PROTECTED DATA OBJECTS REPLICATION IN DATA GRID

G. Aruna Kranthi and D. Shashi Rekha

Sr. Asst. Professor, Department of Computer Engineering, S.R. Engineering College, Warangal, A.P.

raj.kranthi@gmail.com

M.Tech (C.S.E), S.R. Engineering College, Warangal, A.P.

sashi.mtech@gmail.com

ABSTRACT

Secret distribution and erasure convention - based approaches have been used in distributed storage systems to provide the confidentiality, integrity, and availability of critical information. To achieve performance goals in data accesses, these data fragmentation approaches can be combined with dynamic replication. In this paper, we consider data partitioning (both secret distribution and erasure convention) and dynamic replication in data grids, in which security and data access performance are critical issues. More specifically, we investigate the problem of optimal allocation of sensitive data objects that are divided by using secret distribution scheme or erasure convention scheme and/or replicated. The grid topology we consider consists of two layers. In the upper layer, multiple clusters form a network topology that can be represented by a general graph. The topology within each cluster is represented by a tree graph. We decompose the share replica allocation problem into two sub problems: the Optimal Inter cluster Resident Set Problem (OIRSP) that determines which clusters share replicas and the Optimal Intra need cluster Share Allocation Problem (OISAP) that determines the number of share replicas needed in a cluster and their placements. We develop two heuristic algorithms for the two sub problems.

KEY WORDS

Protected data, secret distribution, erasure convention, replication, data grids.

1. INTRODUCTION

Data grid is a distributed computing architecture that integrates a large number of data and computing resources into a single virtual data management system [2]. It enables the sharing and coordinated use of data from various resources and provides various services to fit the needs of high-performance distributed and data-intensive computing. Many data grid applications are being developed or proposed. These data grid applications are designed to support global collaborations that may involve large amount of information, intensive computation, real time, or nonreal time communication. Success of these projects can help to achieve significant advances in business, medical treatment, disaster relief, research, and military and can result in dramatic benefits to the society.

There are several important requirements for data grids, including information survivability, security, and access performance [1], [2]. Replication is frequently used to achieve access efficiency, availability, and information survivability. The underlying infrastructure for data grids can generally be classified into two types: cluster based and peer-to-peer systems [3]. In pure peer-to-peer storage systems, there is no dedicated node for grid applications (in some systems, some servers are dedicated). Replication can bring data objects to the peers that are close to the accessing clients and, hence, improve access efficiency. Having multiple replicas directly implies higher information survivability. In cluster-based systems, dedicated servers are clustered together to offer storage and services. However, the number of clusters is generally limited and, thus, they may be far from most clients.

In this paper, we consider combining data partitioning and replication to support secure, survivable, and high performance storage systems. Our goal is to develop placement algorithms to allocate share replicas such that the communication cost and access latency are minimized. The remainder of this paper is organized as follows: Section 2 describes a data grid system model and the problem definitions. Section 3 introduces a heuristic algorithm for determining the clusters that should host shares. In Section 4, the results of the experimental studies are discussed. Section 5 discusses some research works that are related to this research. Section 6 states the conclusion of this paper.

2. System model and Problem Specification

In this paper, we consider achieving secure, survivable, and high-performance data storage in data grids. To facilitate scalability, we model the peer-to-peer data grid as a two level topology (shown in Fig. 1). One or several such autonomous systems that are geographically close to each other can be considered as a cluster. The system consists of M clusters, $H_1 \ldots, H_M$, which are linked together and form a general

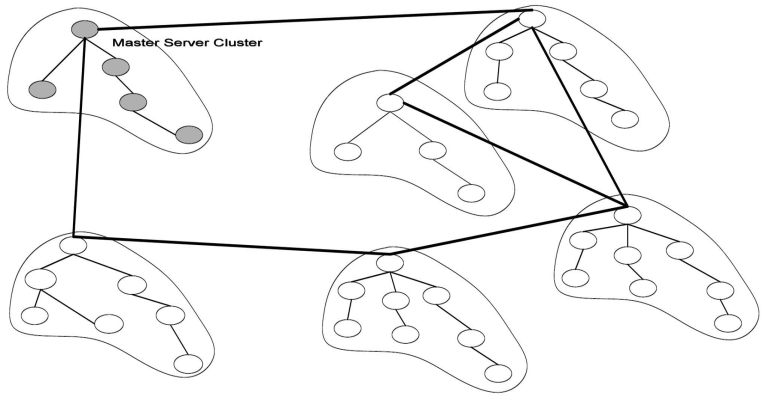

Figure 1 A sample graph of our system topology.

Graph topology $G^C = (H^C, E^C)$. Here, $H^C = \{H_1, \ldots, H_M\}$ and E^C is the set of edges connecting the clusters. Each edge represents a logical link which may be multiple hops of the physical links. It is likely that the clusters are linked to the backbone and should be modeled by a general graph.

