Replica Based Distributed Metadata Management in Grid Environment*

Hai Jin, Muzhou Xiong, Song Wu, and Deqing Zou

Cluster and Grid Computing Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, 430074, China
{hjin, mzxiong, wusong, deqingzou}@hust.edu.cn

Abstract. Metadata management is one of the key techniques in data grid. It is required to achieve two goals: high efficiency and availability. This paper presents a *Replication Based Metadata Management System* (RBMMS) as metadata server implemented in *Global Distributed Storage System* (GDSS). To address the above two goals RBMMS maintains a spares strongly connected graph to describe replica structure and relations among the replicas. The graph is used to propagate updating information and replica discovery in the process of replica addition and removal. Cache module is also implemented in RBMMS to further improve the performance of metadata access. The evaluation demonstrates that RBMMS attains high availability and efficiency of metadata management system.

1 Introduction

Metadata is the descriptive data and all the metadata in data grid [1, 2] compose the metadata catalog [3], which adopts some common structures to express metadata. All the metadata catalogs must satisfy the following requirements: 1) it should be a distributed and hierarchical structure system, such as LDAP [4]; 2) it should not breach current method of metadata expression. This paper does not discuss the expression of metadata, but focuses on the first requirement and presents a metadata management system RBMMS (*Replica based Distributed Metadata Management System*). Utilizing the solution of data replication [5-8] and web cache [9], RBMMS presents the concept of metadata replication and metadata cache, which uses LDAP directory server to store metadata and their replicas, also a cache module is added into metadata catalog. RBMMS achieves the following goals: 1) it provides multiple metadata replicas to support high availability; and 2) replica can be added or deleted dynamically. In order to achieve the above goals, RBMMS uses sparse strongly connected graph to express the metadata replica structure and the relationship among the metadata replicas.

* This paper is supported by National Science Foundation of China under grant 60125208 and 90412010, ChinaGrid project from Ministry of Education of China, and Hubei Natural Science Foundation under grant 2004ABA053.

Metadata management servers must work cooperatively to adapt to the dynamic environment in order to improve system performance in grid environment. Metadata catalogs should be organized hierarchically and symmetrically. Replica can be read and written anytime, and updating operation is to be propagated in the order of the graph. The replicating strategy always selects a nearest metadata server to the client to serve the access request and achieve high performance. The replica sparse strongly connected graph not only provides the sequence of replica updating, but also discovers the state of replica when inserting or deleting replica. When probing some replicas unavailable, system deletes it from the graph and reorganizes the graph. The above mechanism can improve the system availability and avoiding system disordered when several replicas out of work. In addition, metadata cache further improves the system performance. We have implemented RBMMS in *Global Distributed Storage System* (GDSS) [14], which is a storage middleware integrating distributed heterogeneous storage resources and providing global unified data view for users in grid environment.

This paper is organized as follows. Section 2 introduces related work. Section 3 discusses the RBMMS architecture design. Section 4 introduces the work mechanism of replica and cache in RBMMS. Section 5 analyzes the performance and the paper concludes in section 6.

2 Related Works

High performance and reliable metadata service becomes the goal of many projects. The *Storage Resource Broker* (SRB) [10] from the San Diego Supercomputing Center and its associated *Metadata Catalog* [11] provide metadata and data management services. SRB supports a logical name space that is independent of physical name space.

The *Replica Metadata* (RepMec) [13] catalog developed by the European Data Grid's Reptor project is built upon the Spitfire database service. The RepMec catalog stores logical and physical metadata. Among other functions, this catalog is used within the EDG project to map user-provided logical names of data items to unique identifiers called GUIDs. RepMec is used in the Reptor system in cooperation with a replica location service.

The Legion project provides an object-oriented middleware infrastructure for distributed computing environments [12]. The *Legion File System* (LegionFS) provides a Basic File Object with object methods that resemble UNIX read, write and seek system calls. Replication provided by LegionFS by classes maps object identifiers to multiple physical object addresses.

