To appear in the 2003 IEEE International Conference on Cluster Computing (Cluster 2003), December 1-4, 2003, Hong Kong

Design and Performance of the Dawning Cluster File System

Jin Xiong, Sining Wu, Dan Meng, Ninghui Sun, Guojie Li
National Research Center for Intelligent Computing Systems
Institute of Computing Technology, Chinese Academy of Sciences
{xj, wsn, md, snh}@ncic.ac.cn, lig@ict.ac.cn

Abstract

Cluster file system is a key component of system software of clusters. It attracts more and more attention in recent years. In this paper, we introduce the design and implementation of DCFS [1] (the Dawning Cluster File System) — a cluster file system developed for Dawning4000-L. DCFS is a global file system sharing among all cluster nodes. Applications see a single uniform name space, and can use system calls to access DCFS files. The features of DCFS include its scalable architecture, metadata policy, server-side optimization, flexible communication mechanism and easy management. Performance tests of DCFS on Dawning4000-L show that DCFS can provide high aggregate bandwidth and throughput.

1. Introduction

The cluster computing technology[1] has matured as a mainstream method for building high performance computers. A shared global file system, which is also called cluster file system, is required by cluster systems. A cluster file system is marked by three characteristics: a shared file system, a global name space and a system call interface.

Early distributed file systems[2], such as NFS and AFS, are aimed at sharing files among network computers, with limitations when used as a cluster file system. With its single-server architecture and non-UNIX semantics, NFS can hardly support a cluster with large numbers of nodes.

The focal research issues of cluster file system are:

- High performance: including bandwidth and throughput. According to the US DOE&DOD, the requirement of aggregate bandwidth was about 20-60GB/sec in 2002, and will be increased to 50-200GB/sec in 2005.
- High scalability: the ability to support large clusters with hundreds even thousands of nodes. In recent years, there has been the requirement of large file system whose volume is 100TB-1PB servicing more than one thousand nodes.
- High availability: that is, a cluster file system is accessible without stopping the file services in the presence of component failures.
- Manageability: that is, a cluster file system should provide friendly utilities for deployment, configuration and steering.

Generally, there are two approaches to implement cluster file systems all through the ages. One is based on file servers and another is base on shared storage [4]. In order to improve the bandwidth, the disk striping technology in RAID is used in some cluster file systems in which file data are striped across disks of multiple file servers, such as UCB's xFS[9], IBM's GPFS[3], Clemson University's PVFS[7], etc. Some shared storage file systems are symmetric, that is, there is no server, such as Sistina's GFS[4,5], while others are asymmetric, that is, there are some kind of servers. For example, IBM's Tivoli SANergy[11] and SGI's CXFS[6] need a metadata server, and IBM's GPFS need a lock manager. However, SANergy, CXFS and PVFS do not support multiple metadata servers.

The Dawning Cluster File System (DCFS) is a file-server based cluster file system designed for the Dawning4000-L Linux cluster with the following features:

- Single system image: DCFS is a global file system sharing among all nodes. Applications see a single uniform name space, and can use system calls to access DCFS files.
- Scalable architecture: The architecture of multiple storage servers and multiple metadata servers makes DCFS more scalable.
- High bandwidth: With file data striping and server-side optimization, DCFS can provide high aggregate bandwidth.
- Powerful metadata processing: DCFS is able to

[1] This work was supported by the National High-Technology Research and Development Program of China (863 Program) under the grant No.2002AA1Z2102 and the grant No.2002AA104410.

make use of the processing power of multiple metadata servers.

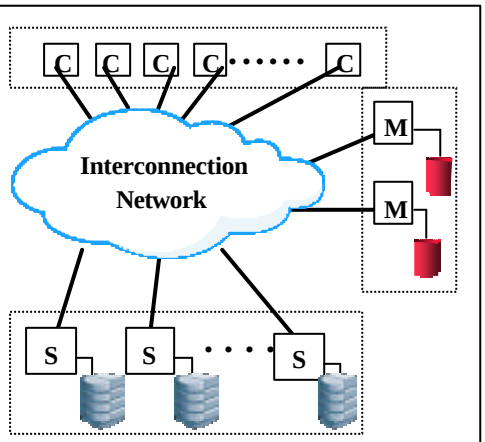
- Flexible communication mechanism: DCFS encapsulates its inside communication into an abstraction layer supporting either TCP or VIA.
- Easy management: DCFS provides management utilities to aid DCFS administration.