Within each cluster, there may be many subnets from the same or multiple institutions. Among all the physical nodes in the cluster, some nodes, such as servers, proxies, and other individual nodes, may be committed to contribute its storage and/or computation resources for some data grid applications. These nodes are connected via logical links. Internet message routing is relatively stable in days or even weeks and the multiple routing paths generally form a tree. Thus, for simplicity, we model the topology inside a cluster as a tree. Consider a cluster H_x. Let $G_x = (V_x, E_x)$ represents the topology graph within the cluster, where $V_x = \{P_{x,1}, P_{x,2}, \ldots, P_{x,Nx}\}$ denotes the set of N_x (N if only considering cluster H_x) nodes in cluster H_x, and E_x is the set of edges connecting nodes in H_x. Also, let P_x^{root} denote the root node in H_x (e.g., $P_x^{root} \in V_x$). We assume that all traffic in H_x goes through the network where P_x^{root} resides. Let $\delta (P_{x,I}, P_{x,j})$ denote the shortest path between $P_{x,I}$ and $P_{x,j}$ in H_x, and $|\delta (P_{x,I}, P_{x,j})$ denote the distance of $\delta (P_{x,I}, P_{x,j})$. Also, let $\delta (H_x, H_y)$ denote the shortest path between H_x and H_y (actually, between P_x^{root} and P_y^{root}), and $|\delta (H_x, H_y)|$ denote the distance of $\delta (H_x, H_y)$. We assume that $\delta (P_{x,I}, P_{x,j})$ for any i, j, and x is much less than $|\delta (H_x, H_z)|$ for any y and z, where $y \neq z$ (i.e., the distance between any two nodes within a cluster is less than the distance between any two clusters).

The data grid (represented by the set of clusters HC) hosts a set of data objects D (D can contain the application data or keys). One of the clusters is selected as the Master Server Cluster (MSC) for some data objects in D, denoted as HMSC (different data objects may have different HMSC). HMSC hosts these data objects permanently (it may be the original data source). These data objects may be partially replicated in other clusters in HC to achieve better access performance. Due to the increasing attacks on Internet, a node hosting some data objects in D has a significant chance of being compromised. If a node is compromised, all the plaintext data objects stored on it are compromised.

2.1. Access Model and Problem Decomposition

Data placement decisions are made based on historical client access patterns. We model access patterns by analyzing the number of read/write accesses from each node or each cluster. Consider a data object d. Let T denote the time period unit for collecting information of access patterns. Let $A^r(P_{x,i})$ and $A^w(P_{x,i})$ denote the numbers of read and write accesses, respectively, initiated from node $P_{x,i}$ over time T. Also, let $A^r(H_x)$ and $A^w(H_x)$ denote the numbers of read and write accesses, respectively, initiated from a cluster H_x over time T. $A^r(H^x) = \sum_i A^r(P_{x,i})$ and $A^w(H_x) = \sum_i A^w(P_{x,i})$. Let W^C denote the total number of update requests on d in G^C, i.e., $w^C = \sum_{Hx} A^w(H_x)$.

Based on the client access patterns, subsets of the m shares of each data in D may be replicated to the clusters in H^C. The set of clusters that hold shares is defined as the cluster level resident set. Let R^C denote the **cluster level resident set**, i.e., $R^C = \{H_x \mid \text{clusters hold shares}\}$. To minimize the communication cost, we consider that a cluster holds either none or at least l distinct share replicas (will be proven in Theorem 3.1). Also, we assume that each cluster holds only distinct shares (i.e., at most m) since, otherwise, extra efforts are required to avoid reading duplicated shares (a large m value ensure that a sufficient distinct shares can be allocated to each cluster). Intracluster residence set is the set of nodes that holds a share replica within a cluster. Let R_x denote the intracluster residence set of H_x, i.e., $R_x = \{P_{x,i} \mid P_{x,i}$ holds a share and $P_{x,i}$ is in $H_x\}$. Correspondingly, $|R_x|$ denotes the number of share replicas in H_x. We say R_x (or R^C) is **connected** if and only if every node in R_x (or R^C) has at least one path to any other node in R_x (or R^C), and each node on the path also belongs to R_x (or R^C). Otherwise, R_x (or R^C) is partitioned. In this case, R_x (or R^C) contains multiple subgraph, $R_{x,1}, R_{x,2}, \ldots, R_{x,n}$ (or $R^C_1, R^C_2, \ldots, R^C_n$), n > 1, $R_{x,i}$ (or R^C_i), $1 \leq i \leq n$, is a connected subgraph, and $R_{x,i}$ and $R_{x,j}$ (or R^C_i and R^C_j), $i \neq j$, are not connected.

Now, consider the intracluster level. Let $\delta(P_{x,i}, R_x)$ denote the shortest path from $P_{x,i}$ to any node in R_x, and $\delta(P_{x,i}, P_x^{root})$ denote the shortest path from $P_{x,i}$ to the root node in H_x. Also, let $\delta(H_x, R^C)$ denote the shortest path from H_x to the closest cluster in R^C, and $|\delta(H_x, R^C)|$ is the distance of path $\delta(H_x, R^C)$ (only counting the cluster level cost). Let $\Gamma(P_{x,i}, R_x, \alpha)$ denote the MST rooted at $P_{x,i}$ and includes a total of α nodes hosting shares in cluster H_x.$|\Gamma(P_{x,i}, R_x, \alpha)|$ represents the total distance of the MST. Let $\Gamma^C(R^C)$ denote the MST from H_{MSC} to all clusters in R^C at the cluster level. $|\Gamma^C(R^C)|$ is the total distance of the MST $\Gamma^C(R^C)$, but only considering the costs at the cluster level (i.e., the distance to the root node of each involved cluster).

