Studying of Multi-dimensional Based Replica Management in Object Storage System*

Zhipeng Tan, Dan Feng, Fei He, Ke Zhou
Key Laboratory of Data Storage System, Ministry of Education
School of Computer, Huazhong University of Science and Technology,
Wuhan 430074, Hubei, China
Corresponding authors: zhipengtan@163.com

Abstract: The Object-based Storage System (OBS) has been proposed as a novel information storage technology in adaptation to the explosive growth in information quantity under the next-generation internet. The OBS treats all storage devices (OSD) and data information as objects, hence, the issues like reducing access delay over WAN, saving limited bandwidth and enhancement of data validity become problems related to the overall performance of OBS. To address these problems, we have the replica-based OBS, which brings forward a new issue on how to manage these replicas. In this paper, we present a multi-dimensional replica management scheme, and study the object searching performance within this mode. We deliver the optimal tree & improved one-path tree on the basis of similarity searching, with detailed replica indexing algorithm and emulation tests. The result of these experiments justifies their better performance in contrast to normal similarity-based indexing algorithm, with lower system cost.

Key Words: Multi-dimensional Architecture; Optimal tree; One-path tree; Replica management; Searching

1 Introduction

In the last few years, object storage system is a research hotspot and it is the key technology for next generation network storage. In the object storage system, object is the base unit of management. Object storage system provides geographically distributed storage resources for large-scale data-intensive applications that generate large data objects. However, ensuring efficient and fast access to such huge and widely distributed data objects is hindered by the high latencies of the Internet. To address these problems we introduce a set of object replication management services

*It was supported by the National Basic Research Program of China (973 Program) under Grant No. 2004CB318201.

Tan Zhipeng is a Ph.D candidate in Department of Computer Science, Huazhong University of Science & Technology. He is interested in computer architecture、information storage and database technology etc..

Feng dan is a professor of Huazhong University of Science & Technology. She is interested in computer architecture、mass information storage and disk array etc..

and protocols that offer high data availability, low bandwidth consumption, increased fault tolerance, and improved scalability of the overall system. Replication decisions are made based on a cost model that evaluates data access costs and performance gains of creating each replica. Replication technology is applied in the filed of grid、data grid、and distributed system. Although there are some research results about replication, there is no research of replication about object storage system. Our study investigates the usefulness of creating replicas to distributed object among the various nodes in the object storage system. The main aims of using replication are to reduce access latency and bandwidth consumption. Replication can also help in load balancing and can improve availability by creating multiple copies of the same object. Since the numerous replicas are redundantly stored, the number of which soars when the amount of object storage nodes and data objects increase, how to manage these replicas becomes an important problem for object storage system. In the object storage system, one of the simplest rules for managing replicated object is where read operations on an object are allowed to read any replica, and write operations are required to write all copies of the object. The rule is termed as read-one & write-all. In this paper, we provide a multi-dimensional-based replica management to control replicas, and the experiments prove its effectiveness to advance performance of object storage system with replication.

The rest of the paper is organized as follows: First, we will give related work in the Section 2; then delivers a Multi-dimensional replica management model of object storage system in section 3; section 4 discusses the implementation of multi-dimensional replica management model; in section 5, the performance of the Multi-dimensional replica management model is analyzed in terms of object availability. At last, we will conclude in section 6.

2 Related Work

Replication has been studied extensively and various distributed replica management strategies have been proposed in the literatures[1, 2, 3]. In the context of object storage technology, replication is mostly used to reduce access latency and bandwidth consumption. But replication will bring large numbers of replicas on the object storage system. Consequently, it is an important matter that how these replicas are managed. There are some researches about the size of the object replication in the grid[8], the placement of object replicas and the selection of consistency algorithms. The replica management system decides when to create and where to place a replica. These decisions are made based on a cost model that evaluates the maintenance cost and access performance gains from creating each replica. The estimates of costs and gains are influenced by many factors, such as run-time accumulated read/write statistics, the chosen consistency algorithm, run-time measured network latency, response time, bandwidth, and replica's size[8]. These parameters are changing during the program execution, so they need to be measured at runtime and fed to an optimization procedure that minimizes object access costs by dynamically changing the replicas number and placement.

