International Journal of Recent Trends in Engineering & Research
ISSN(ONLINE) : 2455-1457
IMPACT FACTOR : 3.344

Data Objects Replication Protection in Data Grid Using Advanced Encryption Standard Algorithm

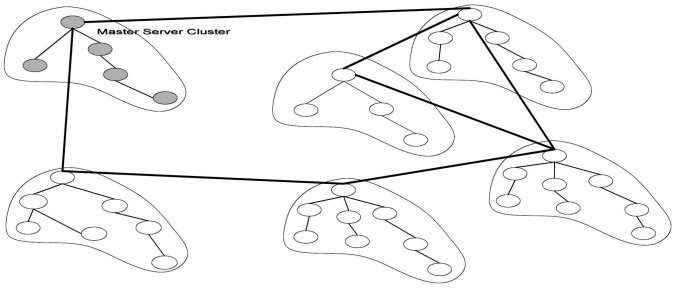
Kadambari Raghu Ram[1], Satish Kumar Satti[2]
[1]*Assistant Professor(C), Department of CSE, University College of Engineering (A), JNTUK, Kakinada*
[2]*Assistant Professor, Department of CSE, B V C College of Engineering, Rajahmundry*

Abstract: We consider combine replication and data partitioning schemes to assure data availability, confidentiality, and timely accesses for data grid applications. Data objects are partitioned into shares and shares and dispersed. The shares may be replicated to achieve better performance and availability. 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. Multiple cluster networks have been formed by implementing K Means clustering algorithm. We develop models for assessing confidentiality, availability, and communication cost for different placements and use the metrics to guide placement decisions. A new probabilistic security Advanced Encryption Standard (AES) and an efficient Base64 Encoder and Decoder algorithm have been developed. Due to the nature of contradicting goals, we model the placement decision problem as a multi-objective problem and use a Bell Man Ford algorithm to determine Optimal Path solution and it is achieve.
Keywords: Data grids, Replication, K Means Cluster, AES, Base64, Bell Man Ford.

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. 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, NASA's Information Power Grid GMESS Health-Grid for medical services, 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.



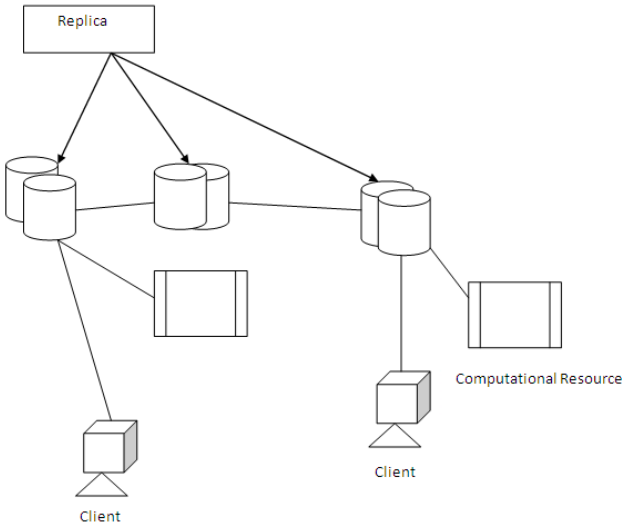
A Sample Graph of our System Topology

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. 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.

II. SYSTEM MODEL AND PROBLEM SPECIFICATION

In this paper, we consider Data Security Efficiency and bandwidth. The data in a data grid can be located at a single site or multiple sites where each site can be its own administrative domain governed by a set of security restrictions as to who may access the data. Likewise, multiple replicas of the data may be distributed throughout the grid outside their original administrative domain and the security restrictions placed on the original data for who may access it must be equally applied to the replicas.



System Architecture

Data access services work hand in hand with the data transfer service to provide security, access controls and management of any data transfers within the data grid. Security services provide mechanisms for authentication of users to ensure they are properly identified. Common forms of security for authentication can include the use of passwords or Kerberos (protocol). Authorization services are the mechanisms that control what the user is able to access after being identified through authentication. Common forms of authorization mechanisms can be as simple as file permissions. However, need for more stringent controlled access to data is done using Access Control Lists (ACLs), Role-Based Access Control (RBAC) and Tasked-Based Authorization Controls (TBAC). These types of controls can be used to provide granular access to files to include limits on access times, duration of access to granular controls that determine which files can be read or written to. The final data access service that might be present to protect the confidentiality of the data transport is encryption and this encryption process could be done by using AES (Advanced Encryption Standard) Algorithm. The most common form of encryption for this task has been the use of SSL while in transport. While all of these access services operate within the data grid, access services within the various administrative domains that host the datasets will still stay in place to enforce access rules. The data grid access services must be in step with the administrative domains access services for this to work.

