Optimal Number of Replicas in Data Grid Environment

Yasser Mansouri
Department of Computer Engineering,
Ferdowsi University of Mashhad
E-mail: ya_ma20, @stu-mail.um.ac.ir

Reza Monsefi
Department of Computer Engineering,
Ferdowsi University of Mashhad
E-mail: rmonsefi@ferdowsi.um.ac.ir

Abstract- **Data replication is a promising technique for increasing access performance and data availability in Data Grid (DG) systems. Current work on data replication in Grid systems focuses on infrastructure for replication and mechanisms for creating or deleting replicas. The important problem of determining minimum number of replicas, as well as their locations in DG has not been well studied.**

In this paper, we propose an algorithm is formulated by using dynamic programming-based algorithm to find Optimal Number of Replicas (ONR) of a dataset over DG systems, such that the read cost (i.e. transferring object over the Data Grid systems to the end-user)plus the cost of storage (i.e. site building cost) is minimized. We have also proposed a sketch of the proof for our algorithm and its integrity.

I. INTRODUCTION

Grid computing is a novel and emerging paradigm, whose goal is providing Virtual Organization (VO) of geographically, distributed users with software/hardware infrastructures that allow the effective sharing of computational and storage resources. We distinguish between computational grid and Data Grid: computational grids address computationally intensive applications that deal with complex and the intensive computational problems usually on relatively small datasets whereas Data Grids address the needs of data intensive applications deal with the evaluation and mining of large amounts of data in the terabyte and Peta byte range. Therefore, data is the most important resource in a DG, where user's jobs require access to a large quantity of data.

One class of grid computing and the focus of this paper is DG that provides geographically distributed storage resources to large Computational scientific applications require accessing, analyzing and storing large amounts of data. Such applications warrant special treatment to scale across distributed computing Platforms like the grid and avid data access bottlenecks. In areas such as genomics, drug discovery, High Energy physics (HEP), astrophysics and climate change modeling, large objects are generated, collected and stored in geographically distributed locations [1,2,3]. **These objects having large amounts of data, so the cost of maintaining a local copy of object on each site that needs the data is extremely expensive.** Moreover, these objects are mostly read-only, since they are input data to the applications for various purposes, such as benchmarking, identification and classification. The need to access and manage several peta byte data in grid environments and high delay in wide area network, which is the base most grid systems, access optimization (i.e. minimizing access cost) and data availability becomes keys challenges to be addressed.

Data replication is an excellent technique to move and cache data close to users. Replication reduces access latency and bandwidth consumption. It also facilitates load balancing and improves reliability by creating multiple data copies. There is a fair amount of work on data replication in DG systems. However, most of the existing works have focused on infrastructure for replications and mechanisms for creating/deleting replicas [4,5,6,7,8,9,10]. We believe that, optimizing overall access cost and reducing the cost of replication are two conflicting goals. So, a strategic for determining optimal number of replicas, as well as their locations; such that overall access cost to be minimized, is necessary.

Some work exist on the placement of replicas in parallel and distributed systems with regular network topologies such as tree, hyper cubes and ring, etc [11,12,13]. However, these algorithms not be directly used by DG environments due to hierarchical network structure and special data access patterns in DG environments that are not common in traditional parallel system.

A number of early works have been studied placing replicas for DG environments. In [14], the author presents a heuristic algorithm, named *proportional share replication* for the placement problem. However, the algorithm does not guarantee to find optimal placement for replicas. Liu et al. [15, 16], suggested efficient algorithm for selecting strategic location for placing the replica so that the workload among these replicas is balanced. Also, they proposed algorithm to decide the minimum number of replicas required when the maximum workload capacity of each replica server is known. These algorithms ensure that locality requirements from the user satisfied. To the best of our knowledge, none of the existing work, has considered structure and special data access in DG, for unknown

number of replicas; find optimal number of replicas for DG tree systems ,such that communication[1] and storage cost is minimized.