To ensure scalability, we use both hierarchical and flat propagation graphs spanning the overall set of replicas to overlay replicas on the object storage system and minimize inter-replica communication costs. For the hierarchical topology, we introduce a modified fat-tree structure with redundant interconnections connecting its nodes; closer the node is to the root, more interconnections it has. The fat-tree was originally introduced by Leiserson[12] to improve the performance of interconnection networks in parallel computing systems. The hierarchical distribution is well suited for multi-tier applications, while the ring topology suits best for the multiple server or peer replica applications. For our flat topology we use the ring one. In the peer-to-peer model, any replica can synchronize with any other replica, and any update can be applied at any accessible replica. The peer model has been implemented in many systems such as Ficus[1], Rumor[2] , Roam[6] ,Bayou[7],and Locus[12]. In the hierarchical model, the replicas are placed at different levels, and communicate with each other in a client-server like scheme. This model has been realized in replication systems such as Coda[10]. To further exploit the properties of both topologies, we use a hybrid topology in which both the ring and fat-tree replica organizations can be combined into multi level hierarchies. This approach improves both the data availability and the reliability of the ring topology and allows for a scalable expansion of the hierarchical distribution. Both the ring and fat-tree connection graphs represent virtual connections between the grid nodes that hold replicas of the same object. Depending on the topology, each node is aware of its neighbors or direct ancestors and children.

As stated above there is no detailed study of management for object replications, which provides an interesting field for exploitation, and in this paper we study the management of object replications specially and analyze performance of object storage with replications. Multi-dimensional-based Replica Management Mechanism should be propitious to find the replication object. First, the replications are placed according to the multi-dimensional replica management model, but when the searching of replication, the structure of multi-dimensional can be transformed structure of tree. The transform is easy. The experiment proved this way is effective for replication management.

3 Multi-dimensional Replica Management Model

In object storage system, there are attributes and operations of object defined, which are judged as two dimensions, while at the same time, we can add more characteristic dimensions to every object, so a Multi-dimensional data structure of replicas can be given according to these attributes. Within three Multi-dimensional data structure, replicas are logically organized into a box with four planes. Figure 1 is an example of the box that consists of four planes with the black circle represents a copy at location A, B, C, … , and X. The Multi-dimensional data structure restricts the number of replicas in each plane, i.e., if l denotes the length (column) of the plane, and w is the width (row) of the plane, then each plane consists l×w replicas. Of course we can get new box from other dimensional data structure, the box must not have four planes only.

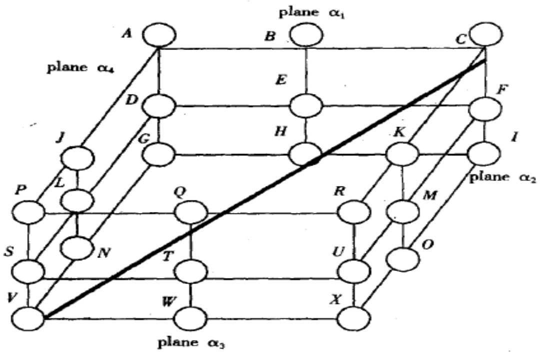

Fig.1: A box of replicas by three Multi-dimensional data structure

Definition1: A pair of replicas that can be constructed from a hypotenuse edge in a box-shape structure are called hypotenuse replicas. For the Multi-dimensional replication management structure, reading operations on a replication object are executed by acquiring a read quorum that consists of any hypotenuse replicas. In the Figure1, replicas {V,C},{I,P},{X,A}, or {G,R} are hypotenuse replicas from which it is sufficient to execute a read operation. Since each pair of them is hypotenuse replicas, read operation can be executed if one of them is accessible. If W is a set of write quorums which consists of groups that are sufficient to execute write operations under a set of hypotenuse replicas, say {V,C}, then from Figure 1, we have

W={{V,C,I,A,G,B},{V,C,I,A,G,E},{V,C,I,A,G,H},{V,C,I,R,X,K},{V,C,I,R,X,M},{V,C,I,R,X,O},{C,V,P,A,G,J},{C,V,P,A,G,I},{C,V,P,A,G,N},{C,V,P,R,X,Q},{C,V,P,R,X,T},{C,V,P,R,X,W}}.

Of course, there is weakness of the Multi-dimensional replication management structure——that is, if all store nodes in a column of each plane are unavailable, the write quorum can not be constructed. For example, when {{B,E,H},{K,M,O},{Q,T,W}, and {J,L,N}} are unavailable, the write operation is suspended even if all the other store nodes are available or accessible. So in this paper, we mainly discuss how to search the replicas needed in object storage system that has plentiful amount of object replicas by Multi-dimensional replication management structure.