The use of replicas allows multiple users faster access to datasets and the preservation of bandwidth since replicas can often be placed strategically close to or within sites where users need them. Bellman Ford Algorithm is implemented for bandwidth guarantee which is helpful to compute path effectively with high bandwidth frequency.

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 $\lvert\delta(H_x, H_y)\rvert$
$T^C(R^C)$	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)
$R_x', R^{C'}$	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 $P_{x,i}$ and $P_{x,j}$ (or simply $\delta(P_i, P_j)$ later)
$T(P_{x,i}, R_x, \alpha)$, $T(P_{x,i}, R_x, l)$, and $T(P_{x,i}, R_x, \lvert R_x\rvert)$	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

Summary of the Frequently Used Notation

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

Let $UpdateCost^C(G^C,R^C)$ denote the total update cost in G^C with the resident set R^C, then

$UpdateCost^C(G^C,R^C) = \omega^C * |\Gamma^C(R^C)|$.

Thus, the total access cost in G^C, denoted as $Cost^C(G^C,R^C)$ 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 inG^C, such that the communication cost $Cost^C(G^C,R^C)$ is minimized. The optimal resident set problem in general graph is proven to be NP-complete. We propose a heuristic algorithm to compute a near-optimal resident set for a general graph. We will show that the heuristic algorithm has an $O(M^3)$ complexity, where M is the number of clusters in the system. We will also prove that our heuristic algorithm is optimal in a tree network, with time complexity O(M).

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 ω^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, . . . ; 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. Note that the simplification will be used. In the situation where multiple clusters are considered, the original notation is used. Note that we only need to consider clusters with l or more share replicas for this subproblem OISAP.

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

$ReadCost(R) = \Sigma_{Pi \in Hx} | \Gamma(P_i, R, l)| * A^r(P_i)$.

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

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

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. Note that $m \geq |R| \geq 1$ (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.

III. IMPLEMENTATION LEVEL SECURED ALGORITHMS

➢ **Constraints in Analysis**

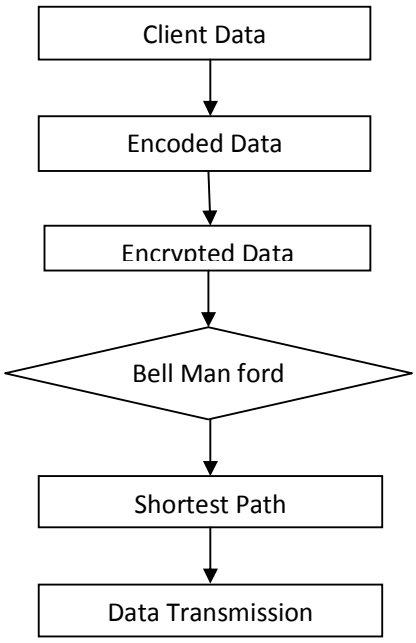
- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

➢ **Constraints in Design**

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses

Constraints in Implementation:

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.



Process Flow

Bell Man Ford Algorithm:

Bell Man Ford Algorithm is mainly used to finding shortest path among nodes in network which is used to increase bandwidth during data transmission in network.
The following steps have been described in Bell Man Ford algorithm for finding shortest path.

- Initialization
  - Each node has 1 row for each destination d
  - Distance of node d to itself is zero: $D_d(d)=0$
  - Distance of other node j to d is infinite: $D_j(d)= \propto$, for $j \neq d$
  - Next node $n_j = -1$ since not yet defined.

- Send Step
  - Send new distance vector to immediate neighbors across local link

- Receive Step
  - For each destination d, find the next hop that gives the minimum distance to d,

- $Min_j \{ C_{ij} + D_j(d) \}$

- Replace old $(n_j, D_i(d))$ by new $(n_j^*, D_j^*(d))$ if new next node or distance found
  - Go to send step

Base64

Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. The term Base64 originates from a specific MIME content transfer encoding.
Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remain intact without modification during transport.