However, we present a novel dynamic algorithm on geographical replications of objects on hierarchical (i.e. tree structure) DG systems for read-only applications. This algorithm finds ONR and location of replicas, such that overall access cost is minimized. The presented algorithm includes polynomial time and space complexity as well.

The rest of the paper is organized as follows. Section 2 formulates minimum replication cost problem for hierarchical DG.A polynomial Optimal Number of Replicas (ONR) solution to the problem is presented in section 3. Section 4 presents sketch of the proof for our algorithm and its numerical example in the next section. Finally, section 6 concludes the paper.

II. DATA GRID MODEL AND BASIC DEFINITION

First, we describe DG Model: We will consider hierarchical DG model in this paper, due to its simplicity and close resemblance to the hierarchical management, usually found in a grid environment. For instance, in LCG (World-Wide Large Hardon Collider Computing Grid)[17] project 150 institutes from 30 countries form in a grid environment. The grid environment is organized as a hierarchical structure, with CERN (the European organization for Nuclear Research) as tier-0, or the root of grid environment. There are 11 tiers-1 sites that are connected directly to CERN and are located in different countries. These sites in tier-1, data are obtained from CERN in LHC. Other tier-2 sites in LCG hierarchy receive data from its corresponding tier-1 site. GriPhyN[18] is another example of hierarchical structure grid including multiple tiers and initially (first) all data are produced in tier-o. Tier-1 in this type of data grid includes several national centers; and below that there are regional centers. As a result, this paper focuses on tree topology.

In this model, a user of a local site at the leaf, accesses an object as follows: First he/she tries to locate the object replica locally. If the object replica is not present, he/she goes to the parent node up the tree to find if a replica is there. That is, the user's request goes up the tree and uses the first replica encountered along the path toward the root. If there is no replica along the way, the hub (i.e. the root of tree) will serve this request. However, in this article, we assumed that all of the tree nodes, including the internal nodes, could request data.

Now, we introduced the notations and definitions used in this paper. Consider M objects hosted by a DG system whose nodes are connected to form a network represented by a Tree $T_r=(V,E)$,where V is the set of nodes(i.e. server or client) and E is the set of physical links between the nodes. Moreover, r is the root of DG system, which we name it "original server" (the hub) and assuming that initially, all M objects are included. Moreover *Anc (v)* is the set of ancestors of node v.

Not that, an object replica can be placed in any tree nodes except the hub r. The entire tree leaves are local sites, where user can access objects stored in DG system. Consider for each object i $(1 \le i \le M)$, every node $v \in T_r$ is associated with a non-negative read rate $r_{v,i}$, which is the number of access during a certain period of time, where node v, requests object i. let $d(u,v)$ be a non-negative cost assigned to link $(u,v) \in E$,which could be interpreted as delay, link cost, hop count, etc. a weight $S_i(v)$ is associated with each node $v \in V$,representing the cost of storing a copy of object i (or site building cost) in node v.

At this time, we are going to calculate total cost of the system, which is the sum of the overall read cost and the overall storage cost. Suppose that the nodes of T issue read request for an object i, and that replicas of that object i can be stored at multiple nodes of T. The set of nodes at which replicas of that object i are places is called an Optimal Location of Replicas for object i (i.e. OLR_i).
The read cost of OLR_i is the cost of servicing the read requests issued from all the nodes of T for object i, is given by:

$$\sum_{v \in V} r_{v,i} \cdot d(v, c(v, OLR_i)) \qquad (1)$$

Where we define $c(v,OLR_i)$ to be the lowest ancestor of $v \in T_r$ that is contained in OLR_i, i.e., the first node in OLR_i that is seen while going up from v to root r. This node, which is $c(v, OLR_i)$ may be located in height l, top of the node v, which denoted with v_l and $d(v,v_l)$ equal with the cost sum over l links. Clearly moreover, node containing replica of object i, does not request data from ancestor. The storage cost of OLR_i is the cost of placing replicas of that object i at all the nodes in OLR_i and is given by:

$$\sum_{v \in ORL_i} S_i(v) \qquad (2)$$

Thus, the total cost for an Optimal Location of Replica (i.e. OLR_i) for T_r is:

$$Cost^i(OLR_i, V) = \sum_{v \in V} r_{v,i} \cdot d(v, c(v, OLR_i)) + \sum_{v \in OLR_i} S_i(v) \qquad (3)$$

