Adaptive Replica Management for Large-scale Object-based Storage Devices

Qingsong Wei Wujuan Lin Yong Khai Leong
Data Storage Institute, Network Storage Technology Division
Email: {WEI_Qingsong, LIN_Wujuan, YONG_Khai_Leong}@dsi.a-star.edu.sg

Abstract

Replica management is basic and challenging issue for distributed storage system designer. The objective of this paper is to dynamically create, migrate and delete replicas among nodes in response to changes in the access patterns.

This paper presents an Adaptive Replica Management Model for large-scale Object-based Storage Devices (OSDs). The model expresses availability and consistency maintenance cost as functions of replica number and suggests lower bound and upper bound on replica reference number based on file availability requirement and available network bandwidth. The model can adapt to the changes of environment and maintains a rational number of replica, which not only satisfies object availability, improves access efficiency and balances overload, but also reduces bandwidth requirement and keeps the whole storage system stable. Our experimental evaluation results demonstrate that our model can perform well for system reliability and performance.

1. Introduction

As a new generation high-performance distributed storage system, object-based storage system is being developed to support high-performance computing environments, which require strong system scalability and reliability. Rather than relying upon a few very large storage arrays, the petabyte-scale object-based storage system have thousands of self-contained Object-Based Storage Devices (OSDs) [1,2], working together to provide low cost, large capacity, high-performance storage service with aggregate storage bandwidth exceeding 100GB/s [3,4].

In a large-scale object-based storage system, files are usually stripped into multiple objects across OSD clusters to improve the system I/O throughput. However, OSDs may be unreachable due to network or node failure in a heterogeneous large-scale storage system. If one of the objects is unavailable, so as the whole file. Improving object availability in such an object-based storage system environment thus becomes a challenging issue for the system designers. Some high performance applications such as scientific application strongly require data to be available when they need it, at least with high probability. Data replication is a well-known solution to increase data availability (or fault tolerance), reduce access latency, balance the workload, etc. If multiple copies of object exist on different OSDs, then the chances of at least one copy being accessible increases. Aggregate data access performance will also tend to increase, and total network load will tend to decrease, if replicas and requests are reasonably distributed [5].

While replication has advantages as mentioned above, it also has significant costs to maintain the data consistency. If one replica is modified, all the replicas of an object must be updated, consuming large amounts of storage and network bandwidth that is relatively rare when comparing to aggregate storage system bandwidth 100GB/s. Although adding replicas of an object can improve availability and balance workload, it will increase consistency maintenance cost, which may essentially result in communication congestion of underlying network. Unreasonable number and distribution of replicas may not improve the whole system performance, but bring unnecessary spending instead.

This paper presents an Adaptive Object Replica Management Model for OSD clusters, in which the availability and consistency maintenance cost are considered as functions of object replica number. Our objective is to dynamically manage replica and maintains a reasonable number of replica, so as to satisfy file availability, improve access efficiency, balance the workload, and reduce bandwidth requirement as well. Our experimental results demonstrate the effectiveness of the proposed adaptive replica management model with respect to the system reliability and performance in terms of request access latency.

The rest of this paper is organized as follows. Section 2 provides an overview of related works in the literature. Section 3 analyzes the basic issues when designing the object replication scheme. We present our adaptive replica management model in Section 4. Section 5 gives implementation details of the model, and we present our evaluation and measurement results in Section 6. We

summarize this paper with the conclusions and possible future work in Section 7.

2. Related works

Replication management scheme has direct relationship with application type and underlying network environment. Data Grid manages large numbers of read-only scientific data and provides data sharing for globally distributed user communities. It improves data access efficiency through replication, but does not consider the consistency maintenance cost. To save latency and bandwidth, K.Ranganathan and I.Foster presented identifying dynamic replication strategies for Data Grid in [6], by applying different replication strategies for different kinds of access patterns without considering consistency issues.

The most common distributed storage systems, such as Napster [7] and Freenet [8], dynamically manage replicas based on file popularity degree, in which once access frequency to a file exceeds a threshold, a new replica of the file will be created. This replication management scheme is based on the assumption that underlying network bandwidth is unlimited and does not consider the costs of replica creation. This scheme can acquire satisfactory performance when bandwidth resource is enough. However, in WAN and wireless network environment, bandwidth resource is very limited and the underlying assumption of high speed networks will no longer hold.

