A Novel Dynamic Metadata Management Scheme for Large Distributed Storage Systems

Yinjin Fu Nong Xiao Enqiang Zhou
School of Computer, National University of Defense Technology
Changsha, Hunan, 410073, China
Email: yinjinFu@gmail.com, xiao-n@vip.sina.com, eqzhou@263.net

Abstract

In large distributed storage systems, metadata is usually managed separately by a metadata server cluster. The partitioning of the metadata among the servers is of critical importance for maintaining efficient MDS operation and a desirable load distribution across the cluster. We present a Dynamic Directory Partitioning (DDP) metadata management scheme, directory metadata and file metadata are managed in different ways, and the dynamically changing workload can be balanced by adjusting the metadata distribution on metadata servers. Our simulation results show that our approach, comparing with other metadata management strategies, has advantages in performance, scalability and adaptability.

1. Introduction

With rapid improvement of PC performance and network bandwidth, large distributed storage system can be built by commodity PC cluster with high cost-performance. The prevailing system architecture for large-scale storage systems is object-based storage architecture nowadays [4]. A large-scale (≥ Petabytes or even Exabytes) storage system can be designed as in Figure 1. The architecture will consist of tens of metadata servers (MDSs), thousands of object-based storage devices (OSDs), and potentially tens of thousands of clients. File data will be striped across many objects on many OSDs that can be directly accessed by clients through the network, while metadata is managed separately by a separate MDS cluster consists of those MDSs. Clients will consult the MDS cluster, responsible for maintaining the file system namespace, to receive permission to open a file and information specifying the location of its content.

As metadata transactions are decoupled from file read and write operations, the MDS cluster need only concern itself with a relatively restricted set of operations. Basic metadata operations include *open*, *close*, *stat*, *getattr*, and *setattr*, while directory operations include *readdir*, *create*, *link*, *unlink*, and *rename*. Although the size of metadata is

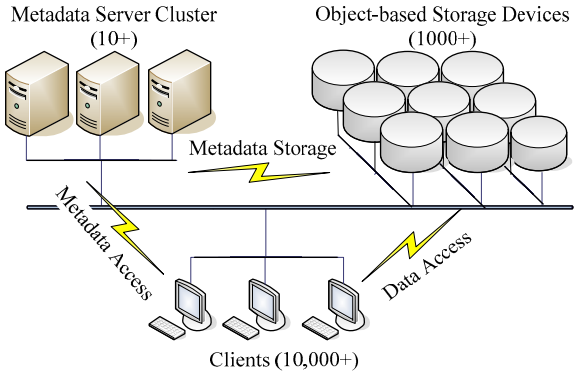

Figure 1. Storage System Achitecture

relatively small compared to the overall size of the system, more than 50% of all file system operations are metadata operations [1], making the performance of the MDS cluster of critical importance. Furthermore, unlike the overall capacity of the OSD cluster can easily scale by increasing the number of devices, metadata exhibits a higher degree of interdependence, making the design of a scalable system much more challenging.

There are a number of key design issues that will significantly affect the performance of a MDS cluster in such a system. For example, improving the effectiveness of MDS caching will be critical for sustaining high throughput and masking slow disk performance, and giving an effective partitioning strategy for the distribution of client metadata requests among MDSs, which will affect observed cache effectiveness and the ability of the system to cope with extreme workloads.

We present a novel dynamic metadata management scheme: Dynamic Directory Partitioning (DDP), to efficiently distribute responsibility for metadata of large distributed file systems across a MDS cluster. Directory metadata and file metadata are managed in different ways to improve the effectiveness of MDS caching and avoid the large-scale metadata migration according to the updating directory attributes. And the dynamically changing workload can be balanced by adjusting the metadata distribution on MDSs. Finally, we evaluate the performance, scalability and adaptability advantages of

978-0-7695-3352-0/08 $25.00 © 2008 IEEE
DOI 10.1109/HPCC.2008.86

our strategy by comparing with other metadata management strategies in a simulation environment.