4 Searching of Multi-dimensional-based Replica Management

Multi-dimensional index technology has been recognized as one of the key solutions to the acceleration of data searching. And there are already numerous hyper-dimensional index methods, for instance, the R-tree[14], R*-tree[15] as well as some variations of the R-tree[14~17], which are all based on space locality, and have been widely used in GIS(Geology Information System). However, these sorts of space locality based index methods has their innate confinement, which proves only to be effective when the following two conditions are met simultaneously: (1) the indexing object must be able to be denoted by a Eigen-value of hyper-vector space; (2) the similarity between objects must be measured by the Euclid Distance. As for the need of quick replica searching in the Object Storage System(OBS) grounded on replicas, we first institute a measurement of space through multi-dimensional replica management, and then elicit the distance-based index theory. Unlike the locality-based index technology, the distance-based one mainly deals with the comparative distance between replicas, without concerning the relative locality between them. As some typical models in this kind, one need to look no further than the M-tree[17], MVP-tree[18] and MB+tree[19], among which the VP-tree and MVP-tree are two canonical space measurement-based static index architectures, while the M-tree becomes the leading one to realize the dynamic. Additionally, the M-tree model has been improved by MB+tree and Slim-trees, for the MB+tree substitutes the hyper-dimensional index structure by two mono-dimensional index(B+tree & Block-tree) to avoid the overlapping partition of the data space, while the Slim-trees adopt a disposal procedure after creating the tree to minimize the total number of nodes and the cover radius of the data node. Nevertheless, the searching performances of these distance-based index technologies are primarily depended on the specific data distribution, and most of which includes some empirical parameters to design the models(the vantage points in VP-tree, for example), far from the “Optimal Searching Performance”. For this reason, with the idea of multi-dimensional replica storage structure under the object storage space, we introduce a brand new distance-based theory of index structure, the optimal tree and the improved one-path tree, by studying their establishment and detailed accomplishment of searching, with corresponding algorithms.

4.1 Similarity searching

We first define the OBS's storage space as a binary group: M=(D，d), in which D is the characteristic space of the object, and d is the length measurement under the D. They together meet the following conditions:

①symmetry：$d(x,y) = d(y,x)$；

②non-negative：when $x \neq y$，$0 < d(x,y) < \infty$，when $x = y$，$d(x,y)=0$；

③triangle inequality：$d(x,y) \leq d(x,z) + d(z,y)$.

From this definition, we conclude that we can only use the 3 conditions defined above when forming the index structure in the distance-based object storage space, in contrary to any other assumptions frequently used in Euclid space. Given a OBS, namely S, and the distance measurement in the object characteristic space, namely d, and Q-the accessing object replica, normally we would searching the certifiable replica through the following two ways:

Threshold-value inquiry-Query(Q，t): given a certain threshold value t, all the target replicas I in the S that fit d(Q，I)≤t.

Best-match inquiry-Query(Q，n): possible n candidate replicas that have the closest distance to the accessing replica in the S.

When concerning the similarity searching, we can use the triangle inequality to reduce the times of distance calculation during the Best-match inquiry process to enhance its efficiency. The detailed method can be described as follows:

Assume I as a replica in the S, and K={ K_1，K_2，… ，K_n } is a set of similar replica objects(-call K_i *the key object*). By using the triangle inequality, we have:

$$d(I,Q) \geq \max_{1 \leq s \leq m} \left| d(I,K_s) - d(Q,K_s) \right| \qquad (1)$$

From (1) we know that, for a random s(1≤s≤n), we have d(I，Q)≥|d(I，Ks)−d(Q，Ks)|, thus derives a lower bound of the distance between I and Q. Consider a storage system S={I_1，I_2，… ，I_n} and a very small set of similar replica objects K={K_1，K_2，… ，K_m}. If for any s and t, the distance between Is and Kt, d(I_s，K_t), has been calculated in advance, then for the similarity searching Query(Q，t), it is only needed to calculate the set {d(Q，K_1)，d(Q，K_2)，… ，d(Q，K_m)}, and it is easy to get the corresponding lower bound of distance by referring to *inequality (1)*. Apparently, if we can prove d(I_s，Q)>t, then we can eliminate Is from the candidate matching set of Q. After this kind of filtration, it is only necessary to compute every remaining object by the linear searching method, and put those that can meet the demand into the searching result set. In this triangle-inequality based similarity searching strategy, we can simply exclude those impossible replica candidates with too long distance from the inquiring replica with the assistance of "distance lower bound", thus reducing the times of distance calculation in the query. With this searching algorithm, it only takes m+u time of distance calculation(u is the number of leaving objects after filtration) and O(mn) times of simple computation. Obviously, this strategy can save a great deal of distance calculation to promote the efficiency of similarity searching remarkably, as long as the prerequisite m+u≤n can be met.

4.2 The partition、Tree structure and Searching

We begin to discuss the partition of object replica set with the multi-dimensional object replica management structure. As for convenient explanation, we treat the terms “replica”, “the characteristic vector of replica” as well as “object” and “the characteristic vector of object” as the same thing, as far as it would not lead to confusion.

4.2.1 The optimal partition of multi-dimensional replica object set

Definition 2(Optimal partition): assume D as a established multi-dimensional replica set, and d as a distance measurement under D. Select two sample point C_1 and C_2 on a random side of the multi-dimensional object replica management structure. Partition set D by these two points into two child set D_1 and D_2, so that for a random point X aside from C_1 & C_2, if $d(X,C_1)\le d(X,C_2)$, then put X into D_1;Or else put it into D_2.

We call it a “Balance Partition” if it fits the condition (1) defined as follow; For any given positive integer h>0, if the partition meets conditions (1)～(3), then we call it “Optimal Partition”, and the relevant point C_1 and C_2 as “Reference Point” to this partition.

