

USING A CLUSTER FILE SYSTEM -- TH-CLUFS -- TO CONSTRUCT A SCALABLE CLUSTER SYSTEM OF WEB SERVERS*

Liu Wei, Zheng Weimin, Shen Meiming, Wu Min, Ou Xinming

Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P.R.China

{lw, oxm, wumin}@est4.cs.tsinghua.edu.cn, {zwm-dcs, smm}@tsinghua.edu.cn

ABSTRACT

In this paper we present a novel cluster file system - TH-CluFS -- in distributed service environment, especially in Internet service area, and use the file system to construct a scalable web server cluster. TH-CluFS has a consistent file view across several servers connected by a high-speed local area network and out-performs the traditional distributed file systems. Due to the advantages of high-speed LAN, the system provides stronger support for the fast file access service than the previous distributed systems, such as NFS, AFS and Coda. In this paper we describe the key issues in cluster file systems and focus on the improvement from traditional distributed file system to current cluster file system - TH-CluFS.

KEYWORDS: Web server clusters, cluster file systems, metadata management, load balancing, file migration

1. INTRODUCTION

With the explosively increasing of Internet accesses, the web servers are suffering from the heavy loads and narrow I/O throughput. Cluster-based web servers provide a kind of feasible solution to many Internet problems, such as server I/O congestion and service failure. Web server clusters provide higher availability, higher scalability, more I/O throughput and more flexible

configuration method than traditional web servers do. In order to achieve the single system image of the cluster, a cluster file system is been built to manage the basic file operations and data management in cluster systems.

The main goal of web server clusters is to connect a set of computers to form a scalable virtual web server. A collection of computers connected by LAN does not itself constitute a cluster system. The collection becomes a system only when the computers in cluster can cooperate each other that the collection behaves as a coherent entity. It should appear the same interface to the users, to the administrators and to the programmers. In addition, the reconfiguration capability of clusters is the basis of scalability. Arbitrary amount of nodes may be configured in clusters. Adding or removing the nodes of a web server cluster to satisfy user variational requirements is one of the most important features of the clusters that insures the users' investments.

We have investigated many different methods to construct our web server cluster, including parallel file systems and distributed file systems. Unfortunately, these technologies are not met all our requirements. Parallel file systems, such as SPIFFI for Intel Paragon[1] and Bridge[2], can provide parallel I/O capability for file accesses, even to an individual file. However, the predefined striping groups restrict the flexibility of system configuration and reconfiguration. Distributed file systems, such as NFS[3], AFS[4][5] and Coda[6], have been widely used in networking world. But NFS can not provide a global file view between different users

* The paper is partially supported by National Key Project of Basic Research G1999032702 and National 863 High-tech Program under grant 863-306-ZT01-03-1.

or programmers. AFS and Coda File System have the similar goals to provide file services for machines around the world, so they have done many works on weak connection and mobile computing[7] that are not required in our cluster system. We expect to build a cluster file system on clusters, not a network file system that provides an approach for clients to access files on servers.

In the paper we present a novel cluster file system, TH-CluFS, that integrates the advantages from parallel file systems and distributed file systems. In the next section, the web server cluster is discussed. The design considerations of our TH-CluFS are presented in Section 3. In section 4 and 5, two key issues, distributed metadata management and I/O load balancing in TH-CluFS, are discussed in details. We have a conclusion in section 6.