2. Related Work

As the partitioning of the metadata among the MDSs is of critical importance for maintaining efficient MDS operation and a desirable load distribution across the MDS cluster, there are some approaches used in traditional and more modern distributed file systems. We divided them into two categories: static strategy and dynamic strategy, according to whether metadata can be dynamically redistributed. Static Subtree Partitioning [9], Static Hashing [8], Lazy Hybrid (LH) [2] and Directory Object IDentifier and Filename Hashing (DOIDFH) [5] are static strategies. Dynamic Subtree Partitioning [3] and Dynamic Hashing [4] are dynamic strategies.

2.1 Static Strategy

Static Subtree Partitioning divides the global name-space into subtrees and manually assigns them to individual file servers by system administrator. NFS [9], Coda [10] and countless other systems use this technique to partition namespace. Although it typically results in good cache performance due to the enforced locality of a particular server's workload, changes in that workload, as when many clients access a few files, result in a poor distribution of load among file servers.

Static Hashing distribute files across servers based on a hash of some unique file identifier, such as an inode number or path name. Lustre [6], zFS [8] and other systems all hash the file pathname and/or some other unique identifier to determine the location of metadata. Hashing provides good load balancing across MDSs and alleviates hot-spots consist of popular directories. But distributing metadata by hashing eliminates all hierarchical locality, and metadata update operations may incur a burst of network overhead.

LH and DOIDFH techniques are based on Static Hashing, LH uses Lazy Policies to defer and distribute update cost. In DOIDFH scheme, files metadata are maintained in different MDSs using file identifier (composed of parent directory OID and filename) hashing, and none metadata migration among MDS cluster after renaming a directory. The two approaches eliminate all hierarchical locality for they distribute metadata by file hashing. They are static strategies still, and cannot balance the dynamically changing workload.

2.2 Dynamic Strategy

Dynamic subtree partitioning can delegate authority for subtrees of the directory hierarchy to different MDSs and the delegations can be nested. To cope with the changing workload, Dynamic Subtree Partitioning leverages the dynamic load balancing mechanism to dynamically redistribute metadata among MDSs. The Ceph file system [7] utilizes a MDS cluster architecture based on Dynamic Subtree Partitioning. However, the scheme is a coarse adjustment strategy and it will take a long time to balance the workload.

Dynamic Hashing uses hashing to distribute metadata throughout the MDS cluster, and periodically redistributes metadata among MDSs. The major disadvantage of Dynamic Hashing scheme is that large-scale files metadata need to be migrated among MDS cluster after renaming a directory.

3. Dynamic Directory Partitioning

We present the distributed metadata management scheme named Dynamic Directory Partitioning (DDP). Like DOIDFH strategy, we divided metadata into directory metadata and file metadata, and hierarchical directories metadata are maintained in each machine of MDS cluster, while files metadata are distributed on different MDS using a dynamic directory partitioning scheme (metadata of files within the same directory are assigned to individual MDS), simultaneously keeping relative dynamic load balancing among MDSs according to their performance differences. The same access control issue is employed as that of LH, and G-HBA [11] scheme can be employed for metadata query. DDP has a better trade-off between workload balancing and locality of reference, it not only avoids the large-scale metadata migration according to the directory renaming, but also improves the effectiveness of MDS caching.

3.1 Metadata Classification and Management

As in traditional file system, file metadata holds file name, ownership, access attributes and storage locations of the file data or object, while directory metadata lists all the files and subdirectories in that directory, and access control permissions. There are over half of metadata operations are related to directory metadata, and only several percents of them are for directory update [1]. In DDP scheme, the global unique identifier for each directory is assigned by the system at the time of creating the directory, and the directory identifier (DirID) will never change even when the directory is renamed. Each file has an identifier which is composed of parent directory DirID and filename. Directory metadata and file metadata are separately managed in DDP. All directory metadata are maintained in each server of MDS cluster, and files metadata are maintained in different MDSs using a dynamic partitioning scheme, while exploiting directory locality. Figure 2 shows the metadata distribution illustration of a directory among MDS cluster.