Therefore, the total cost for M object, which we desire to be minimum is given by:

$$\sum_{i=1}^{M} cost^i(OLR_i, V) \qquad (4)$$

Thus, the Optimization Problem (OP) can be formally defined as follows:

OP: *Given a tree network of DG T_r= (V,E)and M objects, finding ONR_i and OLR_i for all $1 \le i \le M$, $OLR_i \subseteq V$ and $r \in OLR_i$, such that minimize $\sum_{i=1}^{M} cost^i(OLR_i, V)$, where $Cost^i(OLR_i,V)$ is given (3).*

[1] Communication cost and read cost is interchangeable.

III. OPTIMAL NUMBER OF REPLICAS (ONR) ALGORITHM

In this section, we present a solution to the ONR and as well as Optimal Location of Replicas (OLR), for object i in a DG Tree T_r, such that total cost (given in ("3")) is minimized. With out of generality, we assume that initially, all objects are located in root of DG Tree. This assumption is completely in accordance with general state of the problem in real world (for instance GriPhyN or European DG).

At this time, we propose a new dynamic algorithm to the ONR. As show in fig. (1.a), we consider a more generalized problem of ONR in a sub-tree rooted at node v, assuming the lowest ancestor of v that has replica is node v_l. This node (i.e. v_l) is located in distance l link from node v. We assume that, *min_cost (v, v_l)*-value is the minimum cost of the sub-tree rooted at v, where the next replica of object i up the tree is at distance l link from v. Moreover, *OLR* (v, v_l) is an optimal solution for placing replicas in sub-tree T_v.

In order to determine optimal number of replicas for object i, in a way that total cost (i.e." (3)") to be minimum; two cases for calculation *min_cost (v, v_l)*-value for all nodes $v \in T_r$ and for each $v_l \in Anc(V)$ is considered, as follows:

Case1: we assume that in sub-tree rooted at v, no replica i is located in node v. Therefore, in this case value of *min-cost (v,v_l)* equals to sum of following costs:

1) Replication cost in v's children, while we have optimum number of replica and minimum replication cost. In this case, it's clear that the lowest ancestor of v's children is v_l, which contains object i and data are read from it. So, replication cost of children's v is $\sum_{z \in child(v)} \min\_cost(z, v_l)$ where $Z(v)$ be the set of v's children.
2) Reading cost of node v from the lowest its ancestor (i.e. v_l) which contains object i. So, reading cost of node v is $r_{v,i}.d(v,v_l)$.

Therefore, in this case recurrence function can be as follows (see fig. 1.a):

$$min\_cost(v,v_l) = c_1(v,v_1) = \sum_{z \in child(v)} \min\_cost(z,v_l) + r_{v,i}\, d(v,v_l) \quad (5)$$

case2: we assume that in sub-tree rooted in v, one replica of object i is located in node v. Thus, in this case value of *min-cost (v, v_l)* equals to sum of following costs:

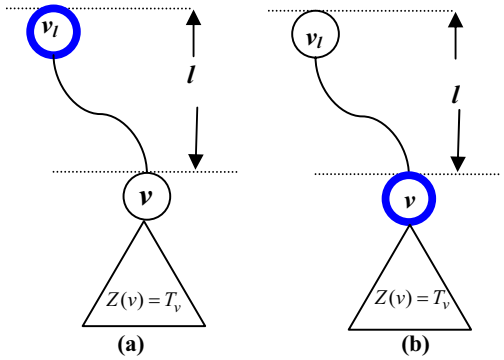

Figure.1. Description of dynamic programming algorithm for C(v,v_l) in a DG Tree: (a). No replica at node v,(b) replica at node v.