2. CLUSTER OF WEB SERVERS

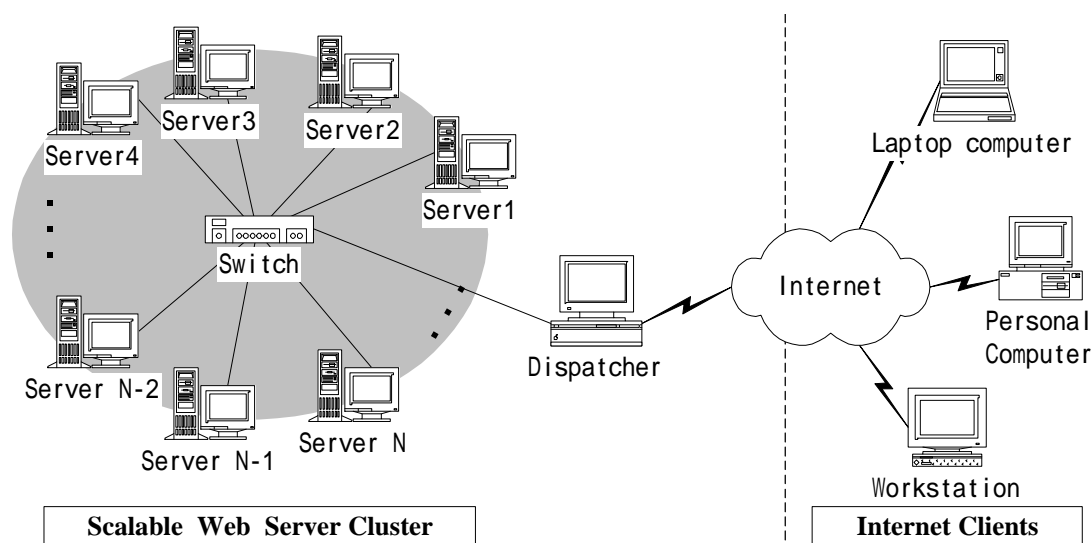


Figure 1 The architecture of scalable Web sever cluster

Cluster of Web Servers is a popular architecture for Internet services. The architecture of Web server cluster is shown in Figure 1. The Internet requests arrive at the *Dispatcher*. And the requests will be forwarded to the back-end servers from *Server 1* to *Server N*. A substantial problem is how to reasonably utilize the resources of the multiple servers, CPU, I/O, disk space,

memory etc. Providing a single image of the web server cluster is the key point of the architecture shown in Figure 1. In traditional server clusters, there are two ways to construct the file space on servers. One is duplicating the file sets across all servers and the other is constructing a distributed file system, such as AFS, Coda, on the servers[13].

TH-CluFS overcomes the shortages of the former two approaches and introduces a simple and effective way to construct a scalable Web server cluster. TH-CluFS involves the files and directories stored on Web servers in the cluster to form a consistent and transparent file space. It can save the disk space tremendously. As we know, in a mirror-based cluster only $1/N$ disk space is used to store “real” data and the other is to store replicated data.

Also, TH-CluFS simplifies the design and implementation of the *Dispatcher*. In general, *Dispatcher* treats the back-end servers have the same file sets, so it

allocates the Internet requests to one of them as it likes. However, TH-CluFS does not maintain multiple copies of a file in the cluster. The *Dispatcher* just forwards requests with Round-robin algorithm. The I/O load balancing will be achieved by TH-CluFS itself by file migration.

3. DESIGN CONSIDERATION

The dominant consideration in TH-CluFS is a global name space that supports real distributed storage in file granularity. As we know, many distributed file systems have a global name space for all users and applications. In general this name space consists of some *volumes*. A volume is a sub-tree of the name space. Each volume has to reside on a single server (or a set of replicated server), although on server might implement many volumes. In fact, these distributed file systems treats the volume as the smallest unit that has an autonomic manager. Although the structure is easy to implement and fits for the distributed environment, there are still some disadvantages when it was ported to cluster file systems. Because a volume can not be dispersed across different servers, we can not avoid the danger of imbalance in storage and I/O seriously. Therefore, in TH-CluFS the concept of volume is inherited. But the *CluFS volumes* have not to stay on a single server, it can be spanned multiple servers that are specified by system automatically. Of course, distributing data in file granularity brings many problems, such as metadata management, name cache, file migration etc. We will cover these problems in section 3.