The remainder of this paper is organized as follows. The design and implementation of DCFS are described in section 2. The performance evaluation of DCFS on Dawning4000-L machine is presented in section 3. Concluding remarks are presented in Section4. Finally, we outline future work in Section 5.

2. DCFS Architecture and Design

2.1. Architecture

The architecture of DCFS is depicted in Figure 1. DCFS splits the file service into two orthogonal parts: operations on file contents and operations on metadata of files and the file system. There are multiple metadata servers providing storage and maintenance of file attributes, directories and the superblock. There are multiple storage servers for the storage and access of actual file contents. The architecture of multiple storage and metadata servers makes DCFS more scalable. The workload can be dispatched to a group of collaborative servers. In DCFS, metadata are stored in regular files of the native file system (such as EXT2) on metadata servers. And file data are stored on multiple storage servers in RAID 0 style of striping.



C: Client node, running kernel module and CLERK
M: Metadata server, running MGR
S: Storage server, running IOS

Figure 1. DCFS architecture

On client nodes which mount DCFS, the DCFS kernel module implements the interface with VFS and a user-level process called CLERK takes charge of communication with metadata servers and storage servers. So that applications can use standard system calls provided by Linux to access DCFS files, just like the way they access local files.

There is a user-level process, called CND, which acts as an agent to maintain the configuration status of DCFS. It can run on whichever nodes of the system.

2.2. Metadata Handling

Metadata processing is very important for file systems. According to SFS3.0[12], metadata requests account for over 60% of all requests in NFSv3. The approach of a single metadata server is simple, but limits the performance and scalability. The approach of multiple metadata servers is certainly more complex and difficult to recovery from failures, but is good for performance and scalability.

Unlike NFS, PVFS, and SANergy, which use the structure of local file system on the server to maintain their file system metadata, DCFS maintains and interprets metadata by itself for performance considerations.

In DCFS, there is a metadata server called super-manager who maintains the superblock, root directory and root inode of DCFS. An approach called first-level subtree distribution shown in Figure 2 is used to distribute the metadata among MGRs. Each metadata server stores and maintains a subtree of DCFS file system's root directory. When creating a new object in the root directory, the super-manager makes the decision that which metadata server is going to store and maintain the new object's metadata. The advantage of this

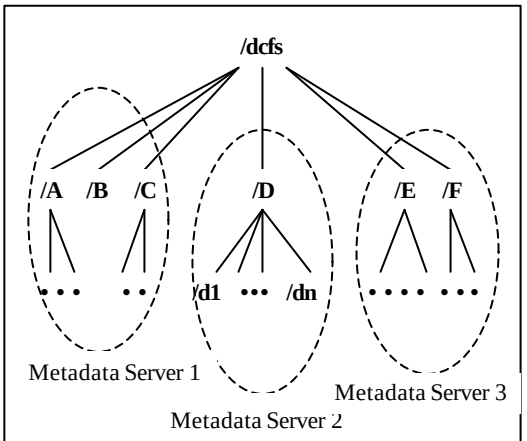


Figure 2. The distribution policy of metadata

distribution policy is that it keeps the parent-child relationship of the objects in DCFS file system. So that majority of metadata operations can be accomplished by requesting only one metadata server. In order to further improve performance, DCFS implements directory cache and inode cache on metadata servers.

2.3. Server-side Optimization

Storage servers are optimized by server-side data caching, multi-threading, and overlapping of disk access with network transfer. There are four kinds of threads on each storage server, as shown in Figure 3. The heartbeat thread is used for state monitoring. The flushing thread runs in the background and periodically flushes dirty cache blocks to disks. The actual file read and write operations are performed by the working threads. The scheduler thread decides whether to dispatch a thread pair or a single thread to serve this request according to the requested data size. A concept of thread pair is proposed, one thread is in charge of transferring data through network and another is responsible for writing data into or reading data from the storage. The thread pair improves the concurrency of disk access and network transfer.