1) Replication cost in v's children, while we have optimum number of replica and minimum replication cost. In this case, the lowest ancestor, which contains object i and data are read from it by v's children, is node v (i.e. data are read from father node).thus, is $\sum_{z \in child(v)} \min\_cost(z,v)$.
2) Reading cost of node v from the lowest its ancestor (i.e. v_1 node) which contains object i. in this case (i.e. case 2) object i is read once, and is replicated in node v. Thus, read cost of node v is $d(v,v_l)$.
3) Storage cost object i in node v. that is, $S_i(v)$.

Therefore, in case 2 recurrence functions can be follows (see fig. 1.b):

$$min\_cost(v,v_l) = c_2(v,v_1) = S_i(v) + d(v,v_1) + \sum_{z \in child(v)} \min\_cost(z,v) \quad (6)$$

Now, we consider general DG tree with n node, and the traversal all the nodes in reverse post-order. That is, with starting from leaf's tree, we calculate all *min_cost (v,v_l)*-values. Notice that, these values, for all DG tree nodes, considering all ancestors of node, is calculated.

Above discussion leads to the following algorithm. Therefore, *OLR (v,v_l)* is trivial if v is a leaf in T_v. In this case, if $s_i(v) + d(v,v_l) \succ r_{v,i}.d(v,v_l)$,no replica need to be placed at v. Otherwise, if $s_i(v) + d(v,v_l) \prec r_{v,i}.d(v,v_l)$, a replica should be placed at v. For each none-leaf (i.e. internal node) v in T_v, we compare two *min-cost(v,v_l)* in(("5")and("6")).Thus, if $c_1(v,v_1) \prec c_2(v,v_1)$, no replica need to be placed at v (see case 1); otherwise, if $c_1(v,v_1) \succ c_2(v,v_1)$, a replica should be placed at v. Thus, the recurrences for dynamic algorithm are given by:

min_cost (v,vl)=

$$\begin{cases} r_{v,i}.d(v,v_l) & \text{if } v \text{ is a leaf and } s_i(v) + d(v,v_l) \succ r_{v,i}.d(v,v_l) \\ s_i(v) + d(v,v_l) & \text{if } v \text{ is a leaf and } s_i(v) + d(v,v_l) \prec r_{v,i}.d(v,v_l) \\ \min[(s_i(v) + d(v,v_l) + \sum_{z \in child(v)} \min\_cost(z,v)), & \\ \quad (r_{v,i}.d(v,v_l) + \sum_{z \in child(v)} \min\_cost(z,v_l))] & \text{if } v \text{ is not a leaf} \end{cases} \quad (7)$$

OLR (v,v_l)=

$$\begin{cases} \Phi & \text{if v is a leaf and } s_i(v)+d(v,v_l) \succ r_{v,i}\ .d(v,v_l) \\ v & \text{if v is a leaf and } s_i(v)+d(v,v_l) \prec r_{v,i}\ .d(v,v_l) \\ \bigcup_{z\in Z(v)} OLR(z,v_l) & \text{if v not leaf and } c_1(v,v_l) \le c_2(v,v_l) \\ v \bigcup_{z\in Z(v)} OLR(z,v) & \text{if v is not leaf and } c_1(v,v_l) \succ c_2(v,v_l) \end{cases} \quad (8)$$

IV. SKETCH OF PROOF

In this section, we present a sketch of proof to the ONR algorithm. As shown in Fig. 1.a, we consider a more generalized problem of placing replicas in a sub-tree rooted at node *v*, assuming the lowest ancestor of *v* that has a replica is node v_l. This new problem is formally defined as follows:

Definition: let node v_l be ancestor of node *v* in tree *T* and T_v be the sub-tree of *T*, which rooted at node *v* (see Fig.1.a).Assume a replica of object is located at node v_l. The problem of finding a replication strategy, i.e. a subset *OLR (v, v_l)* $\subseteq T_v$ with *ONR*, such that *min_cost(OLR* (*v*, v_l))= $\sum_{v\in V} r_{v,j}\, d(v, c(v, OLR_i)) + \sum_{v\in OLR_i} S_i(v)$ is minimized.This problem is referred to as (v, v_l))-optimization problem.