(1) the minimum of $abs(|D_1|-|D_2|)$, or the least comparative number of the data nodes in D_1 and D_2, in which |•| is the operator in the calculation of set base number.

(2) assume$D'_1=\{X|d(X,C_1)-d(X,C_2)\le 2h\wedge X\in D_2\}$,$D'_2=\{X|d(X,C_2)-d(X,C_1)\le 2h\wedge X\in D_1\}$, then demand the least number of data nodes in the set $D'_1\cup D'_2$ by this partition.

(3) $d(C_1,C_2)>h$。

By implementing general optimal methods, such as imitative annealing idea or heredity algorithm, we can simply apply the “Balance Partition” or “Optimal Partition” to the replica object set. For a given replica object set, we can employ balance partition or optimal partition to divide it recursively, and thus establish a corresponding index structured optimal tree over the multi-dimensional replica object set. The basic thinking of optimal tree index structure is to adopt a balance or optimal method to divide the multi-dimensional replica space set I into two child sets, and recursively divide each child one with the same method, until each child set include and only include the needed accessing replica. So, the optimal tree is of a binary tree structure, representing a recursive process of partitioning the replica object space.

4.2.2 The algorithm of the index structured optimal tree establishment

Assume $I=(O_1,O_2, \ldots, O_n)$ is data set including n replica objects, and d is a distance measurement. Then the establishing algorithm of optimal tree can be described as follows:

```
Input: dataset $I$
Output: optimal tree $V$
    (1) if $|I|=0$, then establish a void tree, return.
    (2) else,
      (2.1) use a balance or optimal partition method to partition dataset $I$ into
            two child sets: $D_l$ & $D_r$(the Reference Points are C1 and C2,
            accordingly), and
            $D_l=\{Oi \mid d(C_1,Oi)\le d(C_2,Oi)\wedge Oi\in I\}$,
            $Dr=\{O_j \mid d(C_2,Oj)<d(C_1,Oj)\wedge Oj\in I\}$;
      (2.2) branch root V with $D_l$ and $D_r$ as the left & right child tree;
```

(2.3) if D_l or D_r is leaf node, then calculate $d(C_i,O_j)$, put it into leaf-node distance array $D_{i[j]}$, return.

(3) treat D_l and D_r recursively by using the algorithm above, forming relevant optimal child tree.

Theorem 1: Assume D as a replica object set, d is a distance measurement under D, and D_1 & D_2 are two child sets derived from balance or optimal partition, both of which are dimension-decreased replica object sets, and C_1 and C_2 are sample points to D_1 & D_2. Consider similarity inquiry Query(q,t)(q is the demanded inquiring replica, t is the threshold value). We have, if $d(q,C_1)\le d(q,C_2)$, then if there exists a point $x\in D_2$ to let $d(x,C_1)-d(x,C_2)\le 2t$ stand, we must search D_1 and D_2 to execute similarity inquiry Query(q,t); if not exists, the Query(q,t) only needs to search D_1. Similarly, if $d(q,C_2)\le d(q,C_1)$, then if there exists a point $x\in D_1$ to let $d(x,C_2)-d(x,C_1)\le 2t$ stand, we must search D_1 and D_2 to execute similarity inquiry Query(q,t); if not exists, the Query(q,t) only needs to search D_2.

Prove: In the first circumstance $d(q,C_1)\le d(q,C_2)$, because d is a distance measurement, and by the definition of it we can conclude the following two inequalities:

$d(q,C_1)+d(C_1,x)\ge d(q,x)$,

$d(q,C_1)+d(q,x)\ge d(C_1,x)$, *derivable from the two inequalities.*

$d(q,C_1)\ge|d(q,x)-d(C_1,x)|$, (2)

and $d(q,C_2)\le|d(q,x)+d(C_2,x)|$。 (3)

From (2),(3) we have, $d^2(q,C_1)\ge[d(q,x)-d(C_1,x)]^2$, and $d^2(q,C_2)\le[d(q,x)+d(C_2,x)]^2$**.**

According to the assumption condition $d(q,C_1)\le d(q,C_2)$, we have:

$[d(q,x)-d(C_1,x)]^2\le[d(q,x)+d(C_2,x)]^2$,

$\Rightarrow -2d(q,x)d(C_1,x)+d^2(C_1,x)\le 2d(q,x)d(C_2,x)+d^2(C_2,x)$,

$\Rightarrow 2d(q,x)[d(C_1,x)+d(C_2,x)]\ge d^2(C_1,x)-d2(C_2,x)$,

$\Rightarrow$ $d(q,x)\ge(d(C_1,x)-d(C_2,x))/2$. (4)

From (4) we know, if $d(x,C_1)-d(x,C_2)>2t$ exists, then $d(q,x)>t$. That is to say, if there is a random x in D_2 that leads to $d(x,C_1)-d(x,C_2)>2t$, then x could not be a inquiry candidate. Hence, we only need to search D_1 to execute Query(q,t). Or else, if there exists a point $x\in D_2$ leading to $d(x,C_1)-d(x,C_2)\le 2t$, then we are not sure whether d(q,x) is smaller than the threshold value. In this case, D_1 and D_2 need to be scanned at the same time to execute Query(q,t). Similarly, the conclusion stands in the second circumstance.

