ISSN:0975-9646

D. Subhramanya Sharma et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4) , 2012,4774 -4779

Replication of Authorized Data Objects in Data Grid

D. Subhramanya Sharma, Sudha Gontu

Dept. of CSE, Sri Sivani College Of Engineering, Srikakulam, A.P, India

***Abstract*— Data fragmentation and Secret sharing approaches have been used in distributed storage systems to ensure the confidentiality, integrity, and availability of critical information. Data fragmentation refers to approaches like erasure coding. To achieve performance goals in data accesses, these data fragmentation approaches can be combined with dynamic replication. In this paper, we consider data fragmentation (both secret sharing and erasure coding) 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 authorized data objects that are partitioned by using secret sharing scheme or data fragmentation approach 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 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. The heuristic algorithms achieve good performance in reducing communication cost and are close to optimal solutions.**

Keywords- Secure data, secret sharing, erasure coding, replication, data grids, data fragmentation.

I. 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[1]. Underlying infrastructure for data grids can generally be classified into two types: cluster based and peer-peer systems[4][5]. 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, such as DoD's Global Information Grid (GIG) for both business and military domains[2], NASA's Information Power Grid GMESS Health-Grid for medical services[3], data grids for Federal Disaster Relief, etc. These data grid applications are designed to support global collaborations that may involve large amount of information, intensive computation, real time, or non real 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. For example, consider a first responder team responding to a fire in a building with explosive chemicals. The data grid that hosts building safety information, such as the building layout and locations of dangerous chemicals and hazard containment devices, can help draw relatively safe and effective rescue plans. Delayed accesses to these data can endanger the responders as well as increase the risk to the victims or cause severe damages to the property. At the same time, the information such as location of hazardous chemicals is highly sensitive and, if falls in the hands of terrorists, could cause severe consequences. Thus, confidentiality of the critical information should be carefully protected. The above example indicates the importance of data grids and their availability, reliability, accuracy, and responsiveness. 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.

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. To improve both access performance and availability, it is necessary to replicate data and place them close to the clients, such as peer-to-peer data caching. As can be seen, replication is an effective technique for all types of data grids. Existing research works on replication in data grids investigate replica access protocols resource management and discovery techniques replica location and discovery algorithms and replica placement issues.

Replication of keys can increase its access efficiency as well as avoiding the single-point failure problem and reducing the risk of denial of service attacks, but would increase the risk of having some compromised key servers. If one of the key servers is compromised, all the critical data are essentially compromised. Beside key management issues, information leakage is another problem with the replica encryption approach[6]. Generally, a key is used to access many data objects. When a client leaves the system or its privilege for some accesses is revoked, those data objects have to be re encrypted using a new key and the new key has to be distributed to other clients. If one of the data storage servers is compromised, the storage server could retain a copy of the data encrypted using the old key. Thus, the content of long-lived data may leak over time. Therefore, additional security mechanisms are needed for sensitive data protection. 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.

2. EXISTING SYSTEM:

An existing problems in the fields of science, engineering, and business, which cannot be effectively dealt with using the current generation of supercomputers, in fact due to their size and complexity, these problems are often very numerically and/or data intensive and consequently require a variety of heterogeneous resources that are not available on a single machine. A number of teams have conducted experimental studies on the cooperative use of geographically distributed resources unified to act as a single powerful computer. This new approach is known by several names, such as meta computing, scalable computing, global computing, Internet computing, and more recently peer-to-peer or Grid computing. In grid computing schemes for data partitioning include secret sharing[8] and erasure coding[7]. Both schemes partitioned data into shares and distribute them to different processor to achieve availability and integrity. Secret sharing scheme assure confidentiality even if some share are compromised .

In erasure coding data shares can be encrypted and the encryption key can be secret shared and distributed with the data shares to assure confidentiality[9]. However changing the number of shares in data partitioning scheme is generally costly.