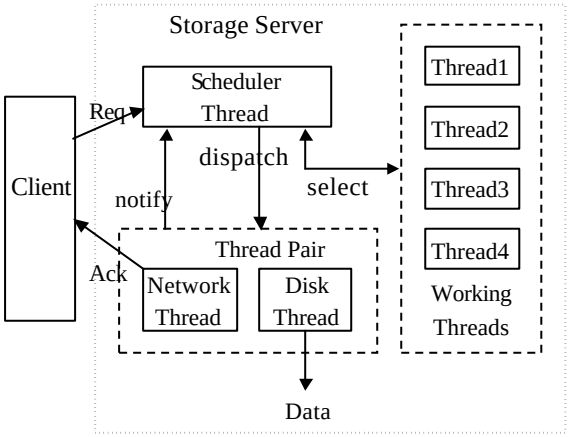


Figure 3. Multi-threaded storage server

2.4. State Monitoring

In order to discover component failures as soon as possible, a status monitoring mechanism is designed. DCFS organizes the servers and clients into two rings: one for all DCFS clients, another for all DCFS servers. Each one on the rings periodically sends heartbeat messages to its left and right neighbors. If some one couldn't get its neighbor's heartbeat for a specific time, then it will send a message to a special HA-master daemon to inform the abnormity. There is a backup of HA-master running on another node. The backup will take over the master when the HA-master is unreachable.

2.5. Communication Layer

Cluster file systems are heavily influenced by interconnection technology. In recent years, there have been many efforts in developing high-speed interconnection networks, such as Gigabit Ethernet, Fibre Channel, Myrinet, SCI, etc. And the developing of high performance communication protocols is also in progress, including VIA and IBA. We make an effort to encapsulate inside communication of DCFS into a flexible communication layer, which can not only use stream type communication protocols such as TCP and UDP, but also exploit high-performance communication protocols, such as VIA, IBA, etc.

2.6. Management

An agent and a set of utilities are designed to complete the management work, including system configuration, starting and stopping DCFS service, creating a new file system, and reporting the status. The agent process holds all the configuration and status information of DCFS and is in charge of receiving management requests from the administration utilities, passing the requests to relevant DCFS client/server daemons and waiting for the answers.

3. Performance Evaluation

The performance of DCFS is evaluated as follows:

- Peak performance: It is the highest performance with fixed number of servers;
- Server speedup: It is defined as the peak performance with N servers divided by the peak performance with one server. It shows the performance enhancement with the increase of the number of servers;
- Efficiency: It is defined as the peak bandwidth divided by the total disk bandwidth of N storage servers when disk performance is dominant. Then the cost of DCFS protocol is approximately 1 minus the efficiency. More formally,

$$E = \frac{BW_{Peak}}{BW_{Disk} \times N_{IOS}} \quad (1)$$

$$C = 1 - E \quad (2)$$

where E is efficiency, C is DCFS protocol cost, N_{IOS} is the number of storage servers, BW_{Disk} is the disk I/O bandwidth, and BW_{Peak} is the peak bandwidth with N_{IOS} storage servers.

- Sustainability: It is the maximum number of clients that can be supported simultaneously by a specific number of servers.

Besides, the performance of DCFS and that of PVFS are compared.

3.1. Test Platform

The performance results of DCFS are gotten on the Dawning4000-L. The machine consists of 322 nodes, with 2 login nodes, 256 compute nodes and 64 database nodes. The nodes are interconnected through four networks: Fast Ethernet, Gigabit Ethernet, Myrinet and a serial network for diagnosis. Our performance tests were done on 32 compute nodes. Each compute node is a Dawning Tiankuo® R220XP server running Linux 2.4.18-3smp, with 2 2.4-GHz Intel® Xeon™ Processors, 2-Gbyte memory and 3 73-Gbyte SCSI disks. Both DCFS and PVFS ran on the Gigabit Ethernet on which the TCP/IP communication bandwidth is about 106.2MB/sec (measured by **netperf**[13] for 16KB messages). The disks are Seagate Ultra320 with model number ST373307LC (with 8MB data buffer and 2.99 msec average latency). Both sequential read and write bandwidth of the disks are about 60MB/sec (measured by **iozone**[14] on EXT2).