The read access protocol tries to read the closest *l* share replicas. Consider a client C sending a read request to $P_{x,i}$. If the local cluster of $P_{x,i}$ (i.e., H_x) holds shares (note that H_x holds either none or at least *l* share replicas), then it reads *l* shares within H_x (the *l* nodes are selected such that the communication cost is minimal). The access cost in this case is $|\Gamma(P_{x,i}, R_x, l)|$ (assume that the communication cost between the client and $P_{x,i}$ is negligible). If H_x does not hold share replicas, then $P_{x,i}$ obtains all *l* shares from the closest cluster H_y where $H_y \in R^C$. The algorithm for transferring shares from the source cluster H_y to the requesting cluster H_x is defined as follows: 1) all the nodes in H_y holding the desired shares send the shares to P_y^{root} (encrypted using the session key between H_y and the client); 2) P_y^{root} puts the pieces into one message and sends it to P_x^{root}; and 3) P_x^{root} sends the shares to the requesting node $P_{x,i}$ and $P_{x,i}$ forwards them to client *C*. Overall,

The *read cost* is $|\delta(P_{x,i}, P_x^{root})| + |\delta(H_x, R^C)| + |\Gamma(P_y^{root}, R_y, l)|$.

Note that $|\Gamma(P_{x,i}, R_x, l)| = 0$ if H_x hosts less than *l* shares,

and $|\delta(P_{x,i}, P_x^{root})| + |\delta(H_x, R^C)| + |\Gamma(P_y^{root}, R_y, l)| = 0$, otherwise.

Let readCost denote the total read cost in the system.

We have

readCost

$$= \sum_{H_x} \sum_{i} \begin{cases} |\Gamma(P_{x,i}, R_x, l)|, & \text{if } H_x \text{ holds at least 1 shares,} \\ |(P_{x,l}, P_x^{root})| + |(H_x, R^C)| \\ \quad + |\Gamma(P_y^{root}, Ry, l)|, & \text{otherwise.} \end{cases}$$

Let *updateCost* denote the overall update cost in the system. We have

$$UpdateCost = w^c * \left(|\Gamma^C(R^C)| + \sum_{Hx} |\Gamma(P_x^{root}, R_x, |R_x|)|\right) + forwardCost.$$

Note that we do not consider the extra cost required for the detection and recovery of an invalid share. With both update and read cost, the total cost becomes *Tcost* = *updateCost* + *readCost*. Table 1 gives a summary of the notation used in this paper.

We can decompose the partitioned data replica problem into two subproblems. The first problem is to decide which cluster should keep the share replicas at the cluster level, and we define it as the Optimal Intercluster Resident Set Problem (OIRSP). The second problem is to decide how many share replicas are needed for each cluster and how to allocate them within the cluster. We define it as the Optimal Intracluster Share Allocation Problem (OISAP). Each cluster is viewed as a single node in OIRSP. In the next two sections, we specify the two subproblems in details.

2.2. OIRSP Specification

We define the first problem, OIRSP, as the optimal resident set problem in a general graph (intercluster level graph) with an MSC H_{MSC}. Our goal is to determine the optimal R^C that yields minimum access cost at the cluster level. For a cluster $H_x \in R^C$ with $| R_x | \geq l$, all read request from H_x are served locally and the cost is 0 at the cluster level. For a cluster H_x with $| R_x | < l$, it always transmits all read access requests in H_x to the closest cluster $H_y \in R^C$ to access l distinct shares, with $|R_y| \geq l$. The read cost of cluster at the cluster level is $A^r(H^x) * | \delta(H_x, R^C)|$. Let $ReadCost^C(G^C, R^C)$ denote the total read cost in G^C with the resident set R^C, then

$$ReadCost^C(G^C, R^C) = \sum_{Hx} A^r(H_x) * | \delta(H_x, R^C) |.$$

Let $UpdateCost^C(G^C, R^C)$ denote the total update cost in G^C with the resident set R^C, then
$$UpdateCost^C(G^C, R^C) = w^C * | \Gamma^C(R^C)|.$$

Thus, the total access cost in G^C, denoted as $Cost(G^C, R^C)$ is defined as follows:

$$Cost^C(G^C, R^C) = UpdateCost^C(G^C, R^C) + ReadCost^C(G^C, R^C).$$

The problem here is to decide the share replica resident set R^C in G^C, such that the communication *cost* $Cost^C(G^C, R^C)$ is minimized.

TABLE 1

Summary of the Frequently Used Notation

H^C	The set of M clusters in the system
H_x, H_y, H_z	Denote individual clusters in H^C
R^C	The entire set of clusters that host shares of data d
R_x	The entire set of nodes that host shares of data d in cluster H_x, and it is changed to R if considering only a single cluster H_x later
$\delta(H_x, H_y)$	Shortest path between clusters H_x and H_y with distance $\|\delta(H_x, H_y)\|$
$\Gamma^C(R^C)$	The minimal spanning tree rooted at H_{MSC} that connects all clusters in R^C
$A^r(H_x)/A^w(H_x)$	The total number of read/write requests from a cluster H_x
V_x	The entire set of N nodes in cluster
$P_{x,i}$	A node in cluster H_x (or simply P_i if considering only a single cluster H_x later)
R_x', $R^{C\prime}$	A resident set that is potentially different from R_x or R^C (will be defined later)
$\delta(P_{x,i}, P_{x,j})$	Shortest path between two nodes $P_{x,j}$ and $P_{x,j}$ (or simply $\delta(P_i, P_j)$ later)
$\Gamma(P_{x,i}, R_x, \alpha)$, $\Gamma(P_{x,i}, R_x, l)$, and $\Gamma(P_{x,i}, R_x, \|R_x\|)$	The minimal spanning tree rooted at $P_{x,i}$ that connects α nodes, l nodes, and all the nodes in R_x
$A^r(P_{x,i})$ / $A^w(P_{x,i})$	The total number of read write requests from a node $P_{x,i}$
updateCost, $WriteCost(R)$, $UpdateCost^C(G^C, R^C)$.	The total update cost in the entire data grid, the update cost inside a single cluster only, and the update cost at the cluster level only, respectively
readCost, $ReadCost(R)$, and $ReadCost^C(G^C, R^C)$	The total read cost in the entire data grid, the read cost inside a single cluster only, and the read cost at the cluster level only, respectively
Tcost, $Cost(R)$, and $Cost^C(G^C, R^C)$	The total access cost in the entire data grid, the total access cost inside a single cluster only, and the total access cost at the cluster level only, respectively