Bye creating a dummy parent p for the root r,a DG Tree $T_r=(V,E)$ can be converted into a new tree $T^+(v \cup \{p\}, E \cup (r,p))$, where *d(r,p)=0*.

Simply, it can be found that the min-cost (v,v_1)-values replica placement problem in T equals to the (r,p)-optimization problem in T^+. Therefore, in order to develop a dynamic programming algorithm, theorem 1 proves that an optimal solution to the (v,v_l)-optimization problem contain optimal solutions to some sub- problems.

Therome1. Assume three nodes v, v_l and z in tree T, where v_l is an ancestor of *v* and *z* is a child of node *v* (see Fig. 2). Let T_z be the set of nodes in the sub-tree of *T* rooted at *z*. Let *OLR(v, v_l)* , *OLR(z,v_l)* and *OLR(z, v_l)* be optimal solutions to the *(v,v_l),(z,v)* and *(z,v_l)*-optimization problems ,respectively.

Therefore, we prove that, (1) if $v \in$ *OLR (v,v_l)*, then the replication strategy $OLR^+ = (OLR(v, v_l) - T_z) \cup OLR(z, v)$ is also an optimal solution to the (v, v_l)-optimization problem; otherwise ,(2) if $v \notin$ *OLR(v, v_l)* , then replication strategy $OLR^{++}(v, v_l)=(OLR(v, v_l)-T_z) \cup OLR(z, v_l)$ is also an optimal solution to the*(v, v_l)*-optimization problem.

Proof: To prove claim(1), we show that, *min_cost (OLR (v,v_l))=min_cost(OLR$^+$(v,v_l))*.since *OLR(z,v)* is an optimal solution to the (z,v)-optimization problem, we have min_cost (OLR (z,v) $\le$ min_cost(OLR(v, OLR(z,v)) $\cap T_z$)).Meanwhile, we can consider *OLR (v, v_l)* as two disjoint subset. It means that, *OLR (v,v_l)-T_z* and *OLR(z,v)*. Thus, min_cost ($OLR^+(v,v_l)$=

min_cost (OLR (v,v_l)-T_z)+min_cost OLR(z,v))
$\le$min_cost (OLR (v, v_l)-T_z)+min_cost OLR(v,v_l) $\cap T_z$)
=min_cost (OLR (v, v_l)). (9)

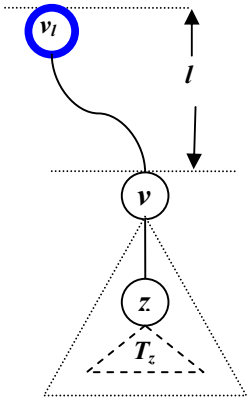

Figure.2.

On the other hand, *OLR (v, v_l)* is an optimal solution to the (v, v_l)-optimization problem thus,

min_cost (OLR (v, v_l)) $\le$ *min_cost (OLR$^+$(v,v_l))*. (10)

We conclude from ("9") and ("10"): *min_cost (OLR (v, v_l)) =min_cost (OLR$^+$(v, v_l))*.Thus, OLR$^+$ (v, v_l) is also an optimal solution to the (v, v_l)-optimization problem.