Inference 1: Assume the same condition as Theorem 1, then:

(1) When $d(q,C_1)\le d(q,C_2)$, if $d(q,C_2)-d(q,C_1)\le 2t$ stands, then we must scan D_1 and D_2 to execute Query(q,t); Or else, we only need to scan D_1;

(2) When $d(q,C_2)\le d(q,C_1)$, if $d(q,C_1)-d(q,C_2)\le 2t$ stands, then we must scan D_1 and D_2 to execute Query(q,t); Or else, we only need to scan D_2;

Prove: In the 1st circumstance $d(q,C_1)\le d(q,C_2)$, let $d'=d(q,C_2)-d(q,C_1)$, then to any $x\in D_2$, suppose $d(x,C_1)-d(x,C_2)=d>0$(see Definition 1). By the triangle inequality, we can have:

$d(q,x)>d(x,C_1)-d(q,C_1)$, $d(q,x)>d(q,C_2)-d(x,C_2)$, thus derives,

$2d(q,x)>[d(x,C_1)-d(x,C_2)]+[d(q,C_2)-d(q,C_1)]=d+d'$,

To be more concise: $d(q,x)>(d+d')/2$。

- If $d'=d(q,C_2)-d(q,C_1)>2t$, then $d(q,x)>(d+d')/2\ge t+d/2>t$. From this can we conclude, the $x\in D_2$ has been excluded from the candidate matching list of q. According to x's random nature, Query(q,t) only needs to scan D_1.
- If $d'=d(q,C_2)-d(q,C_1)\le 2t$, we can not guarantee $d(q,x)>t$, which means that we must scan D_1 and D_2 to execute Query(q,t). Similarly, the conclusion stands in the 2nd circumstance.

4.2.3 The searching over index structured optimal tree

With Inference 1, we can deliver the searching algorithm of optimal tree as follows:
Optimal tree searching algorithm
Input: optimal tree *V*
Output: searching outcome set *result*

```
(1) Select current node;
(2) If current node is a leaf node, then to each data node Pi in the leaf node,
    (2.1) Select distance array $D_1$ and $D_2$;
    (2.2) If $\max\{|d(q,C_1)-D_{1[i]}|,|d(q,C_2)-D_{2[i]}|\}>t$, then Pi is a non-matching
          candidate; Or else, compute the d(q,Pi), if $d(q,Pi)\le t$, then add it into result.
    (2.3) return;
(3) if current node is a internal node, then
    (3.1) compute the distance $d(q,C_1)$ and $d(q,C_2)$;
    (3.2) if $d(q,C_1)\le d(q,C_2)$, meanwhile $d(q,C_2)-d(q,C_1)\le 2t$, then recursively scan
its left & right child trees; or else, recursively scan the left child tree.
    (3.3) if $d(q,C_2)\le d(q,C_1)$, meanwhile $d(q,C_1)-d(q,C_2)\le 2t$, then recursively scan
its left & right child trees; or else, recursively scan the right child tree.
```

It is easy to discover from the searching algorithm of optimal tree that, the optimal tree, a distance based multi-dimensional space index structure, can really advance the speed of similarity searching, while the query efficiency of which would not decrease noticeably as the dimension grows. However, because the optimal tree is also of binary searching structure, the level of the tree can be considerable to high-capacity dataset, let alone the cost of recursive searching the child ones which may influence the total performance. To better implement the optimal tree strategy, a feasible way is to deduct the height of the tree to fulfill the aim of reducing the computation of distance. Another possible way that may contribute to the improvement is to ensure the one-way down searching during the query process, without the cost of scanning branches back and forth. Theoretically, this could double the overall performance of searching. With this idea in mind, we deliver a modified optimal tree index structure: the one-path tree.

4.2.4 Improved indexing one-path tree

Definition 3(Redundant Storage): Assume D, D_1, D_2, C_1 and C_2 has use the same definition as Definition 1. Suppose $D'_1=\{x|d(x,C_1)-d(x,C_2)\le 2h\wedge x\in D_2\}$, $D'_2=\{x|d(x,C_2)-d(x,C_1)\le 2h\wedge x\in D_1\}$, put the data nodes in D_1' and D_2' into D_1 and D_2 overlappingly, thus we have two extended subset $\overline{D1}$ and $\overline{D2}$, or $\overline{D1} = D_1\cup D'_1$, $\overline{D2} = D_2\cup D'_2$. We call the partition above "Redundant Storage". In a storage system like replica-based OBS, which contains a large number of replicas, it naturally results in redundant storage. And in the multi-dimensional replica management mechanism, replica objects on sides formed by different dimensions have dissimilar redundancy, while replicas on various sides tend to have less redundancy or they are totally different object's replicas.