2.3. OISAP Specification

When we consider allocation problem within a cluster H_x, we can isolate the cluster and consider the problem independently. As discussed earlier, all read requests from remote clusters can be viewed as read requests from the root node. Also, the w^C updates in the entire system can be considered as updates done at the root node of the cluster. Thus, we can simplify the notation when discussing allocation within H_x by referring to everything in the cluster without the cluster subscript. For example, we use $G = (P, E)$ to represent the topology graph of H_x, where $P = \{P_1, P_2, \ldots, P_N\}$. Similarly, P^{root} represents the root node of H_x, $\delta(P_i, P_j)$ represents the shortest path between two nodes inside H_x, and R represents the resident set of H_x.

Let *ReadCost* (R) denote the total read cost from all the nodes in cluster H_x:

$ReadCost\ (R) = \sum_{P_i \in H_x} |\Gamma\ (P_i, R, l)| * A^r\ (P_i)$:

For each update in the system, the root node P^{root} needs to propagate the update to all other share holders inside H_x. Let $WriteCost$(R) denote the total update cost in H_x. Then

$WriteCost(R) = w^C * |\Gamma\ (P^{root}, R, |R|)|$.

Let Cost(R) denote the total cost of all nodes in H_x, then

$Cost\ (R) = WriteCost\ (R) + ReadCost\ (R)$.

Our goal is to determine an optimal resident set R to allocate the shares in H_x, such that Cost(R) is minimized.

3. OIRSP SOLUTIONS

In this section, we present a heuristic algorithm for OIRSP. First (in Section 3.1), we discuss some properties that are very useful for the design of the heuristic algorithm. In Section 3.2, we present the heuristic algorithm that decides which cluster should hold share replicas to minimize access cost.

3.1. Some Useful Properties

We first show that if a cluster H_x is in R^C (an optimal resident set), then H_x should hold at least l share replicas (l is the number of shares to be accessed by a read request). If H_x is in R^C and H_x has less than l shares, then read accesses from H_x will anyway need to go to another cluster to get the remaining shares. If H_x holds no share replicas, then read accesses from H_x may need to get the l shares from multiple clusters. These may result in unnecessary communication overhead.

Theorem 3.1. In a general graph $G^C, \forall_x, H_x \in G^C$, $|R_x| = 0$ or $|R_x| \geq l$.

Proof. Assume that there exists one cluster H_x in R^C, such that $|R_x| < l$. When the resident set is R^C, a read request from H_x cannot be served locally and the remaining shares have to be obtained from at least one other cluster in G^C that holds those shares. Thus, $| \delta (H_{x,} R_C| > 0$. Let us construct another resident set R^{C1}. R^{C1} is the same as R^C except that in R^{C1}, H_x holds l distinct shares. Thus, in R^{C1}, $|\delta(H_{x,}R^{C1})| = 0$. So, the read cost for read requests from H_x becomes zero. Also, in G^C, there may be clusters that read from H_x. Assume that H_x is the closest cluster in R^C of H_y (H_y is not in R^C). If the optimal resident set is R^C, then H_y needs to read from H_x and some other clusters since H_x has less than l shares. Thus, we can conclude

$ReadCost^C\ (G^C,R^C) – ReadCost^C\ (G^C,R^{C1})$

$\geq A^r(H_x) *|\delta(H_{x,}R^{C1})|$ *and, hence,*

$ReadCost^C\ (G^C,R^{C1}) < ReadCost^C\ (G^C,R^C)$.

Now let us consider the update cost. Note that we have $UpdateCost^C\ (G^C,R^C) = w^C{*}|\Gamma^C\ (R^C)|$. Because R^{C1} and R^C are actually composed of the same set of clusters, so $|\Gamma^C\ (R^{C1})| = |\Gamma^C\ (R^C)|$. Also, w^C is independent of the resident set. So, we have $UpdateCost^C\ (G^C,R^{C1}) = UpdateCost^C(G^C,R^C)$.

Theorem 3.2. The optimal resident set is a connected graph within the general graph G^C.

Proof. Assume that R^C is an optimal resident set for G^C and it is not connected. Since R^C is not a connected graph, there are two subgraphs R^{C1} and R^{C2} that are not connected. Without loss of generality, assume that cluster $H_{MSC} \in R_1^C$ and R_2^C is the closest subgraph to R_1^C in the update propagation minimal spanning tree of R^C. Since G^C is connected, at least one path existed that connects R_1^C and R_2^C. Let $\delta\ (R_1^C\ , R_2^C\ )$ denote the path connecting R_1^C and R_2^C in G^C with the minimal distance (or minimum number of hops between R_1^C and R_2^C if distance is measured by the number of hops) and let $|\delta(R_1^C\ ,\ R_2^C\ )|$ denote the distance. Since R_1^C and R_2^C are disconnected, there exists a cluster $H_x \in \delta\ (R_1^C, \cancel{R_2}^C)$ and $H_x \in R^C$.