3. PROPOSED SYSTEM:

We consider data partitioning (both secret sharing and erasure coding) 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 partitioned by using secret sharing scheme or erasure coding scheme and/or replicated.

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 need share replicas[10] and the Optimal Intra cluster Share Allocation Problem (OISAP) that determines the number of share replicas needed in a cluster and their placements.

As in proposed system number of shares in a data partitioning scheme is costly it is necessary to add additional shares close to a group of clients to reduce the communication cost and access latency. Thus it is most effective to combine the data partitioning and replication techniques for higher performance secure storage design.

Our goal is to develop placement algorithm to allocate share replicas such that the communication cost and access latency are minimized . We introduce a heuristic algorithm for determining the clusters that should host shares .we introduce another heuristic algorithm for share allocation with in a cluster .

4 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 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 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 Read $\text{Cost}_C(G_C, R_C)$ denote the total read cost in G_C with the resident set R_C, then

Read CostC (GC , RC)= $\sum$Hx Ar (Hx) * | δ(Hx, RC) |.

Update CostC (GC ,RC) = wC * r^C(RC)|.

Let Update CostC (GC , RC) denote the total update cost in GC with the resident set RC, then

Thus, the total access cost in GC, denoted as Cost (GC,RC) is defined as follows:

CostC (GC,RC) = Update CostC (GC,RC)
+ Read CostC (GC ,RC).

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.

5.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 Hx by referring to everything in the cluster without the cluster subscript. For example, we use G = (P, E) to represent the topology graph of Hx, where P = {P1,P2, . . ,PN}. Similarly, Proot represents the root node of Hx, δ(Pi, Pj) represents the shortest path between two nodes inside Hx, and R represents the resident set of Hx

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

$$Read\ cost(R) = \sum_{pi \in Hx} (pi, R, l) * Ar(pi)$$

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

Write Cost(R) = wC * | (Proot, R, |R)|.

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

Cost (R) = Write Cost (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. Note that m≥ R ≥ l (we will prove this in the next section).We propose a heuristic algorithm, with a complexity of $O(N^3)$ to find the near optimal solution for this problem, Where N is the number of nodes in the cluster.

H^C	The set of M clusters in te system
H_X, H_Y, H_Z	Denote individual cluster 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 be 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 $\|\delta(H_x,H_y)\|$
$R^C(r^C)$	The minimal spanning tree routed at H_{msc} that connects all clusters in R^c
$A^r(H_X)\| A^W(H_X)$	The total no. 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
$R_X{}'$,$R^{C'}$	A resident set that is potentially different from R_x or R^c
$\delta(P_{X,I},P_{X,J})$	Shortest path between two nodes P_{xj} and P_{xj}
$\Upsilon(P_{X,I}$, R_X, a), $\Upsilon(P_{X,I}$, R_X, L) and $\Upsilon(P_{X,I}$, R_X, $\|r_X\|)$	The minimal spanning tree routed at P_{xi} that connects a nodes, l nodes and all the nodes in R_x
$Ar(P_{X,I})\|A^W(P_{X,I})$	The total no. of read write requests from a node $P_{x,i}$
Updatecost, Writercost®, $Updatecost^c(G^c,R^c)$	The total update cost in the entire data grid, the updatecost inside a single cluster only, and the updatecost at the cluster level only, respectively
Readcost, readcost(r), $Readcost^c(G^c,R^c)$	The total readcost in the entire data grid, the readcost inside a single cluster only, and the readcost at the cluster level only, respectively
Tcost, Cost(R), and $cost^c(G^c,R^c)$	The total update cost in the entire data grid, the accesscost inside a single cluster only, and the accesscost at the cluster level only, respectively

Table 1: Summary of the frequently used Notation

6. OIRSP SOLUTIONS:

In this section, we present a heuristic algorithm for OIRSP. First (in Section 6.1), we discuss some properties that are very useful for the design of the heuristic algorithm. In Section 6.2, we present the heuristic algorithm that decides which cluster should hold share replicas to minimize access cost.