As shown in above figure, our cluster is based on a

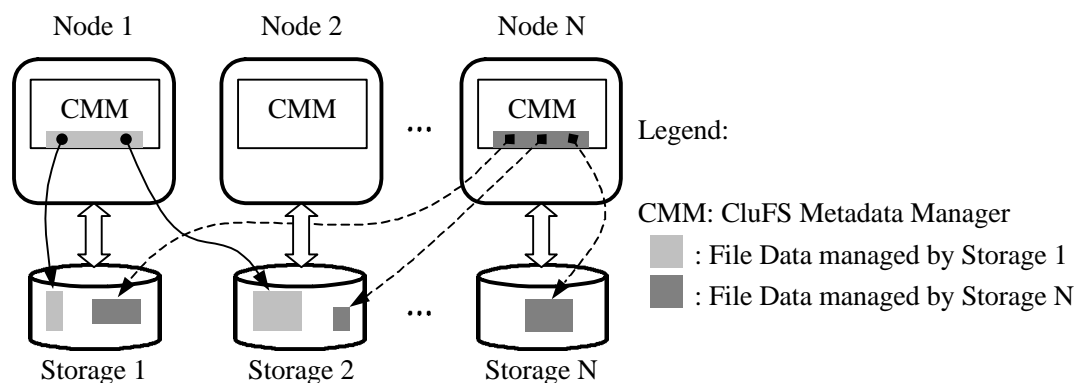


Figure 2 The storage architecture of TH-CluFS

peer-to-peer architecture. On each node, a set of CluFS volumes is managed by CMM. The corresponding file data are dispersed across multiple storage sites. We have been implemented a distributed naming mechanism to

resolve the pathnames.

4. METADATA MANAGEMENT

As we mentioned in last section, the system is on a peer-to-peer architecture that eliminates the boundary between servers and clients. Each node in clusters takes on the tasks of server and client concurrently. The file service in cluster as a part of OS will be easily integrated into distributed operating systems or cluster operating systems. This architecture paradigm can lead to a more efficient supercomputing or transaction processing in the area of parallel and distributed system.

CMM breaks the traditional volume mechanism in distributed file systems. The data in a CluFS volume can be spanned on different servers. It is a “real” distributed and transparent file system. You do not care on which server a volume stays. You just use the volume as a shared disk space. However, in traditional distributed file system a volume implicates a partition of certain server. And the volume in cluster file system is not on a single server or is replicated on multiple servers. It can be automatically span the content of the volume to each server in the cluster.

There are two basic structures in CMM: one is the

global replicated table that maps the pathname to a specified CMM server, the other is the local information database. The whole file system hierarchy is divided into several groups as shown in Figure 3. The global

replicated table has relatively small size so that the update operations to the table degrade the system slightly. The local database stores the most information of those managed CluFS volumes. Although CluFS volumes reside on several nodes, the metadata of volumes have to reside on a single node.

Moreover, we introduce a new name cache algorithm into TH-CluFS. The new name cache structure is based on a tree, not generic plain entry table. It saves the most useful information in the tree node to reduce the network traffic of name resolution. In cluster file system if the pathname is not in cache, the system will start resolve the path component-to-component. We cache the useful information of each component and use an optimal method to replace the useless node in the cache tree. The method is very effective for Internet access because the URL is normally quite long.

5. I/O LOAD BALANCING

With the availability of the high-speed local area networks, it becomes feasible to achieve balancing cluster file systems. The cluster file systems should have the similar I/O balance with the parallel file systems. It is quite interested and challenging to research on file granularity I/O balancing in clusters.

Dynamic I/O balancing scheme is used to support dispersing files in volume across multiple servers. Using server clusters to provide Web service, data balancing is a key factor to system performance. Due to the unexpected Internet access bursts, it may form a hotspot on certain server. The I/O balancing scheme can dynamically transfer the hotspot to a relatively free server. However the dynamic data transferring technique is little used in traditional distributed file systems because they generally have a relatively fixed hierarchy. The traditional concept of volumes also limits the data migration. The adopted method in distributed file

systems is the volume replication which partially solves the I/O bottleneck caused by file access imbalance. Because of the flexibility of CMM technique, we introduce the file migration into our TH-CluFS to

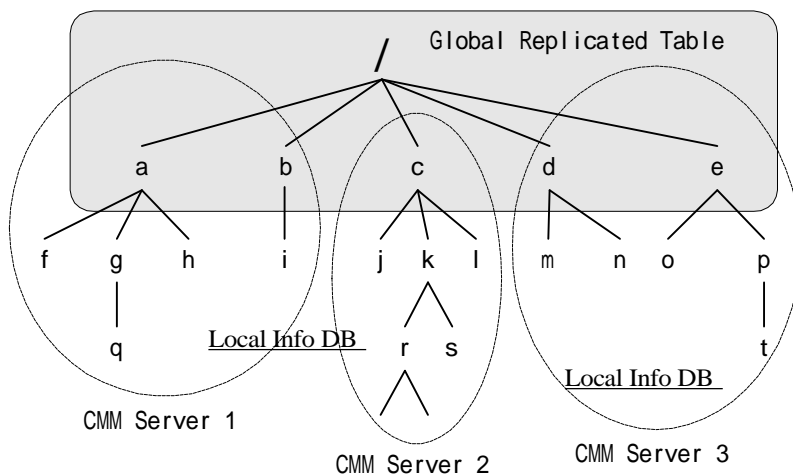


Figure 3 A sample of cluster file system hierarchy

achieve I/O load balancing in file level, not in volume level.