Let us consider a new resident set R^{C1} such that R^{C1} is the same as R_C, except that all clusters on path $\delta\ (R_1^C, R_2^C)$ are in R^{C1}. For each cluster $H^x \in \delta(R_1^C\ ,R_2^C\ )$, $|\delta(H_x\ ,R^{C1})| = 0$. Together with

Theorem 3.1, we know that $\text{ReadCost}^C(G^C, R^C) < \text{ReadCost}^C(G^C, R^{C1})$. For each update in G^C, an update propagation message is propagated from R_1^C to R_2^C through $\delta(R_1^C, R_2^C)$, no matter whether R^C or R^{C1} is the residence set, since $\delta(R_1^C, R_2^C)$ is the shortest path between R_1^C and R_2^C in G^C. Thus,

$UpdateCost^C(G^C,R^{C1}) = UpdateCost^C(G^C,R^C).$

Since R^{C1} yields a lower read cost than and has the same update cost as R^C, we can conclude that $Cost^C(G^C,R^{C1}) < Cost^C(G^C,R^C)$. Thus, R^C is not a minimal residence set. And, we can conclude that the optimal resident set is a connected graph in G^C.

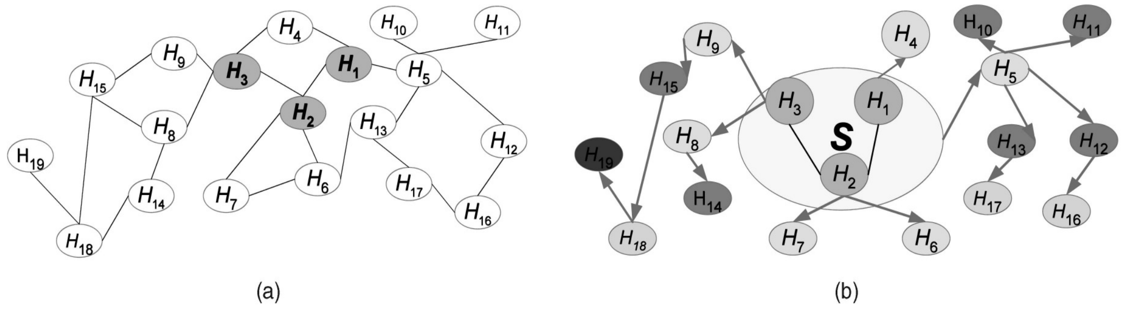

Fig. 2. Sample G^C and SPT (G^C, R^C). (a) The original G^C with $R^C = \{H_1, H_2, H_3\}$. (b) Super node S and $SPT(G^C, R^C)$ constructed by *Build_SPT*.

3.2. A Heuristic Algorithm for the OIRSP

The goal of OIRSP is to determine the optimal resident set R^C in G^C. G^C is a general graph. Each edge in G^C is considered as one hop.

It has been shown that the problem is NP-complete. Thus, we develop a heuristic algorithm to find a near-optimal solution. Our approach is to first build a minimal spanning tree in G^C with R^C being the root and then identify the cluster to be added to R^C based on the tree structure. The clusters in G^C access data hosted in R^C along the shortest paths, and these paths and the clusters form a set of the shortest path trees. Since all the nodes in R^C are connected, we view them as one virtual node S. Then, S, all clusters that are not in R^C, and all the shortest access paths form a tree rooted at S, which is denoted as $SPT(G^C, R^C)$ (an example of the tree is shown in Fig. 2b). We develop an efficient algorithm *Build_SPT* to construct $SPT(G^C,R^C)$ based on the current resident set R^C. To facilitate the identification of a new resident cluster, we also define $V^C(G^C, R^C)$ as the vicinity set of S, where $\forall\, Hx \in\!\!\!/\, V^C(G^C,R^C)$, we have $H_x \in R^C$ and H_x is a neighboring cluster of S. Note that from Theorem 3.2,we know that the clusters in R^C are connected.