6.1. Some Useful Properties:

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

Theorem 6.1 In a general graph G_C,V_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 GC that holds those shares. Thus, | δ (Hx, RC| > 0. Let us construct another resident set RC1. RC1 is the same as RC except that in RC1, Hx holds l distinct shares. Thus, in RC1, |δ(Hx,RC1)| = 0. So, the read cost for read requests from Hx becomes zero. Also, in GC, there may be clusters that read from Hx. Assume that Hx is the closest cluster in RC of Hy (Hy is not in RC). If the optimal resident set is RC, then Hy needs to read from Hx and some other clusters since Hx has less than l shares. Thus, we can conclude

ReadCostC (GC,RC) – ReadCostC (GC,RC1)

≥ Ar(Hx) *|δ(Hx,RC1)| and, hence,

ReadCostC (GC,RC1) < ReadCostC (GC,RC).

Now let us consider the update cost. Note that we have UpdateCostC (GC,RC) = wC*| C (RC)|. Because RC1 and RC are actually composed of the same set of clusters, so | C (RC1)| = | C (RC)|. Also, wC is independent of the resident set. So, we have UpdateCostC (GC,RC1) = UpdateCostC(GC,RC).

Theorem 6.2. The optimal resident set is a connected graph within the general graph Gc.

Proof. Assume that RC is an optimal resident set for GC and it is not connected. Since RC is not a connected graph, there are two sub graphs RC1 and RC2 that are not connected. Without loss of generality, assume that cluster HMSC R1C and R2C is the closest sub graph to R1C in the update propagation minimal spanning tree of RC. Since GC is connected, at least one path existed that connects R1C and R2C. Let δ (R1C , R2C) denote the path connecting R1C and R2C in GC with the minimal distance (or minimum number of hops between R1C and R2C if distance is measured by the number of hops) and let |δ(R1C , R2C)| denote the distance. Since R1C and R2C are disconnected, there exists a cluster Hx δ (R1C, R2C) and Hx RC.

Let us consider a new resident set RC1 such that RC1 is the same as RC, except that all clusters on path δ (R1C, R2C) are in RC1. For each cluster Hx δ(R1C ,R2C), |δ(Hx ,RC1)| = 0.

7. A HEURISTIC ALGORITHM FOR THE OIRSP

The goal of OIRSP is to determine the optimal resident set Rc in Gc. Gc is a general graph.
Each edge in Gc 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 GC with RC being the root and then identify the cluster to be added to RC based on the tree structure. The clusters in GC access data hosted in RC along the shortest paths, and these paths and the clusters form a set of the shortest path trees. Since all the nodes in RC are connected, we view them as one virtual node S. Then, S, all clusters that are not in RC, and all the shortest access paths form a tree rooted at S, which is denoted as SPT(GC, RC) (an example of the tree is shown in Fig. 2b). We develop an efficient algorithm Build_SPT to construct SPT(GC,RC) based on the current resident set RC. To facilitate the identification of a new resident cluster, we also define VC (GC, RC) as the vicinity set of S, where V Hx V C(GC,RC), we have Hx RC and Hx is a neighboring cluster of S. Note that from Theorem 6.2,we know that the clusters in RC are connected.

Build SPT (GC,RC) first constructs VC(GC,RC) by visiting all neighboring clusters of RC. If a cluster Hx in VC(GC,RC) has more than one neighbor in RC, then one of them is chosen to be the parent cluster. Next, Build SPT (GC, RC) traverses GC starting from clusters in VC(GC,RC). From a cluster Hx, it visits all Hx's neighboring clusters. Assume that Hy is a neighboring cluster of Hx. When Build_SPT visits Hy from Hx, it assigns Hx as Hy's parent if Hy does not have a parent.