3.2. Aggregate Bandwidth

The bandwidth of DCFS is determined by three factors: the network bandwidth, the disk I/O rate and the cost of DCFS protocol. We designed two sets of tests: small-file tests and large-file tests. In small-file tests, all files can be fit in the cache of storage servers so that there was no disk I/O. These tests show how well DCFS uses the network bandwidth. On the other hand, in large-file tests, the I/O size was much larger than the cache size of all servers so that the disk I/O is dominant. These tests show how well DCFS uses the disk bandwidth.

Figure 5 and Figure 6 show the aggregate bandwidths of DCFS for small files and large files respectively. In these tests, DCFS was configured with one metadata server, and each client node ran a test process **iozone** that read or wrote a DCFS file of specific size, which is $\frac{256 \times N_{IOS}}{N_{Client}}$ Mbytes for small-file tests and $\frac{6 \times N_{IOS}}{N_{Client}}$ Gbytes for large-file tests, where N_{IOS} is the number of storage servers in use and N_{Client} is the number of client nodes in use. The processes access different files.

Figure 5 is largely similar to Figure 6 except with higher values. And in both situations, one storage server can support more than 22 clients simultaneously without distinct performance decline. The peak bandwidths, speedups and efficiencies in large-file situation are summarized in Table 1. We noted that the performance enhancement is almost proportional to the increase of the number of storage servers in large-file situation. However, the efficiency decreases with increase of the number of storage servers, which means that the cost of DCFS protocol increases when more storage servers are added to the system.

We also noted the notable gap between the write bandwidths and corresponding read bandwidths in large-file tests. We believe the difference is caused by the disks on compute nodes. Table 2 shows the aggregate read and write bandwidths of the disk for various numbers of threads. The aggregate read bandwidth of multiple threads decreased dramatically when the number of threads was more than 1. In contrast, there