Build $SPT(G^C,R^C)$ first constructs $V^C(G^C,R^C)$ by visiting all neighboring clusters of R^C. If a cluster H_x in $V^C(G^C,R^C)$ has more than one neighbor in R^C, then one of them is chosen to be the parent cluster. Next, Build $SPT(G^C, R^C)$ traverses G^C starting from clusters in $V^C(G^C,R^C)$. From a cluster H_x, it visits all H_x's neighboring clusters. Assume that H_y is a neighboring cluster of H_x. When Build_*SPT* visits H_y from H_x, it assigns H_x as H_y's parent if H_y does not have a parent. In this case, H_y is in the same tree as H_x, and H_y*'s* tree root is set to H_x*'s* (which is a cluster in R^C). Since all read requests from H_y go through the root, say H_z, $A^r(H_y)$ is added to $A^r(H_z)^1$ for later use (for new resident cluster identification). In case H_y already has a parent, the distances to S via the original parent and via H_x are compared. If H_x offers a shorter path to S, then H_y*'s* parent is reset to H_x and the corresponding adjustments are made. To achieve a faster convergence for new R^C identification, H_y*'s* parent is also changed to H_x if H_x*'s* tree root H_z has a higher value of $A^r(H_z)^1$, when the distances to S via H_y*'s* original parent and via H_x are equal. The detailed algorithm for *Build_SPT* is given in the following (assume that $V^C(G^C,R^C)$ is already identified). In the algorithm, each node H_x has several fields. H_x. *root* and H_x. *parent* are the root and parent clusters of H_x, respectively. H_x. *dist* is the distance from H_x to H_x*'s root* (at the end of the algorithm, it is the shortest distance). We also use *Next* (H_x) to denote the set of H_x*'s* neighbors.

```
Build_SPT ($G^C$, $R^C$)
{  For all $H_x$, $H_{x1}$ Є $V^C(G^C,R^C)$
{  Insert $H_x$ into Queue; $H_x$. root ◄-- $H_x$; $H_x$. dist ◄— 0;
$A^r(H_x)^1$ $A^r(H_x)$; }
While (Queue # Θ
{ $H_x$ ◄— Remove a node from Queue;
For all $H_y$, $H_y$ Є Next ($H_x$) ^ $H_y$ Є RC
{ If ($H_y$ is not marked as visited) then
{ Insert $H_y$ into Queue; $H_y$. dist ◄— $H_x$. dist + 1;
$H_y$. parent ◄— $H_x$; $H_y$. root ◄— $H_x$. root;
$A^r(H_y. root)^1$ ◄— $A^r(H_y. root)^1 + A^r(H_y)$; Mark $H_y$ as
visited; }
Else
{ If ($H_y$. dist > $H_x$. dist + 1V(($H_y$. dist=
$H_x$. dist + 1) ^ $A^r(Hy. Root)^1 < A^r(H_x. root)^1$)) then
{ $A^r(H_y. root)^1$ ◄— $A^r(H_y. root)^1 - A^r(H_y)$;
$H_y$. dist ◄— $H_x$. dist + 1; $H_y$. parent ◄— $H_x$;
$H_y$. root ◄— $H_x$. root;
$A^r(H_y. root)^1$ ◄— $A^r(H_y. root)^1 + A^r(H_y)$; } }
} } }
```

Actually, the check for H_y. *dist* > H_x .*dist* + 1 in the algorithm is not necessary since a queue is used (a node is always visited from a neighbor with the shortest distance to *S*). A sample general graph G^C with current resident set $R^C = \{H_1, H_2, H_3\}$ is shown in Fig. 2a. The corresponding $SPT(G^C, R^C)$ is shown in Fig. 2b, where R^C is represented by the super node labeled as *S*. When constructing $SPT(G^C, R^C)$, *S's* immediate neighbors, including H_4, H_5, H_6, H_7, H_8, and H_9, are visited first. H_4 is visited twice but H_1 is selected as the parent since H_4 is visited from H_1 first and there is no need for adjustment when it is visited the second time. From the clusters nearest to S, the clusters that are two hops away from S, including H_{10}, H_{11}, H_{12}, H_{13}, H_{14}, and H_{15}, are visited. Finally, the nodes that are further away from S are visited.

We develop a heuristic algorithm to find the new resident set for G^C in a greedy manner. We try to find a new resident cluster in $V^C(G^C,R^C)$ and, once found, update R^C accordingly. The algorithm is shown below. R^C is initialized to $\{H_{MSC}\}$. The algorithm first constructs $SPT(G^C,R^C)$ and identifies $V^C(G^C,R^C)$. Then, a cluster H_y with the highest $A^r(H_y)^1$ is selected. If $A^r(H_y)^1 > w^C$, then H_y is added to R^C. If $A^r(H_y)^1 \leq w^C$, then the algorithm terminates since no other nodes can be added to R^C while reducing the access communication cost. Note that, in each step, only one cluster can be added into R^C because $SPT\ (G^C,R^C)$ and $A^r(H_x)^l$ changes when R^C changes.

```
$R^C \leftarrow \{H_{MSC}\}$;
Repeat
{Build SPT $(G^C,R^C)$;
Select a cluster $H_y$, where $H_y$ has the maximum
$A^r(H_y)^l$ among all clusters in $V^C(G^C,R^C)$;
If $A^r(H_y)^l > w^C$ $R^C \leftarrow R^C \cup \{H_y\}$; }
Until $(A^r(H_y)^l \leq w^C)$
```

Theorem 3.3. In a general graph G^C, if $|R^C| > 1$, then $Cost^C\ (G^C,R^C) < Cost^C(G^C, \{H_{MSC}\})$. Furthermore, every time a new cluster H_x (H_x satisfies the cost constraint) is added to current resident set $R^{Cl}(R^{Cl} \subseteq R^C)$, the communication cost decreases, i.e., $Cost^C(G^C,R^{Cl} \cup \{H_x\}) < Cost^C(G^C, R^{Cl})$.

Proof. According to Theorem 3.1, $\forall_x$, $H_x \in R^C$, $|R_x| \geq l$. The algorithm works by adding one cluster at a t i m e . Let $R^C = \{H_1, H_2, . . ., H_n\}$, $|R^C| = n$ and $H_1 = H_{MSC}$. Assume that H_i is added at the (i- 1) th step to R^C. If we show that after adding each cluster, the cost reduces, then we can conclude that $Cost^C(G^C,R^C) < Cost^C(G^C, \{H_{MSC}\})$. We use induction to prove this.