Proving claim (2) is very similar to that for claim (1). Considering the assumption of theorem1,*OLR(z, v_l)* is an optimal solution to the *(z, v_l)*-optimization problem, we have min_cost*(OLR(z, v_l))* $\le$ min_cost(OLR(v, v_1) $\cap T_z$)). Note that OLR^{++} can be divided into two disjoint subsets: OLR(v,v_1)-T_z and OLR(z,v_1).thus, *min_cost(OLR^{++}(v,v_l)*

= min_cost (OLR^{++}(v,v_1)-T_z)+min_cost(z,v_1))
$\le$ min_cost(OLR(v,v_1)- T_z)+min_cost(OLR(v,v_1) $\cap T_z$)
=min_cost(OLR(v,v_1). (11)

The optimality of OLR(v,v_1) to the (v,v_1)-optimization problem implies that,

min_cost (OLR (v,v_1)) $\le$ *min_cost(OLR^{++}(v,v_l)* (12)

with comparison ("11") and("12"), we have,

min_cost (OLR (v,v_1))= *min_cost(OLR^{++}(v,v_l)*.

Therefore, OLR^{++} is also an optimal solution to the (v, v_l)-optimization problem. Hence, Theorem1 is proven. Now, we conclude two following properties from theorem 2.

Theorem2.Assume node v_l be an ancestrator of node *v* in tree *T*, in which one replica of object i is located, and Z(v) be the set of v's children. Let *OLR (v, v_l)* be an optimal solution to the *(v, v_l)*-optimization problem.

1. if v $\in$ *OLR(v, v_l)* ,then the replication strategy {V} $\bigcup_{z\in Z(v)}$ *OLR (z,v)* is also an optimal solution to the (v, v_l)-optimization problem, where OLR(z,v) is an optimal solution to the (z,v)-optimization problem.
2. Otherwise, if v $\notin$ OLR (v, v_l), then the replication strategy $\bigcup_{z\in Z(v)}$ *OLR (z,v)* is also an optimal to the (v, v_l)-optimization problem, where OLR(v, v_l) is an optimal solution to the (z, v_l)-optimization problem.

Proof: We assume that sub-tree T_v has *k* children. That is,*Z(v)* = {z1,z2,….,zk}. In order to prove properties 1 in theorem 2, if x $\in$ OLR(x,y) and by iteratively applying

claim 1 of theorem1, replication strategies are all optimal solutions to the (v, v_l)-optimization problem as follows:
(OLR(v, v_l)- T_{Z_1}) $\cup$ OLR(z,v),

(OLR(x,y)- T_{Z_1} -T_{Z_2}), $\cup$ OLR(z_1,v) $\cup$ OLR(z_2,v),.....

$$(\text{OLR}(v, v_l)\text{-}\bigcup_{i=1}^{k} T_{Z_i}) \cup (\bigcup_{i=1}^{k} \text{OLR}(z_i,v)). \quad (13)$$

It's clear that, (OLR (v, v_l)- $\bigcup_{i=1}^{k} T_{Z_i}$)={v} because v $\in$ OLR(v,v_l).

Therefore, ("13") = {v} $\cup$ $\bigcup_{i=1}^{k}$ OLR (z_i,v) is an optimal solution to the(v, v_l)-optimization problem, and properties 1 is proven.

Proving properties 2 is completely similar to that for properties 1. Therefore, assuming that if v $\notin$ OLR (v, v_l) and by iteratively applying claim 2 of theorem 1, replication strategy are all optimal solution to the (v, v_l)-optimization problem as follows:
(OLR (v, v_l)- T_{z_1}) $\cup$ OLR (z,v),

(OLR(v, v_l)- T_{z_1} -T_{z_2}), $\cup$ OLR(z_1,v) $\cup$ OLR(z_2,v)

,.....

$$(\text{OLR}(v, v_l)\text{-}\bigcup_{i=1}^{k} T_{Z_i}) \cup (\bigcup_{i=1}^{k} \text{OLR}(z_i,v)). \quad (14)$$

Since v $\notin$ *OLR (v, v_l)*, we have (OLR (v, v_l)- (OLR(v, v_l)- $\bigcup_{i=1}^{k} T_{Z_i}$)= Φ . Thus, ("14") = $\bigcup_{i=1}^{k}$ OLR (z_i,v) is an optimal solution ,and properties 2 is proven. Hence, the theorem 2 is proven.

