacity balancing will be relatively limited. If the capacity status is *Over-threshold*, data migration of a directory between storage boxes must be performed to achieve the capacity balance among directory-based storage boxes. The data mi-

Box ID	Total Vol.	Used Vol.	Threshold	Status	DT - UV
1	20000	16000	15000	O	-1000
2	20000	4000	15000	U	11000
3	20000	8000	15000	U	7000
4	40000	12000	25000	U	13000
Average Usage		10000	(UV = Used Volume, bid=Box ID)		

Average usage = $\sum_{bid}$ UsedVol<bid>/ total number of boxes
Threshold <bid> = (Total Vol<bid> + Average usage)/2
(If DT<bid> > Total Volume<bid> : set DT<bid> = Total Volume<bid>)
Status : (**U**) **Under-threshold** : Volume Used ≤Threshold
(**O**) **Over-threshold** : Volume Used > Threshold

Table 1: Each INT maintains a capacity table about system volume information.

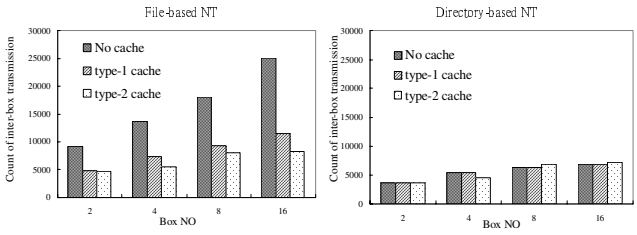

Figure 6: The relationship between the count of inter-box transmission and the number of stroage boxes. (Request frequency = 10, NT cache size = 64, dispatcher cache size = 32)

gration is an important issue but is beyond the scope of this paper.

4. Experimental Studies

In order to assess the performance of our proposed storage system, we implement a trace-driven simulator against a workload obtained from log data of a real Web server. The simulator, coded in C++, has been incorporated with complete functions of each component, including NT, INT, OP, and COMM, to model the storage system environment. In our simulation, it is assumed that each storage box is a machine (with CPU clock rate of 500MHz and main memory of 128MB) running FreeBSD as the operating system. The simulation workload is based on the real Web server log of National Taiwan University.

4.1. Experimental Studies

Several experiments are conducted to measure the performance of our proposed storage system. Since the system is composed of multiple storage boxes, the factors which may affect system performance are investigated, including the number of storage boxes, the frequency of incoming requests, and the count of inter-box transmissions.

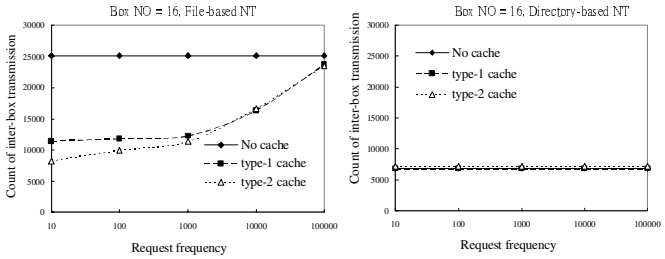

Figure 7: The relationship between the count of inter-box transmission and the request frequency (Box NO = 16, NT cache size = 64, dispatcher cache size = 32)

4.1.1 Experiment One : Inter-box transmission

The relationship between the count of inter-box transmissions and the number of the storage boxes is investigated first. The experimental results are shown in Figure 6 where it can be seen that RR and RQ cost a great portion of the inter-box transmission. The total count of inter-box transmissions increases rapidly in FBM. This is because the mapping tables are structured in a tree architecture and more storage boxes will increase the depth of the tree. The number of RQ/RR increases with the depth of the tree. As shown on Figure 7, adding caches into NT and the dispatcher can reduce the count of RQ and RR significantly. The NT cache is a LRU cache that stores mapping table searching results, (VPF, RPF). The dispatcher cache is a LRU cache that caches the VPF and the id of the storage box. With the dispatcher cache, a new-coming request is sent to the storage box that served a request with the same VPF recently. The result of the experiment that the NT cache is enabled is labeled ”type-1”, and the result of the experiment that both the NT cache and the dispatcher cache are enabled is labeled ”type-2” on Figure 6. The total count of inter-box transmissions increases when the incoming request frequency is extremely high because the updating of the NT cache in FBM cannot keep up with the request arrival rate. Therefore, there are many duplicated inter-box transmissions for RQ and RR as the request frequency increases.