Step 1. We show that $Cost^C\ (G^C; \{H_{MSC};H_2\}) < Cost^C(G^C, \{H_{MSC}\})$. According to the algorithm, $H_2 \in V^C(G^C, \{H_{MSC}\})$, then $UpdateCost^C(G^C,\{H_{MSC}, H_2\}) = UpdateCost^C(G^C,\{H_{MSC}\}) + w^C * |\delta(H_2,\{H_{MSC}\})|$. For each cluster H_x that reads $\{H_{MSC}\}$ through H_2, $\delta(H_x, \{H_{MSC}\})$ is the shortest path in G^C from H_x to $\{H_{MSC}\}$. It is obvious that $H_2 \in \delta(H_x, \{H_{MSC}\})$ and H_2 is the cluster on $\delta(H_x,\{H_{MSC}\})$ right next to H_{MSC}, and $|\delta(Hx,H2)| = |\delta(H_x, \{H_{MSC}\})| - |\delta(H_2, \{H_{MSC}\})|$. Any other path $\delta(H_x,H_2)^1$ or $\delta(H_x, H_{MSC})^1$ has a distance no less than $|\delta(H_x, H_2)|$. With resident set $\{H_{MSC},H_2\}$, $\delta(H_x,H_2)$ will continue to be the least distance path for cluster H_x to read from H_2 in G^C, and $\delta(H_x, \{H_{MSC}, H_2\}) = \delta(H_x, \{H_{MSC}\}) - |\delta(H_2, \{H_{MSC}\})|$. For any cluster H_x that reads $\{H_{MSC}\}$ through H_2, $\delta(H_x, \{H_{MSC}\})$ will, at least, not increase if H_2 is added into the resident set. Then, we can easily get $\text{ReadCost}^C\ (G^C, \{H_{MSC}, H_2\}) + A^r(H_2)1 \leq \text{ReadCost}^C(G^C, \{H_{MSC}\})$.

According to the heuristic resident set algorithm, we know $A^r\ (H_2)^l > w^C$. Thus, $\text{Cost}^C(G^C, \{H_{MSC}, H_2\}) - \text{Cost}^C(G^C, \{H_{MSC}\}) = \text{UpdateCost}^C(G^C, \{H_{MSC}, H_2\}) - UpdateCost^C\ (G^C, \{H_{MSC}\}) + ReadCost^C\ (G^C, \{H_{MSC}, H_2\}) - ReadCost^C(G^C,\{H_{msc}\}) \leq w^C - A^r(H_2)^l * |\delta(H_2, \{H_{MSC}\})| < 0$.

Step 2. Assume that $Cost^C\ (GC, \{H_{MSC}, H_2, . . ., H_k\}) < \text{Cost}^C(G^C, \{H_{MSC}, H_2, . . ., H_{k-1}\})$. We show that $Cost^C\ (G^C, \{H_{MSC}, H_2, . . . , H_k\}) > Cost^C(G^C, \{H_{MSC}, H_2, . . .; H_{k+1}\})$, with $k < n$. It can be seen that the proof is the same as above and we will not show it here.

By induction, we know that $Cost^C\ (G, \{H_{MSC}, H_2, . . . , H_n\}) < Cost^C\{G^C, \{H_{MSC}, H_2, . . ., H_{n-1}\})$. Thus, $Cost^C\ (G^C,R^C) < CostC(G^C, \{H_{MSC}\})$. Also, from the induction process, we can conclude that every time a new cluster Hi joins RC, the communication cost decreases, i.e., $Cost^C(G^C,R^C(\text{i-1})) < Cost^C\ (G^C,R^C)$.

4. PERFORMANCE OF THE OIRSP HEURISTIC ALGORITHM

In this section, we compare the performance of the OIRSP heuristic algorithm with the randomized K-replication, noreplication allocation, and complete replication strategies. We study the impacts of three factors: 1) the graph size, which is the number of clusters in the system; 2) the graph degree, which is the average number of neighbors of a cluster; and 3) the update/read ratio, which is the ratio of the total number of update requests in the entire system to the average number of read requests issued from a single cluster (these are the requests each cluster needs to process).

5. RELATED WORK

Replication techniques are frequently used to improve data availability and reduce client response time and communication cost. One major advantage of replication is performance improvement, which is achieved by moving data objects close to clients. In this paper, we consider the replica placement problem in data grids where critical data objects are partitioned to assure data confidentiality and integrity. The replicas of partitioned shares are dynamically allocated to improve access performance. Our approach minimizes the access cost of partitioned data in data grids, while it ensures the required data confidentiality and integrity.

It can be considered that our work complements the work in such a way that one focus on the performance issues and the other focus on the security assurance issues. Replication techniques are frequently used to improve data availability and reduce client response time and communication cost. One major advantage of replication is performance improvement, which is achieved by moving data objects close to clients.

Data assurance is defined as the probability that the data is not compromised. They consider a two-level network topology where a system is divided into clusters. It assumes that the probability that the data shares are compromised when sent cross clusters is higher than that when transmitted within the cluster or to the clients. Also, when data is secret shared but not replicated, the data
assurance level is higher than that when data is replicated but not secret shared. To achieve better data assurance, a distributed share allocation algorithm is presented to dynamically allocate the original shares to different sub networks based on the client read and write patterns.

The algorithm converges to an optimal allocation that yields maximal data assurance. It simply moves data shares to the clusters where there are more access demands. In this paper, we consider the replica placement problem in data grids where critical data objects are partitioned to assure data confidentiality and integrity. The replicas of partitioned shares are dynamically allocated to improve access performance. Our approach minimizes the access cost of partitioned data in data grids, while it ensures the required data confidentiality and integrity.