■

Finally, we analysis the complexity of ONR dynamic algorithm for a DG Tree in the following Theorem.

Theorem3.Let T_r be a DG Tree with n nodes. We can find optimal number of replicas for T_r in $O(n^2)$ time and space complexity ,such that equations (4) is minimized.

Proof:

ONR algorithm computes each *min_cost(v, v_l)* and *OLR(v, v_l)* in *O(Z(v))* ,where *Z(v)* is the number of v's children . Moreover, it is needed to calculate *O(Anc(v))* in the forms of *OLR (v,v_l)* and *min_cost (v,v_l)* for each $v \in V$, where *Anc(v)* is the number of v's ancestrator T^+. Therefore, the total time complexity of ONR algorithm is given by:

$$O(\sum_{v \in V} (Z(v).Anc(v)) \le O(\sum_{v \in V} |V|.Anc(v))$$

$$= (|V|.\sum_{v \in V} Anc(v)) = o(|V|^2).$$

Since maximum $O(|V|^2)$ OLR-entries and *min_cost -values* should be calculated. The worst case space complexity is $O(|V|^2)$.

V. NUMRICAL EXAMPLE

We consider an example DG Tree given Fig 3. There are 11 nodes (including the root begin the server. The link cost is given as *d(u,v)* ,which node *u* is the parent of node *v*(for example in Fig. 3,*d(N,H)=3*). The node read rate (r_v) and storage cost (s(v)) are given as (r_v,s(v)), Which is shown in fig.1.For instance, node *E* has read rate and storage cost 2 and 3, respectively (i.e. (r_E*=2,s(E)=3*).

We were interested in finding ONR and OLR for object i in DG tree Fig.3 .In this DG Tree, we assume there is one object in tree's root (i.e. object *i*). According to equation 7, our algorithm, calculate min_cost (v,vl)-values.

Now, let us focus on table 1. As indicated in table 1, corresponding to each (v,v_l)-entry, there is one numeric value that is *min_cost(v,v_l)*. This value is the minimum of $c_1(v,v_l)$ and $c_2(v,v_l)$ values.

For example, min_cost(N,H)-value corresponding to (N,H)-entry is 3.Since node N is a leaf and $c_1(N,H)=s(N)+d(N,H)=5 \succ c_2(N,H)=r_N$ .d(N,H)=3.(see "(7)").Thus ,no replica of object is located within node N. So node N reads object from its ancestor (i.e. node H).For the same node(i.e. v=N) and its ancestor, that is v_l=C with distance *l=2* ,min_cost(N,C) shows 5.Since node N is a leaf and $c_1(N,C)=s(N)+d(N,C)=7 \succ c_2(N,C)=r_N$. d(N,C)= 5(see "(7)").So in this case, no replica of object is located in node N.

Take another example. Consider (K,G)-entry which its corresponding min_cost(K,G)-value is 4.According to ("7") ; node K is a leaf and $c_1(K,G)=s(K)+d(K,G)=4 \prec c_2(K,G)=r_K$ .d(K,G)=8.Thus , one replica of object is located in node K.

Now, look at the internal nodes of DG tree. For example, min_cost (H,C)-value corresponding to (H,C)-entry is 15.According to equation ("7"), min_cost (H,C)-value is calculated as follows:

Since node H is not a leaf, thus min_cost (H,C)=min(c_1(H,C),c_2(H,C)) ,which$c_1(H,C)=s(H)+d(H,C)+\sum_{z \in child\ H} \min_cost(z,H)$ =3+2+3(min_cost(M,H))+ 3(min_cost(N,H))+4(min_cost(O,H))=15.

and

$c_2(H,C)=r_H$ .d(H,C)+ $\sum_{z \in childH} \min\_cost(z,C)$ =2+5(min_ _cost (M,C))+5(min_cost(N,C)) +6 (min_cost (O,C))=18.