In this case, Hy is in the same tree as Hx, and Hy's tree root is set to Hx's (which is a cluster in RC). Since all read requests from Hy go through the root, say Hz, Ar(Hy) is added to Ar(Hz)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 Hx are compared. If Hx offers a shorter path to S, then Hy's parent is reset to Hx and the corresponding adjustments are made. To achieve a faster convergence for new RC identification, Hy's parent is also changed to Hx if Hx's tree root Hz hasa higher value of Ar(Hz)1, when the distances to S via Hy's original parent and via Hx are equal. The detailed algorithm for Build_SPT is given in the following (assume that V C(GC,RC) is already identified). In the algorithm, each node Hx has several fields. Hx. root and Hx. parent are the root and parent clusters of Hx, respectively. Hx. dist is the distance from Hx to Hx's root (at the end of the algorithm, it is the shortest distance). We also use Next (Hx) to denote the set of Hx's neighbors.

Build_SPT(G^c,R^c)
{
For all H_x,H_x є V^c (G^c,R^c)
{ Insert H_x into queue; $H_{x,root}$ ← H_x ; $H_{x,dist}$ ←0;
$A^r(H_x)'$ ←$A^r(H_x)$;}
While (queue!=0)
{ H_x← Remove a node from queue;
For all H_y, H_y є Next (H_x) Λ H_y !є R^c
{ 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 ($H_{y.root}$)' ← A^r ($H_{y.root}$)'+A^r(H_y); Mark H_y as visited;}
Else
{
If($H_{y.}$dist > $H_{x.}$dist+1 ν ((H_y .dist= H_x.dist+1)Λ A^r($H_{y.root}$)'<A^r($H_{x.root}$)')) then
{ A^r($H_{y.}$root)'←A^r(H_y.root)'-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)'← A^r($H_{y.}$root)'+A^r(H_y);}}
}
}
}

Actually, the check for Hy. dist > Hx .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 GC with current resident set RC = {H1, H2, H3} is shown in Fig. 2a. The corresponding SPT(GC , RC) is shown in Fig. 2b, where RC is represented by the super node labeled as S. When constructing SPT (GC, RC), S's immediate neighbors, including H4, H5, H6, H7, H8, and H9, are visited first. H4 is visited twice but H1 is selected as the parent since H4 is visited from H1 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 H10, H11, H12, H13, H14, and H15, 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 GC in a greedy manner. We try to find a new resident cluster in V C (GC,RC) and, once found, update RC accordingly. The algorithm is shown below. RC is initialized to {HMSC}. The algorithm first constructs SPT (GC,RC) and identifies V C(GC,RC). Then, a cluster Hy with the highest Ar(Hy)l is selected. If Ar(Hy)1 > wC, then Hy is added to RC. If Ar (Hy)1 ≤ wC, then the algorithm terminates since no other nodes can be added to RC while reducing the access communication cost. Note that, in each step, only one cluster can be added into RC because SPT (GC,RC) and Ar(Hx)1 changes when RC changes.

Repeat ←{H_{MSC}}, R_c
{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);
Until ($A_r(H_y)_l$ ≤ w_C) ← R_c U {H_y};} if$A_r(H_y)$ > W_c R_c

Theorem 7.3. In a general graph GC, if |RC| > 1, then CostC (GC,RC) < CostC(GC, {HMSC}). Furthermore, every time a new cluster Hx (Hx satisfies the cost constraint) is added to current resident set RCl(RCl C RC), the communication cost decreases, i.e., CostC(GC,RCl U{Hx}) < CostC(GC, RCl).

Proof:. According to Theorem 6.1, V x, Hx RC, |Rx|≥ l. The algorithm works by adding one cluster at a t i m e . Let RC= {H1, H2, . . .,Hn}, |RC| = n and H1= HMSC. Assume that Hi is added at the (i- 1) th step to RC. If we show that after adding each cluster, the cost reduces, then we can conclude that CostC(GC,RC) < CostC(GC, {HMSC}). We use induction to prove this.