It is easy to discover from Def 2 that, we would get two overlapping extended subsets after a redundant segmentation, with their redundancy determined by parameter h and the particular replica distribution. In the terms of redundant storage, we can use the inequality $d(C_1,x)-d(C_2,x)>2h$ or $d(C_2,x)-d(C_1,x)>2h$ to remove those replicas which are too distant to become a matching candidate. Although the method described here are probabilistic, we can still assure its correctness on similarity searching. Thus can we deduce theorem 2.

Theorem 2: Assume D, D_1, D_2, $\overline{D1}$, $\overline{D2}$, C_1 and C_2 has use the same definition as Definition 3. And to any given user Query(q,t), we have $h\ge t$, then it leads to the following: if $d(q,C_1) \le d(q,C_2)$, then Query(q,t) only need to search $\overline{D1}$; or else, Query(q,t) only need to search $\overline{D2}$.

Prove: Suppose d(q,C1), d(q,C2). From Def 2 we know, in order to prove that "Query(q, t) only need to search $\overline{D1}$", the only necessity is to prove that to a any given $x \in D_2$, if $x \notin \overline{D1}$, then x can not be a matching candidate of q. On the other hand, from Def 3 we acquire that, if $x \notin \overline{D1}$, $x \in D_2$, then $d(x,C_1)-d(x,C_2)>2h$. By referring to the proving method of "Inference 1", we have
$d(q,x)>(d+d')/2$(d, d′ takes the same meaning as in Infer 1).

Thus derives: $d(q,x)>h+d/2>h \ge t$, thereby, $d(q,x)>t$. So x is not a matching candidate of q, which in another word indicates that it only takes Query(q,t) to scrutinize $\overline{D1}$.

Similarly can we prove, if $d(q,C_2)<d(q,C_1)$, then Query(q,t) only needs to search $\overline{D2}$.

Definition 4(one-path tree): Assume I as a Image characteristic vector set, select parameter § , and utilize some certain optimal partition method in separate the OBS multi-dimensional replica set into two subsets D_L and D_R. Then use redundant storage strategy to extend the two sub ones so that $D_L = D_L \cup D_L'$, $D_R = D_R \cup D_R'$. Use the same recursive partition and disposition regarding every single extended subclass (D_L & D_R), until each of the two includes and only includes the designated number of data nodes or, the subsets are "small enough". If the replica object subset $D_L(D_R)$ satisfies $D_L' = D_R$ ($D_R' = D_L$), then $D_L(D_R)$ can be considered small enough. We call this binary tree architecture through replica object space partitioning the one-path tree.

There is immanent similarity between the one-path tree and the optimal one. And according to Theorem 2, the similarity searching based on one-path tree index structure can be relatively simple, which only takes a single path searching along the one-path tree. Consequently the efficiency of tree searching is boosted prominently.

4.2.5 Searching on one-path tree

```
Input: one-path tree
Output: searching outcome set result
    (1) If current node is a leaf node, then for every data point Pi in the leaf node,
      (1.1) Select the distance array and store $D_1$ & $D_2$;
      (1.2) If $\max\{|d(q,C_1)-D_{1[i]}|, |d(q,C_2)-D_{2[i]}|\}>t$, then $P_i$ is a non-matching candidate(drop it without compute the distance); or else, compute $d(q,P_i)$. If $d(q,P_i) \le t$, then put it into the result.
      (1.3) return
    (2) if current node is an internal node, then
      (2.1) if $d(q,C_1) \le d(q,C_2)$, search the left child tree recursively;
      (2.2) if $d(q,C_2)<d(q,C_1)$, search the right child tree recursively;
```

5 Experiment and Evaluation

In the experiment, we mainly focus on the analysis and comparison of replica-based Object Storage System. We use the two indexing structures: the optimal tree and improved one-path tree, to search for replica object, while recording the searching performance, system cost, access delay queue and I/O flow rate corresponding to the variation of the total amount of replicas and the dimensions of replica management.