3 RBMMS Architecture Design

To achieve high availability and efficiency of metadata service, this paper presents RBMMS that is based on a special replica structure. Each domain has several local metadata servers that replicate metadata information for each other. System creates cache for metadata to improve the access performance. There exists a global metadata server which provides metadata index to support metadata operations across domains.

The granularity of metadata replication in RBMMS is partition, which is a subtree in LDAP directory server. Figure 1 describes a tree in LDAP directory server divided into 4 partitions, that is *P-A*, *P-B*, *P-C*, and *P-D*. When creating metadata replica, RBMMS replicates one or more partitions.

Fig. 1. Relationship between complete directory tree and partitions. A directory tree is divided into several partitions which is the smallest unit in replicating.

The relationship among the replicas in RBMMS is described as a sparse strongly connected graph. The verge of the graph means that the two replicas are connected and the updating order is according to the direction of the verge. The partition replicas build up a strongly connected graph described in Figure 2 in which each node presents a partition replica. There are two types of replicas: one is master replica which is shadowed in Fig. 2 (*P-A*), and the others are slave replicas. All the replicas can be read or written anytime and can be operated as the same propagating protocol for replica updating. However the master replica is the starting point of replica discovery. In RBMMS the initial partition is its master replica. The number of master replica for each partition is one and is fixed in RBMMS.

Fig. 2. Replica sparse strongly connected graph for Partition A. The shadowed node *P-A* is the master replica.

RBMMS maintains replica strongly connected graph to manage distributed replicas, which is composed by the following components. GMS (*Global Metadata Server*) maintains the index information for local metadata servers in all the domains. LMS (*Local Metadata Server*) stores master metadata replica that contains the whole metadata directory tree. There is only one LMS in a domain. RMS (*Replica Metadata*

Server) maintains the slave metadata replicas and the number of RMS can be one or more. MC (*Metadata Cache*) caches hot metadata to improve the read/write efficiency.

4 Metadata Replica Management in RBMMS

Initially metadata is created in LMS and is appointed as the master replica. Through replica creation algorithm RBMMS creates slave metadata replicas in RMS. Replica can be deleted when the capacity of RMS is not enough to contain more replicas.

4.1 Metadata Replica Creation Strategy

Replica creation strategy considers when to replicate, which metadata to replicate, and to which storage resource the replica stores. We consider partition as the smallest unit when creating metadata replica, using PQ algorithm [15] to determine when to run the replicating engine and finally RBMMS puts the replica to the place nearest to users.

RBMMS divides the whole directory tree into several partitions dynamically and records partition index table in LMS, including root node of the partition, partition size, partition response time, access number and slave replica root node. Partition response time (T_{resp}) is the sum of partition service time (T_s), waiting time (T_w), and communication cost (T_c), that is $T_{resp} = T_s + T_w + T_c$. A threshold $T_{threshold}$ is set for partition response time. When $T_{resp} > T_{threshold}$, the RBMMS creates slave replica for this partition.

RBMMS is free to select two parameters, P and Q ($Q > P$). Time is divided into two parts: P and Q alternatively. The concrete algorithm description is as follows: (i) if $T_{resp} < T_{threshold}$ abort replica creation, else go to (ii); (ii) $\forall t \in P$, if $T_{resp} > T_{threshold}$ and $dT_{resp}/dt > 0$, go to (iv), else go to (iii); (iii) $\forall t \in Q$, if $T_{resp} > T_{threshold}$, then go to (iv), else abort replica creation; (iv) Create slave replica.

When the time creating slave replica is determined, according to the use's access mode LMS chooses an RMS which does not contain replica of this metadata. The access mode is limited in three modes: centralized mode, uniform mode, and random mode. With different access mode, we compare the two strategies: fastspread [16] and cascading [16]. The strategy of fastspread creates slave replica at all nodes that are in metadata access path. For cascading it creates slave replica in the upper level of metadata access path and extends to hierarchical structure. For consistency issue, the new replica must be added into replica connected graph.