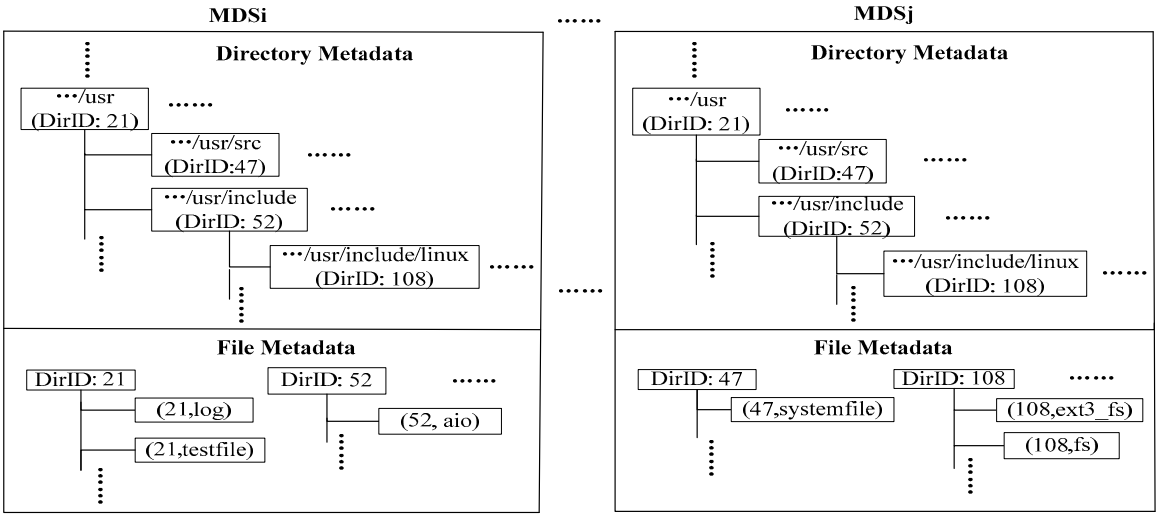

Figure 2. Example of metadata distributions among MDS cluster

When the hierarchical directories metadata is updated, the update message is broadcast to all the other MDSs to maintain the consistency of replicas of the hierarchical directories metadata.

As directory metadata and file metadata are separately managed in this scheme, we can avoid metadata migrations among MDS cluster after renaming a directory, and reduce network overhead induced by some directory operations for servers are able to process requests without communicating with other nodes.

3.2 Dynamic Partitioning Scheme of File Metadata

As the workload changes all the time, static distribution of metadata, although carefully designed, can not always achieve the best performance. We adopt a dynamic load balancing mechanism: Dynamic Relative Load Balancing Strategy to dynamically adjust the metadata distribution to maintain an optimal distribution of the load. The key idea is that the MDS nodes periodically exchange heartbeat messages including a description of their current load level. Then busy MDS nodes transfer part of the metadata of their own to non-busy nodes, and popular files metadata know as hot-spots, are replicated on non-busy nodes.

3.2.1 MDS performance. We simply abstract the MDS cluster as a set of N servers, identified by $MDS_1, MDS_2, ..., MDS_N$. Let C_i represents the capacity of MDS_i, and C represents the total capacity of the MDS cluster. We can measure the performance of the server using weighted sum of five indexes: CPU frequency, memory capacity, network throughput, disk I/O throughput and disk capacity. Assume CPU frequency, memory capacity, network throughput, disk I/O throughput and disk capacity of MDS_i are represented by $V^i_{CPU}, V^i_{mem}, V^i_{net}, V^i_{I/O}$ and V^i_{disk} respectively. Correspondingly, $W_{CPU}, W_{mem}, W_{net}, W_{I/O}$ and W_{disk} represent their weights respectively. Then we can measure the capacity of MDS_i and the total capacity of the MDS cluster: $(0 \le i \le N-1)$