Kavitha Ranganathan proposed a dynamic model-driven replication scheme in [9]. In the scheme, each peer in the P2P storage system runs a model to determine how many replicas of a file are needed to maintain desired availability. More specifically, each peer applies this model to the information about system state and file replication status to determine when and where new replicas should be created. This scheme can ensure file availability in dynamic P2P environment, but again, it does not consider access efficiency and consistency maintenance costs.

3. Issues of Object Replication Scheme

In this section, we will discuss several issues to further understand how replica influences object availability and system performance. We take the following experiments in our OSD prototype.

3.1. Availability

In the object-based storage system, file may be stored in the OSD cluster with two manners: without stripping and with stripping, as shown in Figure 1. In the case of without stripping, a file will be stored in an OSD as a single object, while in the case of with stripping, one file may be mapped to multiple objects and stored in different OSDs for parallel data access. Now, let us assume that the probability of each OSD availability is p ($0<p<1$). For the case of without stripping, the file availability is p. While for the case of with stripping, let say file is stripped into n ($n>0$) objects distributed into n different OSDs, the file availability is p^n and obviously, $p>p^n$. So file to object stripping will decrease file availability and it is necessary to place multiple object replicas to ensue file availability.

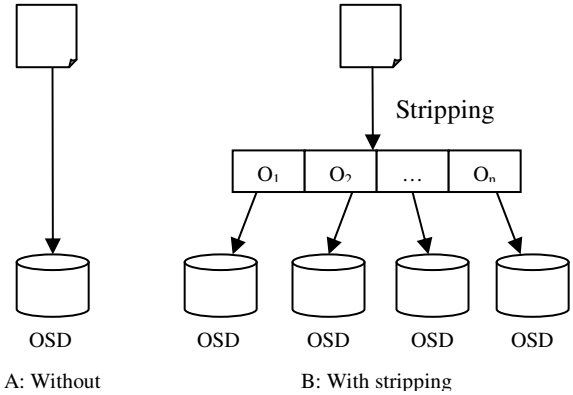

Figure 1. File Stored in the OSD cluster: (a) without stripping and (b) with stripping

We test the relationship of availability and replicas number when the OSD nodes keep a certain online ratio. A given file is stripped into 3 objects and each object has several replicas distributed across OSDs. Figure 2 plotted the result when node online ratio is 0.8 and 0.9.

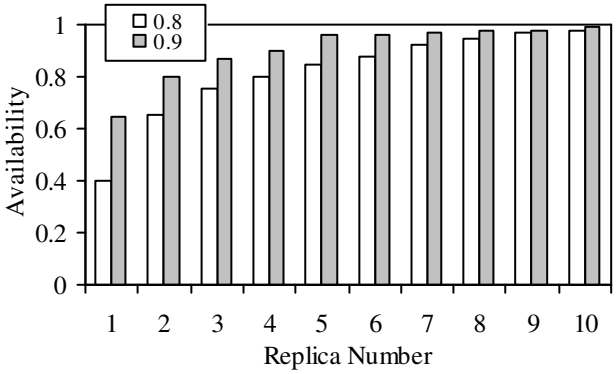

Figure 2. Availability as function of Replica Number

From Figure 2, we observe that file availability increases along with the increase of replica number. When replica number reaches a certain point, the file availability is equal to 1, and there is no meaning to add in more

replicas. The higher the node online ratio, the less replica number it needs for file availability equal to 1.

Therefore, given a certain file availability requirement, we can maintain minimum replica number to ensure the file availability. Adding more replicas will not improve the file availability.

3.2. Consistency Maintenance Overhead

To study consistency maintenance overload, we use trace-driven method to analyze network traffic along with replica number varying when file updates with a certain frequency, as shown in Figure 3.