Step 1. We show that CostC (GC; {HMSC;H2}) < CostC(GC, {HMSC}). According to the algorithm, VC(GC, {HMSC}), then UpdateCostC(GC,{HMSC, H2})= UpdateCostC(GC,{HMSC})+ wC*H2 |δ(H2,{HMSC})|. For each cluster Hx that reads {HMSC} through H2, δ(Hx, {HMSC}) is the shortest path in GC from Hx to {HMSC}. It is obvious that H2 δ(Hx, {HMSC}) and H2 is the cluster on δ(Hx,{HMSC}) right next to HMSC, and |δ(Hx,H2)| = |δ(Hx, {HMSC})| - |δ(H2, {HMSC})|. Any other path δ(Hx,H2)1 or δ(Hx, HMSC)1 has a distance no less than |δ(Hx, H2)|. With resident set {HMSC,H2}, δ(Hx,H2) will continue to be the least distance path for cluster Hx to read from H2 in GC, and δ(Hx, {HMSC, H2}) = δ(Hx, {HMSC}) - |δ(H2,

{HMSC})|. For any cluster Hx that reads {HMSC} through H2, δ(Hx, {HMSC}) will, at least, not increase if H2 is added into the resident set. Then, we can easily get Read CostC (GC, {HMSC, H2}) + Ar (H2)Ł Read CostC(GC, {HMSC}).
According to the heuristic resident set algorithm, we know Ar (H2)l > wC. Thus, CostC(GC, {HMSC, H2}) - CostC(GC, {HMSC}) = UpdateCostC(GC, {HMSC, H2}) - UpdateCostC (GC , {HMSC}) + ReadCostC (GC, {HMSC , H2})- ReadCostC(GC,{Hmsc})≤ wC - Ar(H2)l *|δ(H2, {HMSC})|< 0.

Step 2. Assume that CostC (GC, {HMSC, H2, . . .,Hk}) < CostC(GC, {HMSC, H2, . . ., Hk-1}). We show that CostC (GC , {HMSC , H2 , . . . , Hk}) > CostC(GC, {HMSC, H2, . . .;Hk+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 CostC (G, {HMSC, H2, . . . , Hn}) < CostC{GC, {HMSC, H2, . . ., Hn-1}). Thus, CostC (GC,RC) < CostC(GC, {HMSC}). Also, from the induction process, we can conclude that every time a new cluster Hi joins RC, the communication cost decreases, i.e.,CostC(GC,RC(i-1)) < CostC (GC,RC).

8 .OISAP SOLUTIONS:

Now, we only consider the cost inside a single cluster Hx. As discussed in Section 2, the topology of Hx is a tree, denoted as T. For simplicity, we define the distance of each edge in T uniformly as one hop. In the following, we first show two important properties of the OISAP problem with a tree topology. Then, we give a heuristic algorithm to decide the numbers of shares needed in Hx and where to place them.

If the Hx's resident set R is not connected, then R consists of multiple disconnected sub resident set R1;R2; . . .;Rn, where n > 1, and each sub resident set is connected. We say R is j þ connected in Hx, if and only if minðjRijÞ _ j, where j > 0, Ri, for all i _ n, are sub graphs in Hx, and jRij is the number of server nodes in Ri. We define Ri <pos Rj as follows: I f Ri <pos Rj, then 9Py, Pz 2 Hx, where Py 2 Ri ^ Pz 2 Rj, such that Pz is an ancestor of Py in T. Informally, nodes in Rj are closer to the root than nodes in Ri. Otherwise, Ri _pos Rj.