The three indexing structure, similarity searching, the optimal tree and improved one-path tree, are all realized by the standard C language under the Linux platform. In

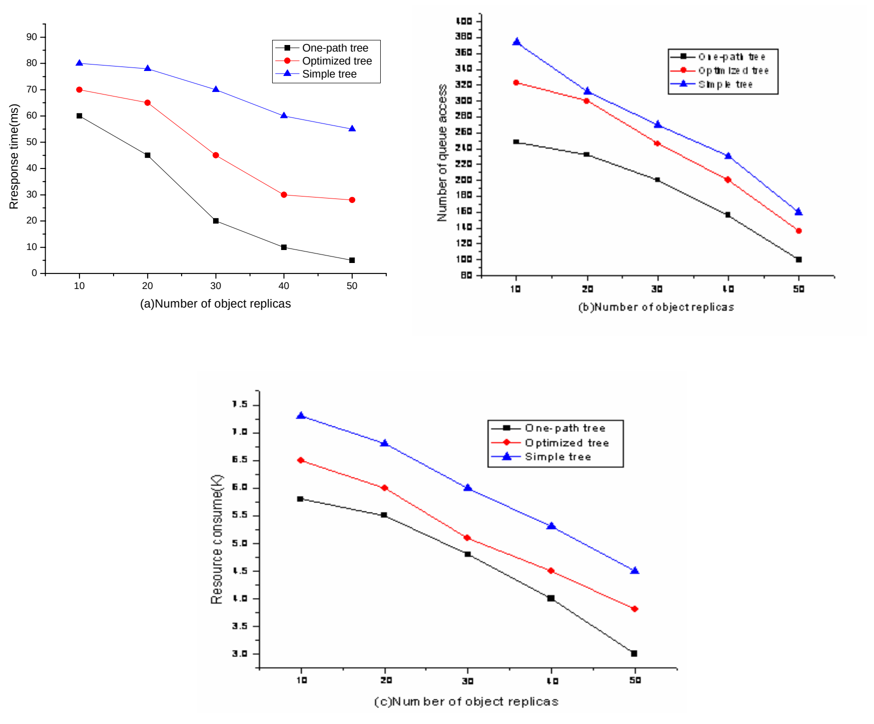

Fig.2. Influence of Performance by Replicas

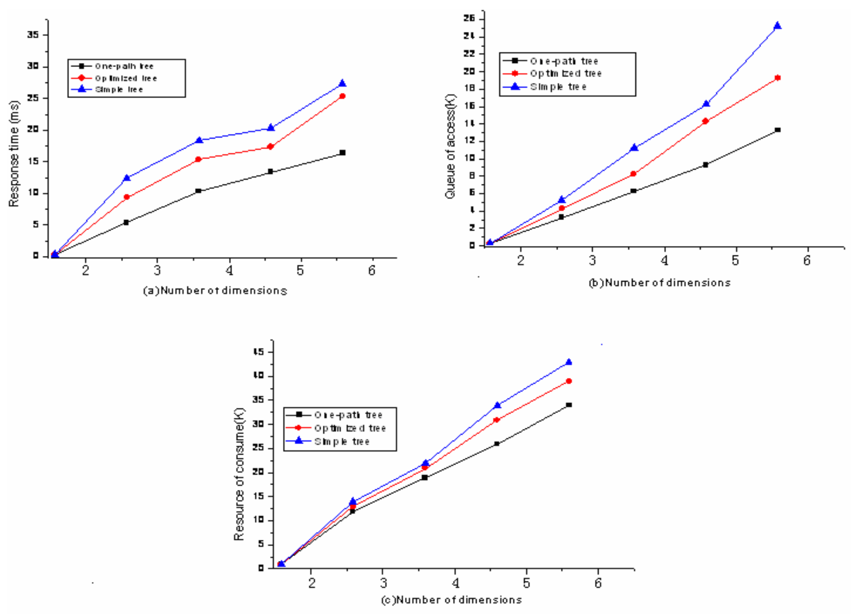

Fig.3. Influence of Performance by Dimensions

the experiment, we study the OBS with 100 storage nodes, and 1000 objects inside. To any object under study, the tested numbers of replicas are 10, 20, 30, 40, 50, which are stored in different nodes. We observe the variation in the searching response time and expense of searching under the three architectures, as well as the access queue generated, with an increasing replicas tendency. Additionally, we observe the three aspects listed above under different replica management dimensions as 2D, 3D, 4D, and 5D. We assume that objects searched are extant object-replicas. The results manifest that the one-path tree has the best searching performance, especially to those queries within a large range.

Figure 2 shows the pattern in the 3 searching architectures: when the object-replica increases, the system response time and system cost drop down, with even shorter waiting queue. It is obvious that the improved one-path tree takes the lead in comparison to the other two.

Figure 3 reveals that when the management dimension increases, the system's reaction time and its consumption, along with the access queue are all increasing consequently.However, the positive effects the multi-dimensional mechanism plays on replica management are evident, at which we would later be discussed in other special papers. In conclusion, the experiment proves that the performance of improved one-path tree is the best structure in comparison to normal similarity searching and optimal tree.