$$C_i = V^i_{CPU} \times W_{CPU} + V^i_{mem} \times W_{mem} + V^i_{net} \times W_{net} + V^i_{I/O} \times W_{I/O} + V^i_{disk} \times W_{disk},$$

weights satisfying: $W_{CPU}, W_{mem}, W_{net}, W_{I/O}, W_{disk} \ge 0$ and

$$C = \sum_{i=0}^{N-1} C_i .$$

3.2.2 Load. The load can be divided into metadata storage load and metadata access load. Let a metric of file number under a directory: H_s represent the metadata storage load caused by file metadata under a directory, and H_a represents the metadata access load caused by file metadata under a directory. Assume the period for MDS nodes to exchange heartbeat messages is T_h, and then we can describe the metadata access load that caused by file metadata under a directory using the following equation:

$$H_a = \exp(-bT_h) H_a^* + (1 - \exp(-bT_h)) \times counter, \quad 0 < b < +\infty .$$

Where H_a^* represent the value of H_a in last period. Unlike the counter which just stores the current access information of file metadata under a directory, H_a also takes past access information into account. b acts as a forgetting factor. The bigger the factor is, the less effect past access information will impose on current access information. The load caused by file metadata under a directory Q_L is computed using the following equation:

$$Q_L = H_s + H_a$$

We assume that there are M directories in the file system, and Q_L^j represents the load caused by the directory satisfying $DirID = j$ ($0 \le j \le M-1$). Let Q_M^i represents the load of MDS_i : $Q_M^i = \sum_j Q_L^j$,for all j , directory j on MDS_i (i.e. file metadata under directory j maintained by MDS_i); The total load caused by file metadata under all directories is computed as following:

$$Q = \sum_{k=0}^{M-1} Q_L^k$$

And we define a threshold Q_T for workload caused by file metadata under a directory Q_L , If it exceeds that threshold, the directory is a hot-spot; otherwise, it is not a hot-spot.

3.2.3 MDS Load Level. We can get the current CPU efficiency: R_{CPU}^i , memory utilization rate: R_{mem}^i , network bandwidth utilization: R_{net}^i , disk I/O bandwidth utilization: $R_{I/O}^i$, and disk utilization factor: R_{disk}^i of MDS_i by exchanging heartbeat messages periodically among MDS nodes. Let L_i represents the load level of MDS_i $(0 \le i \le N-1)$, and it is computed using the following equation: $(0 \le i \le N-1)$

$$L_i = R_{CPU}^i \times W_{CPU} + R_{mem}^i \times W_{mem} + R_{net}^i \times W_{net} + R_{I/O}^i \times W_{I/O} + R_{disk}^i \times W_{disk} .$$

Then we define two thresholds $T_{\max}$ and $T_{\min}$ for MDS load level $(T_{\min} \le T_{\max})$. If the MDS load level exceeds threshold $T_{\max}$, the MDS node is busy; else if the MDS load level lower than threshold $T_{\min}$, the MDS node is idle; otherwise, its load level is normal. It can avoid chattering phenomenon using these two thresholds in dynamic load balancing process.

We can describe the Dynamic Relative Load Balancing Strategy in algorithm as follow:

---

1. Initialize the load of the MDS cluster.
   1.1. Choosing the appropriate hash function H^* , and hash value of all *DirID* distribute evenly on $[l,u]$;
   1.2. Calculate $x_i = \sum_0^{i-1} C_i / C$, Let $x_0 = 0$, $x_N = 1$, then $y_i = (1-x_i)l + x_i u$, $(0 \le i \le N)$;
   1.3. Allot LP_k on MDS_i , if $H^*(k) \in (y_i, y_{i+1}]$, $(0 \le i \le N-1)$.
2. We can get current load indexes: R_{CPU}^i , R_{mem}^i , R_{net}^i , $R_{I/O}^i$ and R_{disk}^i of MDS_i by exchanging heartbeat messages periodically among MDS nodes, and calculate current load level L_i , $(0 \le i \le N-1)$;
3. Comparing current load level of MDS nodes: let $S_M \subseteq \{0,1,2,...,N-1\}$, $\forall i \in S_M$ satisfying $L_i > T_{\max}$, and $S_m \subseteq \{0,1,2,...,N-1\}$, $\forall i \in S_m$ satisfies $L_i < T_{\min}$; If $S_M \ne \varnothing$ and $S_m \ne \varnothing$, for all $i \in S_M$ and all directory j on MDS_i , do the follow steps:
   3.1 If $Q_L^j > Q_T$, then directory j is a hot-spot, and copy it to a non-busy MDS node MDS_j $(j \in S_m)$, and if a replica's popularity declines, and $Q_L^j < Q_T / N$, this replica should be freed;
   3.2 Else transfer part of the metadata of busy MDS nodes to non-busy nodes as following:
      3.2.1 Calculate the following equations to get x_i $(0 \le i \le N-1)$
      $Q = \sum_{k=0}^{M-1} Q_L^k = \sum_{i=0}^{N-1} Q_M^i = \sum_{i=0}^{N-1} x_i$, $x_i / C_i = Q / C$;
      3.2.2 Let $y_i = Q_M^i - x_i$, and define two subsets: $A, B \subseteq \{0,1,...,N-1\}$, $A = \{i \mid y_i < 0\}$, $B = \{i \mid y_i > 0\}$;
      3.2.3 Define a set operator: $sum(S) = \sum_{i \in S} y_i$, $S \subseteq \{0,1,...,N-1\}$, then for $i \in A$,find a subset $U \subseteq B$,satisfying $|y_i + sum(U)| \le |y_i + sum(O)|$, $\forall O \subseteq B$;
      3.2.4 For $j \in U$, find a set of directory on MDS_j , sum of Q_L of these directories approximate to y_j , and transfer file metadata under these directories to MDS_i ;
      3.2.5 Modify sets: $A = A - \{i\}$, $B = B - U$, if $A \ne \varnothing$ and $B \ne \varnothing$, goto step 3.2.3;
   Else if $S_M \ne \varnothing$ and $S_m = \varnothing$, the MDS cluster is overloaded, add a new MDS node to balance the load.
   Else if $S_m = \{0,1,2,...,N-1\}$, the MDS cluster is idle, delete a MDS node from the MDS cluster.
4. After a period time T_h , goto step 2.

In Dynamic Relative Load Balancing Strategy, we can redistribute the file metadata among MDSs according to their performance and the current load level, the amount of moved metadata is minimal to keep the relative load balancing. The strategy can efficiently support addition, removal, and replacement of metadata servers. When new MDSs are added, they can just be treated as existing MDSs, but their Q_M values are 0. To moving metadata from leaving MDSs, it moves all metadata resided on these candidate MDSs to all other MDSs. At the same time, it tries to ensure that the relative load levels are still balanced among the remaining MDSs.

4. Evaluation

In this section, we will discuss and evaluate the effectiveness of special design choices of our dynamic metadata management scheme. Dynamic directory partitioning strategy was implemented within an event-driven simulation environment using Matlab 6.5, along with static subtree partitioning, static hashing of either files or directories, dynamic subtree partitioning and dynamic hashing metadata strategies to serve as points of comparison. Our simulations of scientific computing workloads on a homogeneous MDS cluster which contains 7 servers are based on a recent analysis of parallel file system traces on the website of SNIA.

We plan to evaluate specific performance trade-offs related to:

- Cache overlap penalties for duplicate prefixes across metadata nodes
- Performance and scalability benefits obtained by the MDS cluster load balancing
- Flexibility of the MDS cluster in efficiently supporting directory renaming, file and directory permission changes, and the addition, removal, and replacement of metadata servers

4.1 Metadata cache efficiency

The performance of metadata partitioning strategies is tightly linked to metadata cache efficiency. One of the primary factors affecting cache utilization is the need to cache prefix inodes of ancestor directories for the purposes of path traversal. Figure 3 shows percentage of cache devoted to prefix inodes as MDS cluster size scales. The overhead associated with caching prefix inodes for dynamic hashing is the highest among the four strategies, because directories metadata are scattered throughout the hierarchy and the prefix directory inodes to locate them must be replicated widely throughout the cluster. Dynamic subtree partitioning has slightly more prefixes than static subtree partitioning due to the re-delegation of subtrees nested within the hierarchy. And the metadata cache utilization for dynamic directory partitioning close to dynamic subtree partitioning.