Theorem8.1: Let RS denote the resident set computed by the node joing phase of SDP tree. If the constraint |r|<m is removed,then RS is the optimal resident set such that cost(RS) is minimal.
Proof :Assume that RS is not optimal suppose that there exists an optimal resident set RS' such that cost(RS)<cost(RS)'.two cases exist.
Case1: RS c RS'. According to theorem, and the SDP tree algorithm RS and RS' are both connected , and node in RS' – RS must be a desended of some node in RS. Otherwise, SDP-tree would have added the node into RS. Let p_y denote a node such that it's parent node is in RS and p_y !є RS while p_y belongs to RS'. According to SDP-tree, IF p_y is not added in RS, it is only because that adding p_y will increase access cost. From lemma , we know that adding any subset of desendants of p_y together with p_y would also increase the cost. Thus, there exists no SR' such that RS c RS'.
Case2: RS !c RS' ^ RS!= RS'. Let p_y be the first node that SDP- tree adds, such that p_y belongs to RS. And p_y not belongs to RS'. Let R' denote the resident set that SDP-tree computed before adding p_y. Note that R'!=$^{\theta}$ (because R' contains at least P^{root}) , nad R' c RS'. According theorem, RS' is a connected subgraph in T. since p_y !є RS' then no node in the sub tree rooted at p_y should be in RS'. According to the SDP-tree algorithm, f p_y є RS then diff(p_y) is minimal among the neighbouring nodes of r'.

Two cases should be considered.1) cost(R')ᴗ(p_y)<cost(R').the node p_y is a neighbour of some node in R'. This means cost(RS'ᴗ{ p_y})<cost(RS'), which is a contradiction to the assumption.2) |R'|<l, diff(p_y)>=0,and diff(p_y) is minimal among the neighbouring nodes of R'. According to lemma, for any node p_x such that p_xєRS' and $p_{x!}$єR',0<= diff(p_y)<= diff(p_x). If for node p_x such that p_x є RS' and P_x !є RS, diff(p_y)= diff(p_x) , then there exist no $p_z$such that p_zєRS' and p_x!єR',0<= diff(p_y)<= diff(p_x). If for any node p_x such that p_xRS' and diff(p_x)<= diff(p_z). Otherwise p_x is added to RS before P_z. thus cost(RS)<=cost(RS'), which is contradictory to the assumption. If there exists some node p_x such that p_x є RS', p_x !є RS'and diff(p_y) < diff(p_x), then let p_z be a leaf node of the tree composed only by nodes in RS' and p_z is a desecandent of p_x . according to lemma, diff(p_y)< diff(p_x)<= diff(p_z). Now, construct another resident set RS'' such that RS'' =RS'ᴗ{p_y}-{p_z}. We know that cost(RS'')<cost(RS'), which contradics the assumption. If there exists some node Px such that Px є2 RS', Px є RS, and diff(Py)Þ < diff(PxÞ,) then let Pz be a leaf node of the tree composed only by nodes in RS^1 and p_z is descendant of p_x.

9. CONCLUSION:

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 sub problems. Experimental studies show that the heuristic algorithms achieve good performance in reducing communication cost and are close to optimal solutions.

10. FUTURE ENHANCEMENTS:

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. We plan to investigate their tradeoffs and some preliminary analysis results are available in [38]. 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]. Global Information Grid, Wikipedia.

[3]. http://www.ccrl-nece.de/gemss/reports.shtml, 2008.

[4]. 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.

[5]. 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.

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

[7]. T. Wu, M. Malkin, and D. Boneh, “Building Intrusion Tolerant Applications,” Proc. DARPA Information Survivability Conf. and Exposition (DISCEX), 2000.

[8]. A. Shamir, “How to Share a Secret,” Comm. ACM, vol. 22, 1979.

[9]. J. Wylie, M. Bakkaloglu, V. Pandurangan, M. Bigrigg, S. Oguz, K. Tew, C. Williams, G. Ganger, and P. Khosla, “Selecting the Right Data Distribution Scheme for a Survivable Storage System,” Technical Report CMU-CS-01-120, Carnegie Mellon Univ., 2000.

[10]. L. Xiao, I. Yen, Y. Zhang, and F. Bastani, “Evaluating Dependable Distributed Storage Systems,” Proc. Int’l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA), 2007.