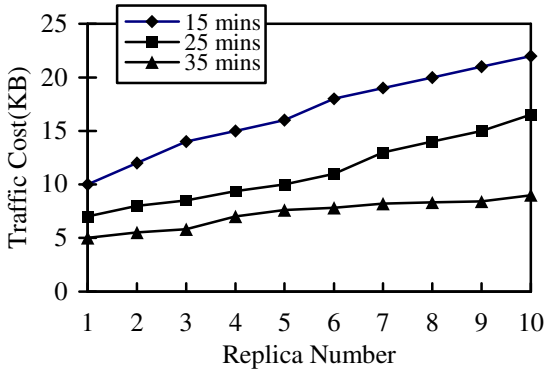

Figure 3. Traffic Cost varies with replica number with different update frequency: 15 minutes, 25 minutes, and 35 minutes

As expected, when object update frequency is certain, network traffic increases linearly along with replica increasing. And when replica number is certain, network traffic increases along with object update frequency increasing.

In a network storage system, the network bandwidth resource is very limited and crucial to the overall performance. Although more replicas mean high availability and high performance, it also consumes more network resource to maintain data consistency, which may lead to tremendous network resource consumption and essentially decrease the data access performance.

From the above experiment and analysis, we can find that there is a trade-off between data availability/efficiency and data consistency. In order to improve availability and performance, we need to add more replicas, while need to pay more cost for the data consistency. In view of these issues, we designed a dynamic replica management mechanism to consider the above two factors. Our mechanism not only improves object availability and access efficiency, but also reduces object consistency maintenance overload.

4. Adaptive Replica Management Model

Our proposed model will answer following two key questions: how many replicas the system should keep at least to maintain certain object availability? How many replicas the system can support at most to maintain object consistency under a certain network environment?

4.1. Availability

To determine the number of replicas that guarantees the required availability of an object, we need to identify the system parameters that affect data availability. The reachable probability or failure rate of OSDs in the network is the key factor of object availability.

Followings are some definitions that will be used throughout the rest of this paper.

Node Available: refer to the event of node reachable, denoted as NA, and P(NA) is the probability of node available.

Node Unavailable: refer to the event of node unreachable, denoted as $\overline{NA}$. $P(\overline{NA})$ is the probability of node unavailable, and $P(\overline{NA}) = 1 - P(NA)$.

Object Available: refer to the event of object O_i available, denoted as OA_i, and $P(OA_i)$ is the probability of the object O_i available.

Object Unavailable: refer to the event of object O_i unavailable, denoted as $\overline{OA_i}$. $P(\overline{OA_i})$ is the probability of the object O_i unavailable, and $P(\overline{OA_i}) = 1 - P(OA_i)$.

File Available: refer to the event of file available, denoted as FA, and P(FA) is the probability of the file available.

File Unavailable: refer to the event of file unavailable, denoted as $\overline{FA}$. $P(\overline{FA})$ is the probability of the file unavailable, and $P(\overline{FA}) = 1 - P(FA)$.

Let say file is stripped into m objects marked as F= $\{O_1, O_2, \ldots, O_m\}$, and the M objects are distributed into n OSD nodes marked as $H=\{H_1, H_2, \ldots, H_n\}$. Object O_i has R_i replicas and unreachable probability of OSD H_j is p_j. In the object-based storage system, all OSDs are independent from each other.

As for object O_i which has R_i replicas across R_i OSD nodes, if all the R_i OSDs are not available, the object O_i will be not available. So we have,

$$P(\overline{OA_i}) = P(\overline{NA}_1 \times \overline{NA}_2 \times \ldots \times \overline{NA}_{R_i})$$

The OSD nodes are independent, we have,

$$P(\overline{OA_i}) = P(\overline{NA}_1) \times (P(\overline{NA}_2) \times \ldots \times P(\overline{NA}_{R_i}) = \prod_{j=1}^{R_i} p_j$$

To get the whole file F, we must get all the m objects. Any object unavailable will cause file unavailable. So,

$$P(\overline{FA}) = P(\overline{OA}_1 \bigcup \overline{OA}_2 \bigcup ... \bigcup \overline{OA}_m)$$
$$= \sum_{i=1}^{m} P(\overline{OA}_i) - \sum_{1 \le i \prec j \le m} P(\overline{OA}_i \bigcap \overline{OA}_j) +$$
$$... + (-1)^{m-1} P(\overline{OA}_1 \bigcap \overline{OA}_2 \bigcap ... \bigcap \overline{OA}_m) \qquad (1)$$

Therefore, the file availability is:

$$P(FA) = 1 - P(\overline{FA})$$

Generally, there are two types of distribution policies: objects independent and objects dependent. Here we only consider the first distribution policy in our object-based storage system.