4.2 Metadata Replica Deletion Strategy

This subsection introduces how to delete slave replica. In RBMMS the master replica will never be deleted except that user deletes the related data. The reason of replica deletion includes no enough disk capacity and expensive cost of maintaining replicas. RBMMS randomly selects 10 slave replicas periodically and checks the access number of them. The slave replica with the least access number will be deleted and the three with the least access number will be considered as the replicas that will be deleted in the next operation. The deletion operation performs until system gets enough disk capacity.

The neighbors of the deleted replica are notified to find a new replica as their neighbor from the slave replica and create new verges, so that the strong connectivity is ensured.

4.3 Management of Metadata Cache

Metadata Cache utilizing access locality is stored in MC, containing the hot metadata to improve the metadata read/write efficiency. Different from metadata replica, metadata cache may contain incomplete information. When receiving the read/write request from user, RBMMS first checks whether the required metadata is in cache. If does, it directly reads cache; if not, it reads from a LMS or RMS and also caches the metadata. In order to confirm the metadata consistency, all the write operation is performed in MC and later updated to metadata server, which can reduce the cost of network communication.

5 Experiments

This section evaluates the performance and availability of RBMMS. We first consider the performance of RBMMS in structure of a fixed single domain; and then evaluate the cost of dynamic reorganization of replica strongly connected graph when some of the replicas are unavailable. The experiment result includes disk capacity consumption for replica storing, response time of metadata service, time of replica creation and updating under the two strategies of replica creation with different access modes. Through the experiment results, RBMMS can choose the best strategy to manage metadata.

There is no cache in the following experiments. When the access mode is centralized mode, which means all the access in RMS3, the replica distribution under the two strategies are described in Fig. 3(i). The result is in Table 1, in which the response time of fastspread is 18% faster than that of cascading and also the capacity consumption of fastspread is 2 times larger than that of cascading. The distribution of metadata replica in Fig. 3(ii) describes the access mode in uniform mode in which access to Partition A is performed in RMS3 and RMS4. Under this access mode, the response time of fastspread is 21% faster than cascading and the capacity consumption of fastspread is 3 times larger than that of cascading, shown in Table 1. The distribution of metadata replica in Fig. 3(iii) describes the access mode in random mode where access to Partition A is performed randomly. Under this access mode, the response time of fastspread is 8% faster than cascading and the capacity consumption of fastspread is 2 times larger than that of cascading, shown in Table 1.

From the above three results we can see that the response time of fastspread is faster than cascading in all cases, but also costs more disk capacity. Therefore, we decide that RBMMS utilizes fastspread under centralized access mode and uses cascading under uniform or random access mode. According to this decision, we compare the performance of RBMMS with and without MC, respectively. We conclude that adding cache into RBMMS, performance improves about 20%, shown in Fig. 4.

Table 1. Performance comparison for two strategies with centralized, uniform and random access mode

Strategy	Capacity consumption (MB)			Response Time (ms)			Replica creation Time (min)			Replica updating Time (min)		
	Central	Uniform	Random	Central	Uniform	Random	Central	Uniform	Random	Central	Uniform	Random
Fast-spread	4.4	6.5	8.7	10.4	10.8	10.4	6	6	6	40.4	60.7	80.9
Cascading	2.2	2.2	4.4	12.6	13.7	11.3	6	6	6	40.4	20.2	40.5

Fig. 3. The performance comparison of two strategies under the centralized, uniform and random access modes

Fig. 4. Performance comparison of RBMMS with cache and without cache

When a RMS is unavailable for a long time (for example 0.5 weeks), replica stored in it is shipped to other RMS to substitute the unavailable replica and reorganize replica strongly connected graph. Because of the size of metadata is much less than those of data, the performance of reorganization can achieve high efficiency. In the following experiment, we still configure RBMMS with a single domain structure and cache works. The initial system is shown in Fig. 5(i), in which *P-A* is the master replica in

LMS1 and $P-A1 \sim P-A4$ are the slave replicas stored in RMS1~RMS4 respectively. If RMS1 is unavailable, RMS3 and RMS4 get information about RMS2 from LMS1 because RMS2 is nearest to them. RMS3 and RMS4 update their information of RMS and set RMS2 as their father node. LMS1 is notified about all the update information. LMS1 notifies RMS2 that sets RMS3 and RMS4 as its son nodes. The new reorganized graph is shown in Fig. 5(ii). In our experiment, the size of metadata in RMS1 is about 6MB (with metadata number 33900), and it takes about 7 minutes to complete the whole process.