4.2 Metadata distribution

When the access load is distributed evenly in the MDS cluster, then metadata distribution reflects load balance. Figure 4 shows the metadata distribution among 7 MDSs. File hashing and dynamic directory partitioning show the best balance of metadata. Metadata distributions among MDSs using static subtree partitioning and directory hashing are imbalanced, especially the distribution for static subtree partitioning.

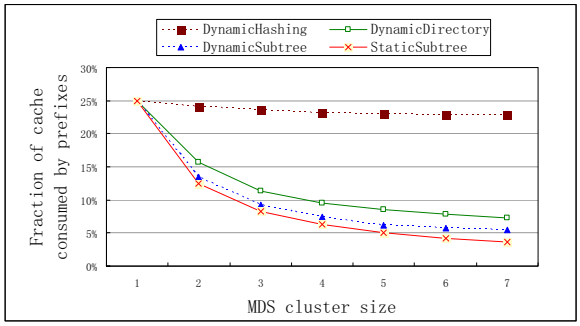

Figure 3. Metadata cache efficiency

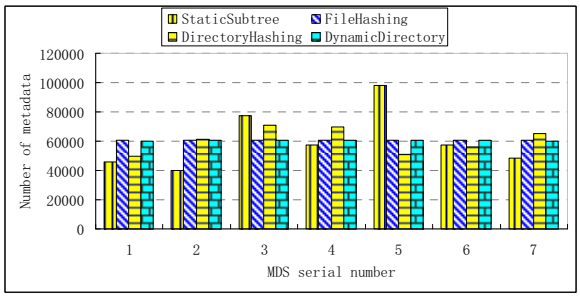

Figure 4. Metadata distribution among MDSs

4.3 Workload Equalization

In a real workload environment, the access load distributes unevenly for some directories or files could be more popular, or thousands of clients to access the same file at the same time or over a short period of time. To balance the workload, the metadata management scheme must dynamically redistribute metadata in MDS cluster. Figure 5 shows workload distribution of the MDS cluster which contains 7 MDSs. The dynamic directory partitioning strategy can periodically transfer file metadata under some directories, or their replicas when they are hotspots, from busy metadata servers to other non-busy metadata servers to balance the workload. Dynamic hashing also performs well in balancing the workload. While the workload balancing for dynamic subtree partitioning or file hashing is worse due to dynamic subtree partitioning is a coarse adjustment strategy and file hashing is a static strategy.

4.4 Metadata migration

Dynamic directory partitioning scheme maintains hierarchical directories to support the standard directory semantics of general-purpose storage systems. A consequence of this design choice is that certain operations: changing permissions on a directory or file, changing the name of a directory, removing a directory, and adding or removing MDSs in the cluster, can incur