AES Algorithm:

Cipher Key

AES is a symmetric key encryption algorithm. It uses a cipher key whose length is 128 bits, 192 bits or 256 bits. The algorithm with cipher key length 128, 192, 256 bits is denoted AES-128, AES-192, AES-256, respectively.

State

The process of encrypting (decryption) of plaintext (ciphertext) to ciphertext (plaintext) generates intermediate 128-bit results. These intermediate results are referred to as the State.

Data Blocks

AES operates on an input data block of 128 bits and its output is also a data block of 128 bits. During the process, the data block is call a State.

Round Keys

AES-128, AES192, AES-256 algorithm expends the a cipher key 10, 12, 14 round keys from, respectively. Each Round key is 128-bit in length. The algorithm for deriving the round keys from the cipher key is the called Key Expansion.

Addroundkey

AddRoundKey is a (128-bit, 128-bit) Æ 128-bit transformation, which is defined, is a bit-wise xor of its two arguments. In the AES flow, the arguments are the State and the round key.

Key Generation Algorithm

Parameters
Nb = 4
Nk = number of doublewords in the cipher key (4, 6, 8 for AES-128, AES-192, AES-256, resp.)
Nr = number of rounds in the cipher (Nr=10, 12, 14 AES-128, AES-192, AES-256, resp.)
The KeyExpansion routine
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
word tmp
i = 0
while (i < Nk) Advanced Encryption Standard (AES) Instructions Set White Paper
12
w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
i = i+1
end while
i = Nk
while (i < Nb * (Nr+1)]
tmp = w[i-1]
if (i mod Nk = 0)
tmp = SubWord(RotWord(tmp)) xor RCON[i/Nk]
else
if (Nk > 6 and i mod Nk = 4)
tmp = SubWord(tmp)
end if
w[i] = w[i-Nk] xor tmp
i = i + 1
end while

AES Encryption:

;; Data is a 128-bit block to be encrypted. The round keys are stored in Round_Key_Encrypt

Tmp = Add Round Key (Data, Round_Key_Encrypt [0])
For round = 1-9 or 1-11 or 1-13:
Tmp = ShiftRows (Tmp)
Tmp = SubBytes (Tmp)
Tmp = MixColumns (Tmp)
Tmp = AddRoundKey (Tmp, Round_Key_Encrypt [round])
end loop
Tmp = ShiftRows (Tmp)
Tmp = SubBytes (Tmp)
Tmp = Add RoundKey (Tmp, Round_Key_Encrypt [10 or 12 or 14])
Result = Tmp

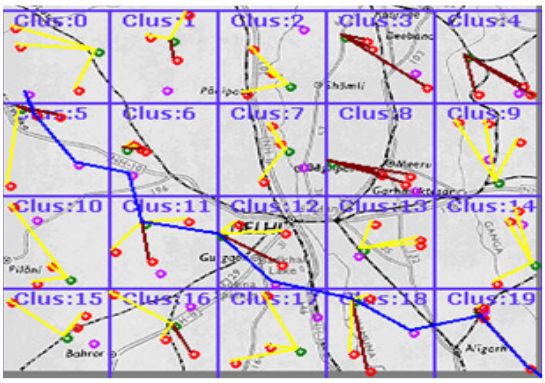
AES Decryption:

;; Data is a 128-bit block to be decrypt. The round keys are stored in Round_Key_Decrypt

Tmp = Add Round Key (Data, Round_Key_Decrypt [0])
For round = 1-9 or 1-11 or 1-13:
Tmp = InvShiftRows (Tmp)
Tmp = InvSubBytes (Tmp)
Tmp = InvMixColumns (Tmp)
Tmp = AddRoundKey (Tmp, Round_Key_Decrypt [round])
end loop
Tmp = InvShift Rows (Tmp)
Tmp = InvSubBytes (Tmp)
Tmp = AddRoundKey (Tmp, Round_Key_Decrypt [10 or 12 or 14])
Result = Tmp

IV. EXPERIMENTAL STUDY

We conduct experiments to evaluate the performance of the heuristic algorithms for secret share allocation and also the secret sharing and provide high bandwidth. The OIRSP heuristic algorithm is compared with the randomized K-replication, no-replication allocation, and complete replication strategies, to study its performance and effectiveness on reducing communication cost at the cluster level. In the randomized K-replication, share replicas are randomly allocated among K clusters, where K is the number of clusters holding replicas computed by the OIRSP heuristic algorithm. In the complete replication strategy, share replicas are allocated in every cluster. In the no-replication allocation strategy, there is no replication in any cluster.