Fig. 5. (i) Initial structure of RBMMS system, (ii) Structure of RBMMS after reorganization when RMS1 is out-of-work

6 Conclusion and Future Work

In this paper, we present a replica based solution to implement metadata management in grid environment. We use a spares strongly connected graph to construct and maintain the relationship among replicas. Through that, the RBMMS system achieves a good result of availability and efficiency. In addition, we implement cache component that can farther improve the performance of RBMMS.

It is a challenge to test RBMMS in dynamic wide area network. In the future work we will present solutions to make RBMMS satisfied in such environment. How to utilize sparse strongly connected graph to maintain the consistency of replica is another issue in the system, which will also be resolved in the future.

References

1. I. Foster, "The Grid: A New Infrastructure for 21st Century Science", *Physics Today*, 2002, 55(2): 42.
2. B. Allcock, J. Bester, J. Bresnahn, A. Chervenak, I. Foster, and C. Kesselman, "Data Management and Transfer in High Performance computational Grid Environments", *Parallel Computing*, 2002, 28(5): 749-771.
3. I. Foster and C. Kesselman, "Computational Grids", *The Grid - Blueprint for a new Computing Infrastructure*, San Francisco: Morgan Kaufmann Publisher, 1999, pp.15-51.
4. G. von Laszewski and I. Foster, "Usage of LDAP in Globus", ftp://ftp.globus.org/pub/globus/papers/ldap_in_globus.pdf

5. C. Ferdean and M. Makpangou, "A Scalable Replica Selection Strategy based on Flexible Contracts", *Proceedings of Third IEEE Workshop on Internet Applications*, San Jose, California, 2003.
6. H. Lamahamedi and B. Szymanski, "Data replication strategies in grid environments", *Proceedings of Fifth International Conference on Algorithms and Architectures for Parallel Processing*, Beijing, 2002.
7. K. Ranganathan and I. Foster, "Design and Evaluation of Replication Strategies for a High Performance Data Grid", *Proceedings of International Conference on Computing in High Energy and Nuclear Physics*, Beijing, 2001.
8. M. Karlsson and C. Karamanolis, "Choosing Replica Placement Heuristics for Wide-Area Systems", *Proceedings of 24th International Conference on Distributed Computing Systems*, Hachioji, Tokyo, Japan, 2004.
9. S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A decentralized peer-to-peer web cache", *Proceedings of 21th ACM Symposium on Principles of Distributed Computing*, Monterey, California, 2002.
10. C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker", *Proceedings of CASCON'98 Conference*, 1998.
11. MCAT, MCAT – A Meta Information Catalog (Version 3.0).
12. B. White, M. Walker, M. Humphrey, and A. Grimshaw, "LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Application", *Proceedings of SC'2001*, November 2001.
13. L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, "Replica Management in Data Grids", *Global Grid Forum 5*, 2002.
14. H. Jin, L. Ran, Z. Wang, C. Huang, Y. Chen, R. Zhou, and Y. Jia, "Architecture Design of Global Distributed Storage System for Data Grid", *High Technology Letters*, Vol.9, No.4, December 2003, pp.1-4.
15. B.-D. Lee and J. B. Weissman, "An Adaptive Service Grid Architecture Using Dynamic Replica Management", *Proceedings of 2nd International Workshop on Grid Computing*, Denver, Colorado, 2001.
16. K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid", *Proceedings of the International Workshop on Grid Computing*, Springer-Verlag, 2001.