6 Conclusion

In the object storage system, managing such huge amounts of objects in a centralized manner is almost impossible due to extensively increased data access time. So object replication is a key technique to manage large object in a distributed manner. By its nature, we can achieve better performance (access time) by replicating object in geographically distributed object stores. In object storage system, user's job may probably require the access to large number of objects, and if the required objects are replicated in the node in which the job is executed, the job is able to process object without any communication delay. However, if required objects are not in the site, they should be fetched from the other nodes. Object fetch takes very long time because the size of a single replica may reach giga-byte scale in some applications while the network bandwidth between nodes is limited. As a result, job execution time becomes lengthy due to delay of fetching replicas over Internet. So searching object is the key factor in replica-based object storage system. In this paper, we advance a model of multi-dimensional based replica management model, and study the searching of object within this model. On the ground of similarity search, we advance optimal tree and improved optimal tree——one-path tree. And the paper gives two kinds of index algorithm under two tree structures, and then tests the algorithm with imitation. At the same time, there are problems emerged on the replicas in the object storage system, such as the placement of replicas、consistency of replicas、granularity of replicas and so on, which would be discussed in other papers.

Reference

1. R. Guy, J. Heidmenn, W. Mak, T. Page Jr., G. Popek, and D. Rothmeier, "Implementation of the Ficus Replicated File system," Proceedings of the summer Usenix Conference, 1990.

2． R. Guy, P. Reiher, D. Ratner, M. Gunter, W. Ma, and G. Popek, ”Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication,” Workshop on Mobile Data Access, November 1998.
3． T. Page, R. Guy, J. Heidemann, D. Ratner, P. Reiher, A. Goel, G. Kuenning, and G. Popek, ”Perspectives on Optimistically Replicated, Peer-To-Peer Filing,” Software - Practice and Experience, Dec. 1997.
4． K. Ranganathan and I. Foster, ”Identifying Dynamic Replication Strategies For a High performance Data Grid,” Proceedings of the International Grid Computing Workshop, Denver, November 2001.
5． K. Ranganathan and I. Foster, ”Design and Evaluation of Replication Strategies for a High Performance Data Grid,” International Conference on Computing in High Energy and Nuclear Physics, Beijing, September 2001.
6． D. H. Ratner, ”Roam: A Scalable Replication System for Mobile and Disconnected Computing,” PhDThesis, University of California, Los Angeles, Los Angeles CA, January 1998.
7． D. Terry, M. Theimer, K. Peterson, A. Demers, M. Spreitzer, and C. Hausen, ”Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System,” Proceedings of the fifteenth Symposium on Operating systems Principles, pp 49-70 ACM, October 1983.
8．S. Vazhkudai, S. Tuecke, I. Foster, ”Replica Selection in the Globus Data Grid,” Proceedings of the First IEEE/ACMInternational Conference on Cluster Computing and the Grid (CCGRID 2001), pp. 106-113, IEEE Computer Society Press, May 2001.
9． M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, ”Understanding Replication in Databases and Distributed Systems,” Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS’2000).
10． M. Satyanarayanan, J. Kister, P. Kumar, M. Okasaki, E. Siegel, and D. Steere, ”Coda: A Highly Available File System for a Distributed Workstation Environment,” IEEE Transactions on Computers, 39(4):447- 459, April 1990.
11． G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel, ”Locus: A Network Transparent High Reliability Distributed System,” Proceedings of the Eighth Symposium on Operating Systems Principles, pp 169-177 ACM, December 1981.
12．C. H. Leiserson, ”Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing,” IEEE Transactions on Computers, vol. C-34, no. 10, pp. 892-901, October 1985.
13． Guttman A. R-Trees: A dynamic index structure for spatial searching. In: Yormark B, ed. Proc. of the ACM SIGMOD Conf. Boston,1984. 47~57.
14． Berkmann N, Krigel HP. Schneider R, Seeger B. The R*-tree: An efficient and robust access method for points and rectangles. In:Hector GM, Jagadish HV, eds. Proc. of the ACM SIGMOD Conf. Atlantic, 1990. 322~331.
15． Katayama N, Satoh S. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In: Peckham J, ed. Proc. Of the ACM SIGMOD Conf. Tucson, 1997. 369~380.
16． White DA, Jain R. Similarity indexing with the SS-tree. In: Stanley YWS, ed. Proc. of the 12th Int’l Conf. on Data Engineering. New Orleans: IEEE Computer Society, 1996. 516~523.
17． Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd VLDB Conf. Athens: Morgan Kaufmann Publishers, 1997. 426~435.
18．Bozkaya T, Ozsoyoglu M. Distance-Based indexing for high-dimensional metric spaces. In: Peckham J, ed. Proc. of the ACM SIGMOD Conf. on Management of Data. Tucson, 1997. 357~368.
19．Ishikawa M, Chen H, Furuse K, Yu JX, Ohbo N. MB+tree: A dynamically updatable metric index for similarity search. In: Lu HJ, Zhou AY, eds. Proc. of the 1st Int’l Conf. on Web Age Information Management. Lecture Notes in Computer Science 1846, Springer-Verlag, 2000. 356~373.