The SDP-tree algorithm for OISAP is compared with the optimal allocation algorithm and randomized M -replication, to see how well the SDP-tree algorithm performs in terms of reducing communication cost within a cluster. In the randomized M-replication, M shares are randomly allocated among the nodes in a cluster, where M is computed by SDP-tree and it is the number of nodes that hold shares. The underlying network topology for the experimental studies is created by using a topology generator, Inet. Inet has a lower bound on the total number of nodes in the network. We removed the bound so that the graph with different number of clusters (or nodes) can be created. We have also modified the generator on read/write request generation for each node. The numbers of read and write requests on the nodes in the system are generated randomly following a uniform distribution. The metric we consider is the communication cost, which is the product of the number of messages and the number of hops along the message propagation path. To avoid biased access patterns and topology structures, we repeat the experimental steps 100 times. The final result is the average of the 100 trials.

V. 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 full replication all servers keep a complete set of the data objects. This is frequently not feasible for a large data set. More importantly, full replication schemes might incur unnecessary communication overhead when both read and update accesses are considered. In partial replication, each server maintains a subset of data objects, depending on the tradeoff of the read and update costs. For both full and partial replication, an important issue is where to place the replicas. Many research efforts have been devoted to the optimal placement of distributed files, data objects, and other network services. Facility location problem is one big branch. It considers the optimal placement of one file or data object (set) and supports read operations only. Many research works are done along this direction, such as p-median problem and the approximate p-median problem. Another branch of research on optimal data placement is the file allocation (optimal resident set) problem. It considers both read and update operations. A lot of research efforts have been devoted to this research branch, the optimal file allocation problem in tree networks, general graph networks, completely connected networks, and ring networks, respectively, the ADR algorithm, and the optimal placement of k replicas in a tree network.

All the replica placement works discussed above only consider the allocation of one single data object, which is equivalent to considering multiple independent data objects. None of them addressed the allocation of multiple dependent objects such as data that are partitioned into multiple shares. Recently, some research works have been conducted on distributed secure data systems, dynamically allocate the original shares to different subnet-works 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. Performance issues, such as access communication cost and access latency, are not considered in this paper. The work in considers a secure data storage system that offers various levels of security guarantees, assuming data are secret shared. The entire set of shares are replicated and statically distributed to the nodes in the network. It focuses on secure access protocols rather than on replica placement.

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.

VII. FUTURE ENHANCEMENT

Providing security to the Desktop Data Grid is critical for its deployment in production environments, where its full potential can be developed in terms of storage (i.e. for backup and caching) and computing power. However, it is mandatory to establish guarantees over the security provided to the data and metadata being managed in these systems in order to avoid unnecessary risks. Towards this goal, we presented a proposal for enhancing the overall security of the Desktop Data Grid, with a protocol that makes use of three basic techniques (cryptography, data fragmentation and a quantitative security metric) to cope with untrusted Volunteer nodes participating in these environments. The proposed protocol was focused on vulnerabilities indicated by applying an extended security analysis framework, developed in our previous research for the Data Grid storage services

VIII. CONCLUSION

We have addressed the problem of Data Security in Data Grid environments by investigating the use of a new set of AES (Advanced Encryption Standard) and Base64 Encoding algorithm that can be used to improve Security, reliability, protection of original user data. We have also used bell man ford algorithm that dynamically evaluates the replica placement policy by comparing the replica maintenance costs and data access gains of creating a replica at any given location and its improve bandwidth.

REFERENCES

1. 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.
2. 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.
3. H. Stockinger, “Distributed Database Management Systems and the Data Grids,” Proc. 18th IEEE Symp. Mass Storage Systems, 2001.
4. A. Samar and H. Stockinger, “Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication,” Proc. IASTED Int’l Conf. Applied Informatics (AI), 2001.
5. S. Lakshmanan, M. Ahamad, and H. Venkateswaran, “Responsive Security for Stored Data,” IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 9, 2003.
6. K. Ranganathan and I. Foster, “Identifying Dynamic Replication Strategies for a High Performance Data Grid,” Proc. Second Int’l Workshop Grid Computing, 2001.
7. 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.
8. Manghui Tu, Peng Li , Bhavani Thuraisingham and Latifur Khan, Transactions on Dependable and Secure Computing, vol. 7, no. 1, 2010.