In this case, objects are independent and object O_i unavailable can not result in object O_j unavailable. It satisfies following condition:

$$P(\overline{OA}_i \bigcap \overline{OA}_j) = P(\overline{OA}_i) \times P(\overline{OA}_j) \qquad i \neq j$$

To simplify model, let say file F is divided into m objects marked as F={O_1,O_2,…., O_m}, and distributed across n OSD nodes. Each object has k replicas. The probability of a node down is p.

Now for object O_i, which has k replicas across k OSDs, if all of the k OSDs are not available, the object O_i will be unavailable. So the probability of object O_i unavailable is:

$$P(\overline{OA}_i) = p^k$$

$$P(\overline{OA}_1) = P(\overline{OA}_2) = ... P(\overline{OA}_m) = p^k$$

According to the formula (1), the probability of file unavailable is:

$$P(\overline{FA}) = P(\overline{OA}_1 \bigcup \overline{OA}_2 \bigcup ... \bigcup \overline{OA}_m)$$
$$= C_m^1 p^k - C_m^2 (p^k)^2 + ... + (-1)^{m+1} C_m^m (p^k)^m$$
$$= 1 - (1 - p^k)^m$$

And hence, the probability of file available is:

$$P(FA) = 1 - P(\overline{FA}) = (1 - p^k)^m$$

Assuming the expected availability for file F is A_{expect}, to ensure the availability needed for a given file, we need:

$$(1 - p^k)^m \ge A_{expect} \qquad (2)$$

The minimum value of k can be calculated from the above formula for any given desired availability. For example, for p=0.1, A_{expect}=0.8 and m=3, the model suggest a minimum of replicas for this availability is 2. If p=0.1, A_{expect}=0.99 and m=3, the model suggests a minimum of replicas for this availability is 3.

Once OSD knows the minimum number of object replica k_{min} for an object, it can dynamically manage replica according to existent replica of this object. Let us assume current replica number of an object is r and r is less than k_{min}, then the OSD knows that it has to create (k_{min} – r) more replicas of the object and distribute them to OSD clusters.

4.2. Consistency maintain overload

In the Object-based Storage System, some OSD-SCSI commands ([10]) such as writing object, setting/modifying object attributes will result in object consistency maintenance. Once an object or object replica in an OSD is modified, the OSD will notify other OSDs storing this object to modify object data and update object meta-data.

In the Object-based Storage System, intelligent OSD can track object access, and record access frequency. If access to an object exceeds access frequency threshold, a new replica object will be created in another OSD.

The total overload of adding a replica includes momentary replica transfer overload and upper replica consistency maintenance overload. The influence of replica consistency maintenance overload is long-term. This paper studies the relationship of consistency maintenance overload and replica number. If current replica number is k-1, we consider the overload of maintaining k replica consistency. Followings are some definitions:

k be the replica number of each object,

F_u be object update frequency,

S_{obj} be object size in bytes,

L_{msg} be length of metadata update message,

$B_{traffic}$ be the network traffic overload of an object's consistency maintain,

B_{sys} be average network bandwidth that system can provide.

α is a adjustable factor to calculate available network bandwidth.

Then, updating the k object replica will cost:

$$S_{obj} \times k$$

Overload of refreshing metadata is:

$$L_{msg} \times k$$

Then, one update operation will cost:

$$S_{obj} \times k + L_{msg} \times k$$

The overload of file update in a unit time is:

$$B_{traffic} = (S_{obj} + L_{msg}) \times k \times F_u$$

To ensure the system performance with a given network environment, we need:

$$B_{traffic} \le \alpha B_{sys}$$

Then:

$$(S_{obj} + L_{msg}) \times F_u \times k \le \alpha B_{sys}$$

Combining the analysis in section 4.1, we can get the following formula, describing our adaptive replica management model:

$$\begin{cases} [1-p^k]^m \geq A_{except} \\ (S_{obj} + L_{msg}) \times F_u \times k \leq \alpha B_{sys} \end{cases} \quad (3)$$

Each OSD runs the model and dynamically adjust the number of object replica. From the above model, the minimal replica number k_{min} and maximum replica number k_{max} can be calculated, which are the criterion for system to add replica or delete replica dynamically.