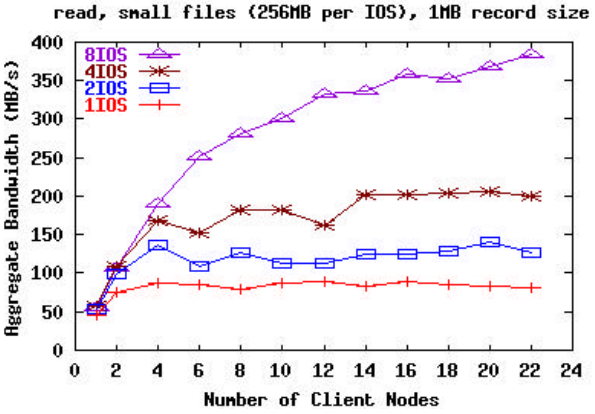


Figure 5(a) DCFS read bandwidth for small files

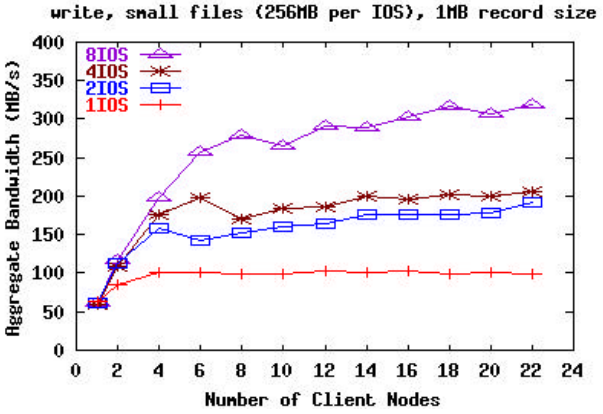


Figure 5(b) DCFS write bandwidth for small files

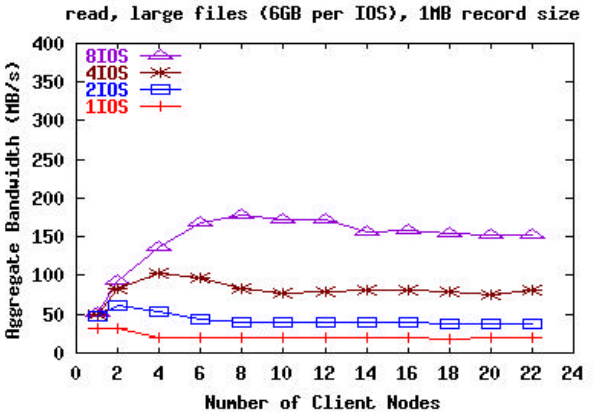


Figure 6(a) DCFS read bandwidth for large files

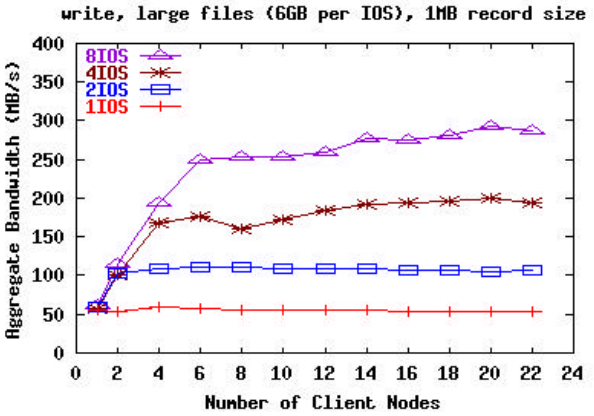


Figure 6(b) DCFS write bandwidth for large files

Table 1 Peak Bandwidths, Speedups and Efficiencies Corresponding to Figure 6

Num. Of IOSes	Total RW Size (MB)	Read			Write		
		Peak Bandwidth (MB/sec)	*Speedup*	*Efficiency*	*Peak Bandwidth (MB/sec)*	*Speedup*	*Efficiency*
1	6144	31.192	1	89.28%	59.578	1	94.13%
2	12288	60.790	1.95	86.98%	111.725	1.88	88.26%
4	24576	100.347	3.22	71.81%	198.84	3.34	78.54%
8	49152	178.361	5.72	64.82%	292.576	4.91	57.78%

Disk bandwidth for 4 threads to access 6Gbytes is about 34.943 MB/sec for read, and about 63.292MB/sec for write.

Table 2 Disk Bandwidths with Various Numbers of Threads

Num. Of Threads	Total Size (MB)	Read Bandwidth (MB/sec)	Write Bandwidth (MB/sec)
1	6144	66.800.	63.193
2	6144	41.476.	63.902
4	6144	34.943.	63.292
6	6144	33.860.	62.789
8	6144	33.880.	62.709
10	6144	33.850	61.661
12	6144	33.399	61.809
14	6144	33.155	60.922
16	6144	34.107	61.355

The bandwidths are sequential read/write bandwidths measured by iozone on EXT2. (Total access size is 6 Gbytes, and record size is 16 Kbytes.)

was no distinct decline for aggregate write bandwidth. In performance tests of DCFS, there were 4 working threads for each storage server to perform file read and write. That is why read bandwidth is much lower than corresponding write bandwidth in the large file tests. This result suggests that the number of threads on storage servers should be adjustable according to disk behavior.

3.3. Throughput

The file creation and removal rates are suitable to show the metadata processing performance. In DCFS, creation and removal for 0-byte files are done completely by metadata servers. However, for nonzero-length files, both metadata servers and storage servers participate in the operations.

Figure 7 shows the test results. In these tests, DCFS was configured with 4 storage servers, and each client node ran a test process that created or removed 5000 0-byte files or 5000 4-Kbyte files in a directory. And for 22 client nodes, these tests created (removed) totally 110,000 files, which were averagely distributed in 22 directories.

For 0-byte files, when there was only one metadata server, both file creation and file removal rates reached the peaks at 10 client nodes, 545.07 files/sec for creation and 3903.97 files/sec for removal. When the number of metadata server was more than 2, the tests did not reach the peak rates within 22 client nodes.

The file creation rate for 4-Kbyte files was much lower than that for 0-byte files by comparing Figure 7(c) with Figure 7(a). The file creation rate for 4-Kbyte files enhanced approximately 50% when the number of metadata servers increased from 1 to 2, but the file creation rates with 4 or 8 metadata servers were not higher than that with 2 metadata servers. We believe it is because the performance bottleneck in this situation is file write on storage servers. And the flat organization of files on storage servers, that is, all the files on storage servers are put into a single directory, is another reason. The figure of file removal rates for 4-Kbyte files is not presented here, because it is almost same as Figure 7(b).