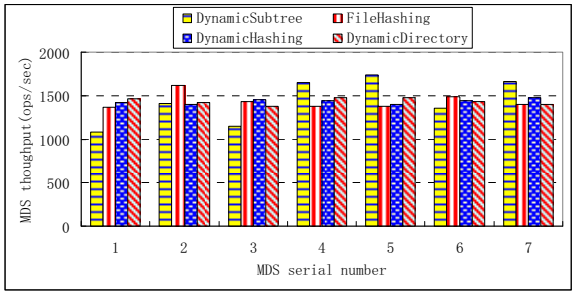

Figure 5. Workload distribution of MDS cluster

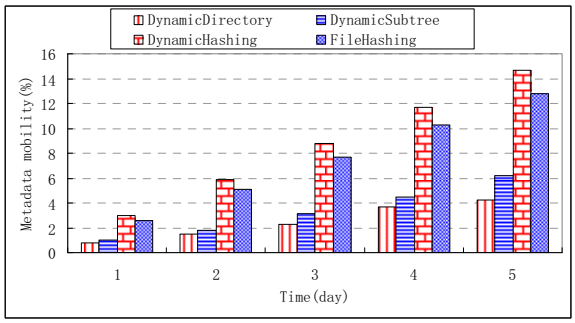

Figure 6. The changing metadata mobility

significant overhead, especially the overhead incurred by metadata migration. Figure 6 shows the changing metadata mobility of the distributed file system. Large amounts of metadata should be migrated for directory update in file hashing or dynamic hashing. Dynamic directory partitioning and dynamic subtree partitioning can avoid metadata migration incurred by directory update; there is only a small quantity of metadata need to be moved to balance the changing workload. And the amount of moved metadata in dynamic directory partitioning scheme is approximate to minimal when keeping load balancing of the MDS cluster as in algorithm for Dynamic Relative Load Balancing.

5. Conclusions

We have presented Dynamic Directory Partitioning (DDP), a novel dynamic metadata management mechanism for large distributed storage system. It employs the idea that directory metadata and file metadata are separately managed to avoid large numbers of metadata migrations among MDSs after renaming a directory. And we introduce a dynamic relative load balancing scheme to balance the changing workload and minimize the amount of metadata migration while maximizing overall scalability, and improve the effectiveness of MDS caching by exploiting directory locality. Our simulations indicate that the dynamic directory partitioning scheme can accommodate a variety of load distribution policies, avoid metadata access bottleneck, improve the effectiveness of cache on MDSs and minimize the amount of metadata need to be migrated when keeping the load balancing.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 60736013 & No.60573135, and the National High Technology Research and Development Program of China (863 Program) under Grant No. 2006AA01A106.

References

[1] D. Roselli, J. Lorch, and T. Anderson. “A compareison of file system workloads,” In *Proceedings of the 2000 USENIX Annual Technical Conference,* pp. 41–54, 2000.

[2] S. A. Brandt, E. L. Miller, D. D. E. Long, et al. “Efficient Metadata Management in Large Distributed Storage Systems,” In *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST’03),* pp. 290–298, 2003.

[3] S.A. Weil, K. T. Pollack, S. A. Brandt, et al. “Dynamic Metadata Management for Petabyte-scale File Systems,” In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC’04*), pp. 4-15, 2004.

[4] W. Li, W. Xue, J. Shu, et al. “Dynamic Hashing: Adaptive Metadata Management for Petabyte-scale File Systems,” In *23rd IEEE/14th NASA Goddard Conference on Mass Storage System and Technologies,* pp. 93–98, 2006.

[5] D. Feng, J. Wang, F. Wang, et al. “DOIDFH: an Effective Distributed Metadata Management Scheme,” *The 5th International Conference on Computational Science and Applications,* pp. 245–250, 2007.

[6] P. J. Braam. The lustre storage architecture. *http://www.lustre.org,* 2003.

[7] S. A. Weil, S. A. Brandt, E. L. Mille, et al. “Ceph: A Scalable, High-Performance Distributed File System”, In *Proceedings of the 7th symposium on Operating systems design and implementation,* pp. 307–320, 2006.

[8] O.Rodeh, A.Teperman. “zFS:A Scalable Distributed File System Using Object Disks,” In *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST’03),* pp.207–218, 2003.

[9] B. Pawlowski, C. Juszczak, P. Staubach, et al. “NFS version 3: Design and implementation,” In *Proceedings of the Summer1994 USENIX Technical Conference,*1994.

[10] M. Satyanarayanan, J. J. Kistler, P. Kumar, et al. “Coda: A highly available file system for a distributed workstation environment,” *IEEE Transactions on Computers,* 39(4):447–459, 1990.

[11] Y. Hua, Y. Zhu, H. Jiang, et al. "Scalable and Adaptive Metadata Management in Ultra Large-scale File Systems," The 28th International Conference on Distributed Computing Systems, pp. 403-410, 2008.