4.1.2 Experiment Two : Average response time for each operation

In this experiment, the factors that influence average response time for each operation are considered. Figure 8 shows the relationship between the average response time and the number of storage boxes when the request frequency is 10000. The average response time decreases as the number of storage boxes increases. Therefore, increas-

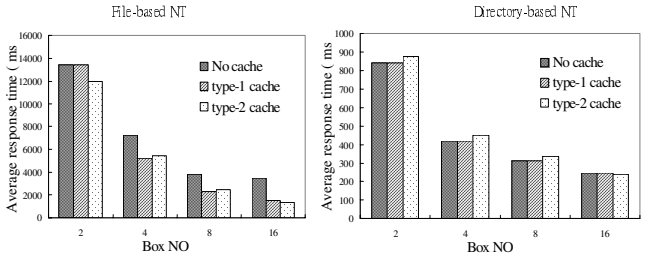

Figure 8: The relationship between the average response time for each request and the number of storage boxes (Request frequency = 10000, NT cache size = 64, dispatcher cache size = 32)

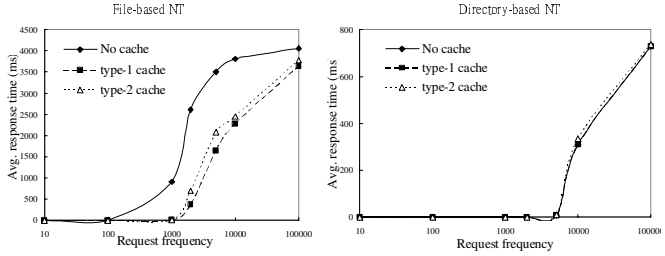

Figure 9: The relationship between the the average request response time and the request frequency (Storage box number = 8, NT cache size = 64, dispatcher cache size = 32)

ing the number of storage boxes can help in digesting request operations especially when the request frequency is high. The average response time for each operation increases with the incoming request frequency as shown in Figure 9. The average response time in DBM is much shorter than that in FBM because the inter-box transmission for RQ and RR will lengthen the delay. When the request frequency is low, the average response time only varies slightly for different numbers of storage boxes. However, the average response time varies with the storage box number significantly when the request frequency is high. Clearly, adding a NT cache in FBM can help in reducing the operation response time since the delay cost for RQ and RR can be reduced.

5. Conclusions

In this paper, the design of a new network-attached storage system is proposed. We have prototyped several key operations in a FreeBSD 4.1 kernel. To provide more performance insights, we conducted extensive performance studies in this paper. According to the simulation analysis, it was shown that NT with the directory-based mapping outperforms NT with the file-based mapping in many aspects. Also, the amount of inter-box transmissions is an important factor for the system performance. Note that there are tradeoffs between the system performance and the flexibility on file migration. Since cross-box directories cannot be used in the systems configured in directory-based mapping, the flexibility on file migration among storage boxes is limited. In light of the experimental results, a new consolidated file system with good scalability, availability, and flexibility is being implemented in full scale.

References

[1] Emc. http://www.emc.com.

[2] Network appliance. http://www.networkappliance.com.

[3] N. Allen. Don't waste your storage dollars: what you need to know. In *Research note, Gartner Group*, March 2001.

[4] T. E. Anderson, M. D. Dahlin, J. M. Neefe, and et. al. Serverless Network File Systems. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pages 109–126, Dec 1995.

[5] G. A. Gibson, D. F. Nagle, and et. al. File Server Scaling with Network-Attached Secure Disks. In *Proceedings of ACM SIGMETRICS*, pages 272–284, 1997.

[6] J. H. Hartman and J. K. Ousterhout. The Zebra striped network file system. In *Proceedings of the 14th SOSP*, pages 29–43, December 1993.

[7] J. S. Heidenmanin and G. J. Popek. File-System Development with Stackable Layers. *ACM Transactions on Computer Systems*, 12(1):58–89, February 1994.

[8] R. Hurley and S.-A. Yeap. File Migration and File Replication:A Symbiotic Relationship. In *IEEE Transactions on Parallel and Distributed Systems*, pages 578–586, June 1996.

[9] G. Ma, A. Khaleel, and A. N. Reddy. Performance Evaluation of Storage Systems Based on Network-Attached Disks. *IEEE Transactions on Parallel and Distributed Systems*, 11(9):956–968, September 2000.

[10] G. Ma and A. L. N. Reddy. An Evaluation of Storage Systems based on Network-attached Disks. In *1998 International Conference on Parallel Processing*, pages 278–285, 1998.

[11] M. K. McKusick, K. Bostic, and et. al. *The Design and Implementation of the 4.4BSD Operating System*. Addison-Wesley Publishing Company, Inc., 1996.

[12] M. Richey. Emc vs. network appliance. http://www.fool.com/portfolios/rulemaker/2000 /rulemaker000821.htm, Auguest 2000.

[13] G. C. Skinner and T. K. Wong. Stacking Vnodes: A Progress Report. In *Proceedings of the Summer USENIX*, pages 161–74, June 1993.

[14] R. Wang and T. Anderson. xFS: A Wide Area Mass Storage File System. In *Fourth Workshop on Workstation Operating Systems*, pages 49–69, October 1993.

[15] S. Wilson. *Managing a Fibre Channel Storage Area Network*. Storage Networking Industry Association, November 1998.