I consider the dynamic replication in this paper as by sending the selected message file from the client in to Two different levels. In the First level I consider the file to be partitioned in to three different Individual Routers like Router A, Router B and Router C. Now these three partitions will select the path dynamically, that is initially we do not know exact bandwidth for every Router. Dynamically the partitioned data will select the particular Router. In the Second level these three partitions takes three different Sub Routers i.e., Sub Router A, Sub Router B and Sub Router C dynamically. That is Router A can select either Sub Router A1 or Sub Router B1or Sub Router C1 etc .

In the same way for Sub Routers A2, B2 and C2. Every Router selects the path dynamically same as the initial message file does. Finally from these Sub Routers the individual partitions are combined and the original message is reached to the particular server without any loss of message. In these two different levels the message file from the client selects the path dynamically. This improves the data confidentiality and integrity.

6. CONCLUSIONS AND FUTURE RESEARCH

We have combined data partitioning schemes (secret sharing scheme or erasure coding scheme) with dynamic replication to achieve data survivability, security, and access performance in data grids. The replicas of the partitioned data need to be properly allocated to achieve the actual performance gains. We have developed algorithms to allocate correlated data shares in large-scale peer-to-peer data grids. To support scalability, we represent the data grid as a two-level cluster based topology and decompose the allocation problem into two subproblems: the OIRSP and OISAP.

The OIRSP determines which clusters need to maintain share replicas, and the OISAP determines the number of share replicas needed in a cluster and their placements. Heuristic algorithms are developed for the two subproblems. Several future research directions can be investigated. First, the secure storage mechanisms developed in this paper can also be used for key storage. In this alternate scheme, critical data objects are encrypted and replicated.

The encryption keys are partitioned and the key shares are replicated and distributed. To minimize the access cost, allocation of the replicas of a data object and the replicas of its key shares should be considered together. We plan to construct the cost model for this approach and expand our algorithm to find best placement solutions. Also, the two approaches (partitioning data or partitioning keys) have pros and cons in terms of storage and access cost and have different security and availability implications.

Moreover, it may be desirable to consider multiple factors for the allocation of secret shares and their replicas. Replicating data shares improves access performance but degrades security. Having more share replicas may increase the chance of shares being compromised.

Thus, it is desirable to determine the placement solutions based on multiple objectives, including performance, availability, and security.

REFERENCES

[1] M. Baker, R. Buyya, and D. Laforenza, "Grids and Grid Technology for Wide-Area Distributed Computing," Software-Practice and Experience, 2002.

[2] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services," Proc. ACM/IEEE Conf. Supercomputing (SC), 2002.

[3] I. Foster and A. Lamnitche, "On Death, Taxes, and Convergence of Peer-to-Peer and Grid Computing," Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS), 2003.

[4] V. Matossian and M. Parashar, "Enabling Peer-to-Peer Interactions for Scientific Applications on the Grid," Proc. Ninth Int'l Euro-Par Conf. (Euro-Par), 2003.

[5] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, and S. Tuecke, The Security Architecture for Open Grid Services, Version 1, 2002.

[6] V. Paxson, "End-to-End Routing Behavior in the Internet," IEEE/ACM Trans. Networking, vol. 5, no. 5, pp. 601-615, 1997.

[7] A. Mei, L.V. Mancini, and S. Jajodia, "Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems," IEEE Trans. Parallel and Distributed Systems,

vol. 14, no. 9, 2003.

[8] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist,V. Welch, I. Foster, and S. Tuecke, The Security Architecture for Open Grid Services, Version 1, 2002.

[9] www.gloriad.org/gloriad/projects/project000053.html, 2008.

[10] M. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," J. ACM, vol. 36, no. 2, 1989.

[11] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid," Proc. Second Int'l Workshop Grid Computing, 2001.

[12] A. Shamir, "How to Share a Secret," Comm. ACM, vol. 22, 1979.

[13] H. Stockinger, "Distributed Database Management Systems and the Data Grids," Proc. 18th IEEE Symp. Mass Storage Systems,2001.

[14] B.M. Thuraisingham and J.A. Maurer, "Information Survivability for Evolvable and Adaptable Real-Time Command and Control Systems," IEEE Trans. Knowledge and

Data Eng., vol. 11,no. 1, Jan. 1999.

[15] M. Tu, "A Data Management Framework for Secure and Dependable Data Grid," PhD dissertation, Univ. of Texas at Dallas, http://www.utdallas.edu/~tumh2000/ref/Thesis-Tu.pdf,July 2006.

[16] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal Placement of Replicas in Trees with Read, Write, and Storage Costs," IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 6, 2001.

[17] O. Kariv and S.L. Hakimi, "An Algorithmic Approach to Location Problems—II: The p-medians," SIAM J. Applied Math., vol. 37, no. 3, 1979.

[18] H. Krawczyk, "Distributed Fingerprints and Secure Information Dispersal," Proc. 12th Ann. ACM Symp. Principles of Distributed Computing (PODC), 1993.

[19] H. Krawczyk, "Secret Sharing Made Short," Proc. 13th Ann. Int'l Cryptology Conf. (Crypto), 1993.

[20] S. Lakshmanan, M. Ahamad, and H. Venkateswaran, "Responsive Security for Stored Data," IEEE Trans. Parallel and Distributed Systems, vol. 14,