- **When to replicate:** Node unreachable and access frequency increasing will evoke replication. As for the first case, if some OSDs storing replica is unavailable and the current replica number is less than the minimal replica number k_{min}, adding a new replica into the OSD clusters is needed.

  On the other hands, the large volume of requests arriving at popular objects can result in saturating even the most powerful OSD node, especially during peak hours. If the average access frequency for an object exceeds the threshold for replication, a new replica will be added to a selected OSD for the purpose of workload balance.
- **Where to replicate:** OSD weight is the criterion to select OSD to store replica and serve object request in term of CPU performance, disk utilization, network interface speed, current workload and so on. The lightest weight OSD will be selected to accept replica. B+trees are an attractive choice for managing index and list structures because they maintain records in sorted order and scale efficiently in both time and space. OSD Weight B+Tree indexed by the OSD weight can rebuild in case of changes of environment and return lightest weight or heaviest weight OSD quickly.
- **When to delete:** if the current replica number is more than the maximal replica number k_{max} and the average access frequency for an object does not exceed the threshold for replication, a selected replica will be marked as invalid.

  Access to an object will not always keep high. Once the average access frequency for the object drops down and becomes less than the threshold for deletion and current replica number is more than k_{min}, the extra replicas will be marked as invalid gradually.

  All invalid replicas will not be involved in consistency update. Once the access frequency rebounds, we will validate the invalidated replica if its version is enough new. The other invalid replicas will be eventually overwritten according to local storage policy.
- **Where to delete:** which replica will be deleted is based on OSD Weight. The heaviest weight OSD will be selected to delete replica.

5. Implementation

To implement above replica management policy, we introduce Minimal Replica Number as object attribute.

In our implementation, file storing follows three steps. Firstly, after creating a file entry in Metadata Server (MDS) and setting file availability attributes, client strips a file into multiple objects and calculates the minimal object replica number k_{min} according to file availability set by user. Secondly, original objects will be created in Master OSD and the k_{min} is set as object attribute of Minimal Replica Number. Finally, the Master OSD replicates k_{min} -1 objects to Slave OSDs based on the OSDs weight, fills in the Object Replication Table and transfers it to MDS, as shown in Figure 4.

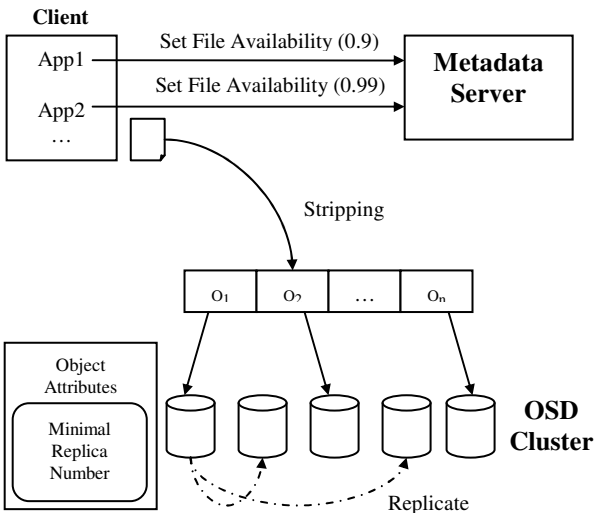

Figure 4. File is striped and replicated across OSD clusters

Figure 5 shows the proposed framework of dynamic replica management in the OSD. This framework consists of following components reside in the OSD:

1) **Request Portal:** Access to an object will firstly go to its Master OSD and then will be served by Master OSD or forwarded to Slave OSDs based on OSD weight to balance the workload. Request Portal processes the incoming client request.
2) **Access counter:** counts the request for an object and calculates the average access frequency as following equation:

$$Access_{\text{average}} = Access_{total} \Big/ Num_{replica} \quad (4)$$

Where, $Access_{average}$ represents per-OSD average access frequency, $Access_{total}$ stands for total access amount to an object in unit time, and $Num_{replica}$ represents active replica number.

If the average access frequency for an object exceeds the Threshold for Replication $T_{replica}$, it sends a signal to Replica Manager.