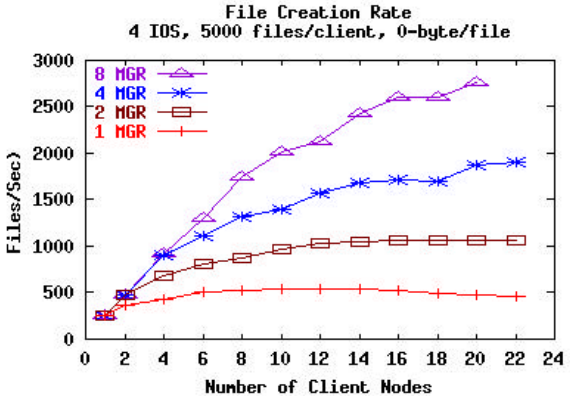


Figure 7(a) DCFS file creation rate for 0-byte files

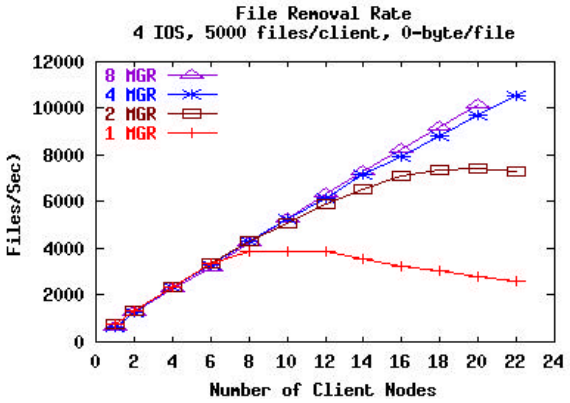


Figure 7(b) DCFS file removal rate for 0-byte files

3.4. Comparison

We compared the performance of DCFS with that of PVFS version 1.5.4[8]. There are many differences between DCFS and PVFS. There are multiple metadata servers in DCFS, while there is only one metadata server in PVFS. DCFS implements multi-threaded storage server. However, PVFS's storage servers (IODs in term of PVFS) are single-threaded.

Figure 8 shows I/O bandwidth of DCFS and PVFS. In these tests, both DCFS and PVFS were configured with 8 storage servers and one metadata server. And both of them used the Gigabit Ethernet for communication. In

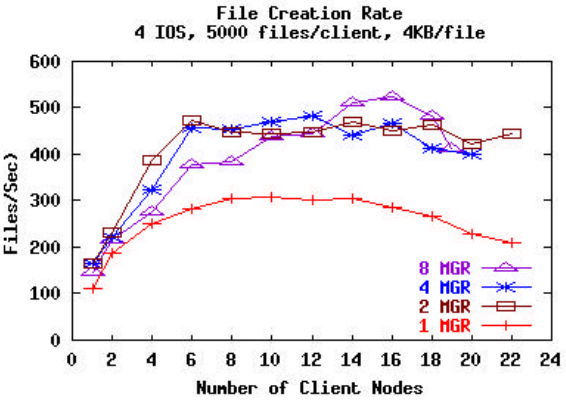


Figure 7(c) DCFS creation throughput for 4096-Kbyte files

Figure 8(a), each client node ran a test process that read/wrote a file of $2048/N_{Client}$ Mbytes, where N_{Client} is the number of client nodes. Similarly, in Figure 8(b), each test process read/wrote a file of $49152/N_{Client}$ Mbytes.

For small files, read bandwidth of PVFS is higher than that of DCFS. However, for large files, read bandwidth of PVFS is lower than that of DCFS. Moreover, write bandwidth of DCFS in either situation is higher than that of PVFS. The higher bandwidth of DCFS for large files indicates DCFS exploits disk I/O better than PVFS does, because storage servers in DCFS are multi-threaded so that disk accesses are overlapped with network transfers.