There are two policy adopted to implement load balancing. One is the static file assignment algorithm; the other is the real-time dynamic file migration policy. The static policy is mainly for file or directory creating. When a new object (file, directory or others) requests for creating operation, the static policy will determine a suitable site for the new object. Also a reassignment tool is given to optimize the file storage. The dynamic file migration policy is a kind of short-term file migration policy. The current I/O state information is collected and a migration determination is made dependently on the information.

6. CONCLUSION

In this paper we present the consideration and design of a novel cluster file system, named TH-CluFS that provides a single and shared name space for users and applications. A consistent file system view is built on the basis of CMM without the central name server. Furthermore, CMM supports file migration effectively. Our cluster file system can easily achieve a high

available, scalable and balancing file system with CMM.

7. REFERENCES

- [1] Craig S. Freedman, Josef Burger and David J. DeWitt, SPIFFI—A Scalable Parallel File System for the Intel Paragon, In *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 11, pp. 1185-1120, Nov. 1996
- [2] Peter C. Dibble, Michael L. Scott and Carla Schlatter Ellis, Bridge : A High-Performance File System for Parallel Processors, In *Proceedings of Eighth International Conference on Distributed Computing Systems*, pp. 154-161, Jun. 1988
- [3] Brian Pawlowski, Chet Juszczak, Peter Staubach, Carl Smith, Diane Lebel and David Hitz, NFS Version 3 Design and Implementation, In *Proceedings of Summer 1994 USENIX Conference*, pp137-151, Boston
- [4] M. Satyanarayanan and Mirjana Spasojevic, AFS and the Web: Competitors or Collaborators?, In *Proceedings of the Seventh ACM SIGOPS European Workshop*, Connemara, Ireland, Sep. 1996
- [5] Mirjana Spasojevic and M. Satyanarayanan, An Empirical Study of a Wide-Area Distributed File System, In *ACM Transactions on Computer Systems*, 14(2), May. 1996
- [6] M. Satyanarayanan, Coda: A Highly Available File System for a Distributed Workstation Environment, In *Proceedings of the Second IEEE Workshop on Workstation Operating Systems*, Pacific Grove, CA, Sep. 1989
- [7] James J. Kistler and M. Satyanarayanan, Disconnected Operation in the Coda File System, In *ACM Transactions on Computer Systems*, Vol. 10, No. 1, pp. 3-25, Feb. 1992
- [8] Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann, and Garret Swart, The Echo distributed file system, *Research Report III*, Compaq Systems Research Center, Palo Alto, CA, September 1993
- [9] T. Anderson, D. Culler, D. Patterson and the NOW team, A Case for NOW (Networks of Workstations), *IEEE Micro*, pp 54-64, 1995
- [10] D. Arredondo, M. Errecalde, S. Flores, F. Piccoli, M. Printista and R. Gallard, Load Distribution and Balancing Support in A Workstation-Based Distributed System, *Operating Systems Review*, Vol. 31, No. 2, Apr 1997, pp46-59
- [11] Cheng-Zhe Yang and Yen-Jen Oyang, Clue Tables: A Naming Mechanism for Building Highly Available and Scalable Distributed File Systems, *IEEE TENCON'93*, Beijing, pp. 291-293
- [12] Brent Welch and John Ousterhout, Prefix Tables: A Simple Mechanism for Locating Files in a Distributed System, *Technical Report No. UCB/CSD 86/261*, University of California Berkeley, Oct. 1985
- [13] Eric Dean Katz, Michelle Butler and Robert McGrath, A Scalable HTTP Server: The NCSA Prototype, In *Computer Networks and ISDN Systems*, Vol. 27, Sep. 1994