3) **Request scheduler:** distributes client request across Slave OSDs based on OSD Weight.
4) **Status monitor:** detects OSD failure, updates OSD Weight and notifies Replica Manager.
5) **Replica manager:** decides if, when, and where replicas should be created or invalidated according to Object Replica Table and OSD Weight Tree.

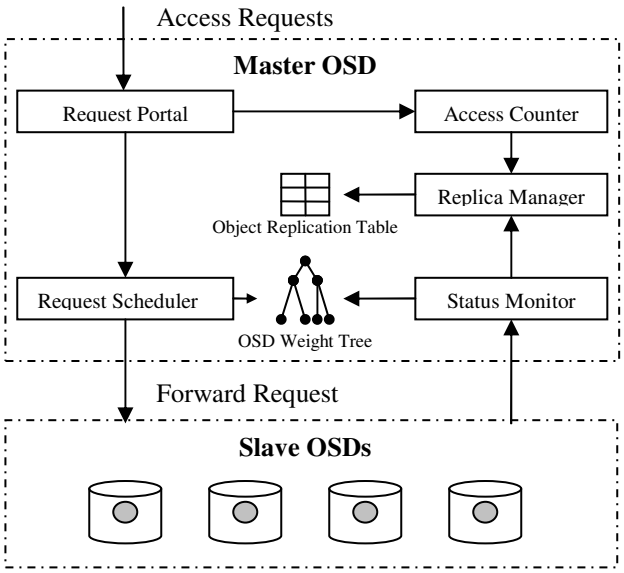

Figure 5. Framework of dynamic replica management in OSD

To avoid extra replicas being created in the event of more than one OSD replicating the same object simultaneously, Master OSD will be assigned by MDS during file creation. If a Master OSD fails, MDS can detect this failure in time and will assign a Slave OSD to play the role of Master OSD.

Before deletion, Master OSD will always re-calculate the average access frequency as following formula:

$$Access_{\text{average}} = Access_{total} \Big/ \left(Num_{replica} - 1\right) \qquad (5)$$

If the new average access frequency is less than the previous average access frequency, then a replica will be marked as invalid. Otherwise does not invalidate a replica to avoid average access frequency increasing after deletion.

6. Evaluation

We have evaluated our adaptive replica management model on our OSD prototype. Efficiency and correctness of the model will be tested in two prospects: reliability and performance.

6.1. Reliability

In our experiments, file is stripped into 3 objects. Each object has multiple replicas distributed across the OSD nodes. Our model computes the minimum number of replicas that are necessary to achieve an expected availability in the presence of node failures.

One way to check the accuracy of our model is to fix the number of replicas existent in the system at any time and measure data availability. Figure 6 compares the numbers of replicas corresponding to required availability values. For example, for the probability of 0.1 of OSDs being down and a required availability of 0.8, the model requires 2 replicas. When the system maintains the number of replicas per object at 2, we measure the average availability per file to be around 0.72. Though the model predictions are not this accurate at all points in the graph, the simulations validate the general trend pointed to by the model.

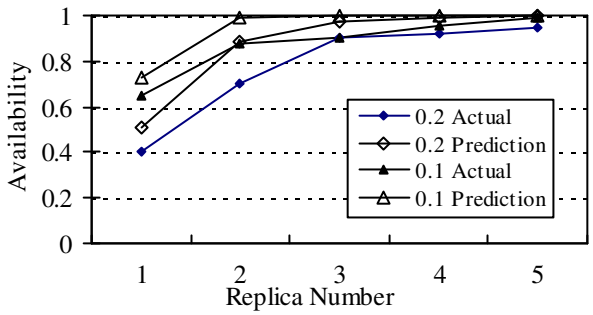

Figure 6. Model prediction versus actual behavior for different values of node reliability

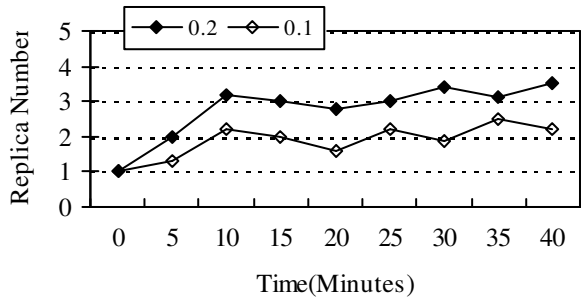

Figure 7. Dynamic replication for different probabilities of OSD being down at 0.1 and 0.2

We then test the dynamical feature of the model. We set the expected availability as 0.8 and started the tests with only one replica per object. As shown in Figure 7, the number of replicas is maintained at a constant level, thus ensuring constant data availability over time.