Figure 9 shows the file creation and removal rates of DCFS and PVFS. In these tests, both DCFS and PVFS were configured with one metadata server and 4 storage servers. And both used the Gigabit Ethernet for communication. As shown in Figure 9, for both 0-byte files and 4-Kbyte files, DCFS exhibits much higher file creation and file removal rates than PVFS does. These results indicate that DCSF is able to provide better throughput than PVFS when both of them are configured with one metadata server. We believe this owns to the metadata policy of DCFS. DCFS maintains and interprets the metadata stored in the local files on metadata servers. However, PVFS builds the whole directory tree in the local file system on its metadata server. Moreover, according to Section 3.3, when DCFS is configured with more metadata servers, its throughput can be enhanced greatly.

4. Conclusions

In this paper, we have described the design and implementation of the Dawning Cluster File System (DCFS) for Dawning4000-L Linux cluster. In order to achieve scalability and high performance, DCFS uses

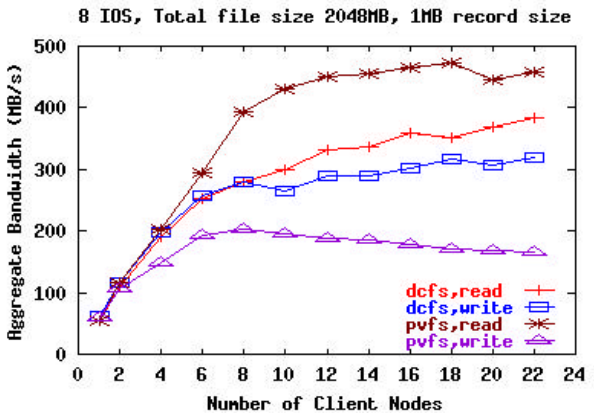


Figure 8(a) Bandwidth of DCFS and PVFS for small files

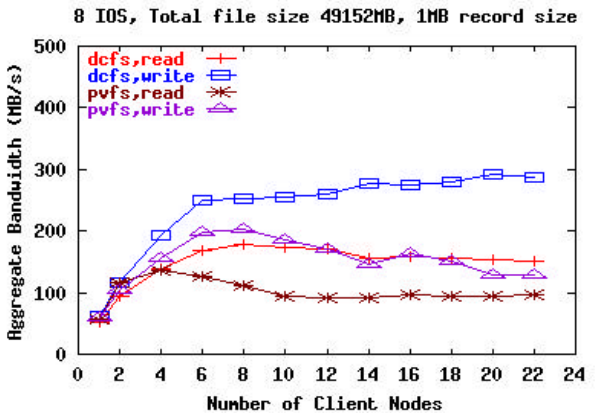


Figure 8(b) Bandwidth of DCFS and PVFS for large files

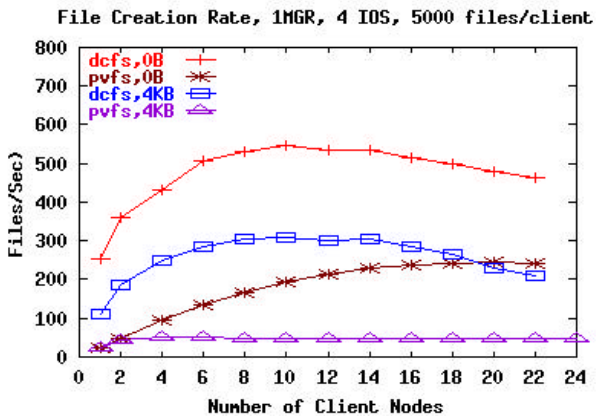


Figure 9(a) File creation rate of DCFS and PVFS for 5000 files per client

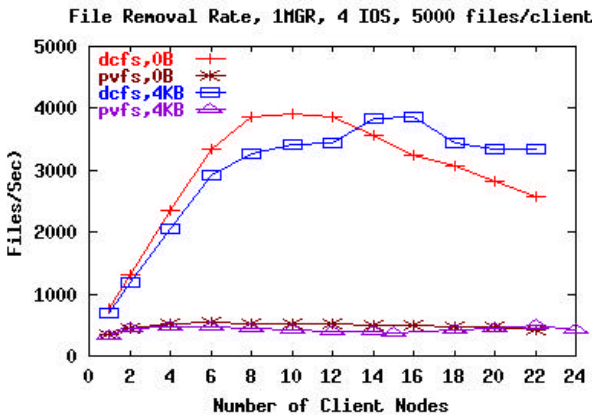


Figure 9(b) File removal rate of DCFS and PVFS For 5000 files per client

different policies for file data access and metadata access. File data are stored on multiple storage servers in a RAID-0 style of striping. Metadata are distributed among multiple metadata servers in a first-level subtree approach. Storage servers are multi-threaded to overlap disk access with network transfer.