6.2. Performance

In the test, we set file access frequency threshold T=20 and initially maintain one replica per object. We can get the average access latency and average replica number by increasing file access frequency. The test result is shown in the Figure 8 and Figure 9.

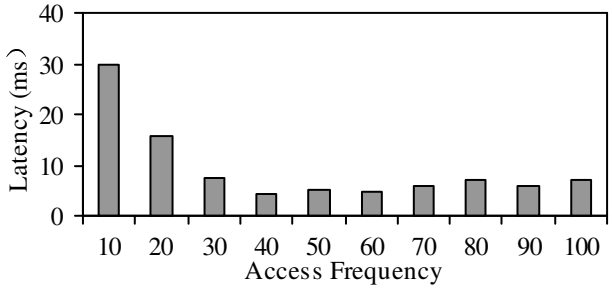

Figure 8. Access latency varies with access frequency

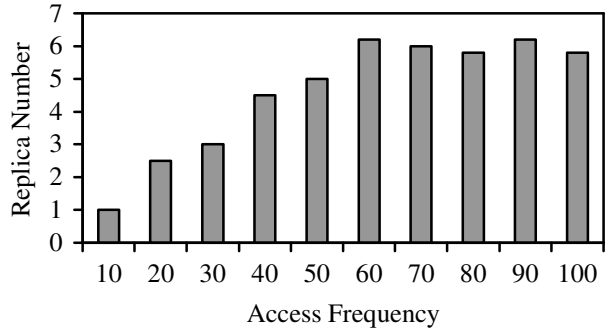

Figure 9. Replica number varies with access frequency

From Figure 8 and Figure 9, we can observe that as access frequency increases, access latency drops and replica number increases initially, but both of them become stable over time.

7. Conclusions

Replication is very useful and challenging for large-scale distributed storage system such as P2P storage system and object-based storage system. The current target is to dynamically create, migrate and delete replicas among nodes in response to changes in the access patterns and system environment.

This paper presents an adaptive replica management mechanism for OSD clusters, which regards availability and consistency maintenance overload as functions of replica number. Our objective is to build up a dynamic model to adapt to the changes of OSD clusters and satisfy both file availability and performance of storage service.

In the future work, we will introduce more object attributes to T10 OSD-SCSI commands set [10] to design OSD Qos model according to object-based storage requirements such as availability, access delay, I/O speed and workload.

References

[1] Thomas M. Ruwart, OSD: A Tutorial on Object Storage Devices, *Proceedings of the 19th IEEE/10th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2002)*.

[2] Alain Azagury, Vladimir Dreizin and Michael Factor, Towards an Object Store, *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2003).*

[3] Feng Wang, Scott A. Brandt, Ethan L. Miller, and Darrell D. E. Long, OBFS: A File System for Object-based Storage Device. *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST2004)*.

[4] Andy Hospodor, Ethan L. Miller, Interconnection Architectures for Petabyte-Scale high-performane storage system, *Proceedings of the 21st IEEE /12th NASA Goddard Conference on Mass Storage Systems and Technologies(MSST2004)* .

[5] Thanasis Loukopoulos, Ishfaq Ahmad and Dimitris Papadias, An Overview of Data Replication on the Internet, *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN.02)*.

[6] K. Ranganathan and I. Foster, Identifying Dynamic Replication Strategies for a High Performance Data Grid, presented at *International Workshop on Grid Computing, Denver, CO, 2001*.

[7] Napster project home page. *http://ww.napster.com.*

[8] FreeNet home page. *http://freenet.sourceforge.net*.

[9] K.Ranganathan, A.Iamnitchi and I.Foster, Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities, *Proceedings of the Workshop on Global and P2P Computing on Large Scale Distributed Systems, Berlin, May 2002*.

[10] Information technology–SCSI Object-Based Storage Device Commands (OSD), Oct. 2004, *http://www.t10.org/ftp/t10/drafts/osd2/osd2r00.pdf.*