The performance of DCFS on Dawning4000-L is encouraging. DCFS exhibits satisfactory speedup for both bandwidth and throughput. And compared with PVFS, DCFS exhibits higher bandwidth for large files, which owns to the multi-threaded storage servers of DCFS, and much higher file creation and removal rates in all situations, which owns to the metadata policy of DCFS

Compared with the previous cluster file system COSMOS[10] for Dawning2000 and Dawning3000, DCFS has similar architecture. However, DCFS has very different implementation policies, including the metadata policy, the file striping policy, the communication mechanism, disk access mechanism, the caching policy and the management policy.

5. Future Work

More performance analysis of DCFS is needed, especially under heavy-load conditions of large numbers of clients, and under conditons of 8 or more storage servers.

As the performance tests shown, the efficiency of DCFS is not satisfactory when the number of storage servers is 8 or more. We believe this is because of the cost of DCFS protocol, which needs to be further improved.

Although multiple metadata servers in DCFS improve the metadata processing performance, they complicated the failure recovery. It is difficult to ensure the metadata on all metadata servers consistent when failures occur. We are working on this problem.

The metadata distribution and workload may unbalance according to the current metadata distribution policy in DCFS, so better metadata distribution policy is needed.

In order to support high-performance scientific computing applications, we plan to implement a user level MPI-IO library on top of DCFS. This will greatly improve the bandwidth for MPI-IO based applications.

6. Acknowledgements

We would like to thank Guozhong Sun, Rongfeng Tang and Zhihua Fan for helping us to do the performance tests.

7. References

[1] Kai Hwang, Zhiwei Xu, *Scalable Parallel Computing: Technology, Architecture, Programming*, The McGraw-Hill Companies, February 1998

[2] Uresh Vahalia, *UNIX Internals: The New Frontiers*, Prentice-Hall,Inc., October 1995

[3] Frank Schmuck, Roger Haskin, " GPFS: A Shared-Disk File System for Large Computing Clusters", *Proceedings of First USENIX Conference on File and Storage*, January 2002

[4] Steven R. Soltis, Thomas M. Ruwart, Matthew T.O'Keefe, " The Global File System", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, volume 2, March 1996, pp. 319-342.

[5] Matthew T.O'Keefe, " Shared File Systems and Fibre Channel" , *Proceedings of The Sixth Goddard Conference on Mass Storage Systems and Technologies in cooperation with the Fifteenth IEEE Symposium on Mass Storage Systems*, March 1998, pp. 1-16.

[6] " CXFS™: A Clustered SAN Filesystem from SGI", White Paper, Silicon Graphics, Inc. September 2001

[7] Philip H. Carns, Walter B. Ligon III, Robert B. Ross, Rajeev Thakur, " PVFS: A Parallel File System for Linux Clusters", *Proceedings of the 4th Annual Linux Showcase and Conference*, October 2000, pp. 317-327

[8] PVFS, http://parlweb.parl.clemson.edu/pvfs/

[9] Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, and Randolph Y. Wang, "Serverless Network File Systems", *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, December 1995, pp. 109-126.

[10] Cong Du and Zhiwei Xu, "Performance Analysis of a Cluster File System", *Proceedings of the 2000 International Conference on Parallel and Distributed Technologies and Applications (PDPTA'2000)*, June 2000, pp. 1933-1938.

[11] "IBM Tivoli SANergy Administrator's Guide Version 3 Release 2", IBM Corporation, October 2002

[12] SPEC, "SFS 3.0 Documentation Version 1.0", Standard Performance Evaluation Corporation, 2001

[13] netperf, http://www.netperf.org/netperf/NetperfPage.html

[14] iozone, http://www.iozone.org/