Thus, considering min_cost (H,C)-value, one replica is located within node H. Similarly, according to the ("7"), we calculate all min_cost (v,v_l)-values in table1 by traversal all nodes in reverse post-order , That is, beginning with tree leaves.

Now, considering table1, we determine OLR with breadth-first traversal, from top to the bottom of the tree. For example, according to ("8") and min_cost (B,r)=8,one replica of object is located in node B. Similarly, considering min_cost (D,r),min_cost(C.r) and ("8"), one replica of object is placed on nodes C and D. Therefore,

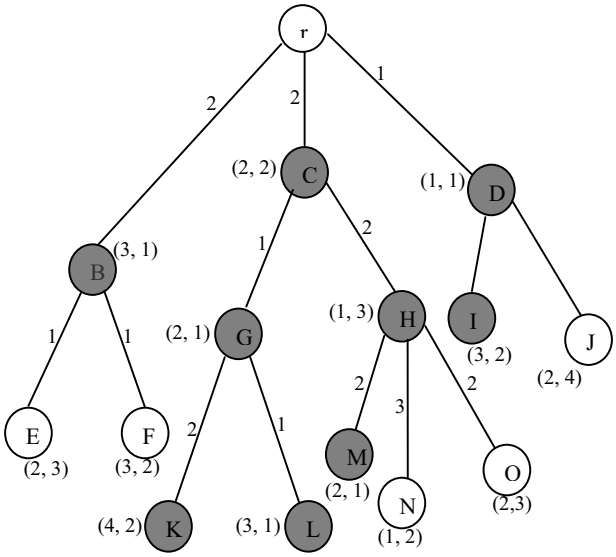

Figure3.General DG Tree example. rating read and storage cost are given as(r_v,S_v)

Table1.The optimal solution for the example General DG Tree

(v,v_l)	**c_1(v,v_l)**	**c_2(v,v_l**	**min_cost(c_1,c_2)**
(O,H)	5	4	4
(O,C)	7	8	7
(O,r)	9	12	9
(N,H)	5	3	3
(N,C)	7	5	5
(N,r)	9	7	7
(M,H)	3	4	3
(M,C)	5	8	5
(M,F)	7	12	7
(L,G)	2	3	2
(L,C)	3	6	3
(L,r)	5	12	5
(K,G)	4	8	4
(K,C)	5	12	5
(K,r)	7	10	7
(F,B)	3	3	3
(F,r)	5	9	5
(E,B)	4	2	2
(E,r)	5	9	5
(J,D)	7	9	7
(J,r)	8	8	8
(I,D)	4	6	4
(I,r)	5	9	5
(H,C)	15	19	15
(H,r)	17	27	17
(G,C)	8	10	8
(G,r)	10	18	10
(B,r)	8	17	8
(C,r)	27	31	27
(D,r)	12	14	12

considering min-cost (v,v_l)-values in table 1 and (“8”), we will determine other replicas location.

So, for DG Tree in Fig.3, ONR is 9(except node r) and OLR shows as “shaded’’ nodes in Fig.3 , which total cost for replication in this DG tree is 83.

VI. CONCLUTION

In this paper, we have investigated the Optimal Number of Replicas (ONR) and Optimal Locations of Replicas (OLR) in a DG Tree system. In particular, we give a novel dynamic algorithm for ONR and OLR, such that the read and storage cost is minimized.

We formulate ONR and OLR problems by dynamic programming-based algorithm. Our algorithm takes time and space complexity at worst-case $O(n^2)$, which n is number of nodes in the DG Tree.

REFERENCE

[1] B. Allcock, J. Bester, J. Bresnahan, A. L.Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke, *Data management and transfer in high performance computational grid environments*, Parallel Com-puting Journal 28 (2002), no. 3, 749{771.

[2] D. Bosio, J. Casey, A. Frohner, and L. Guy et al, Next generation eu datagrid data manage-ment services, Computing in high energy physics (CHEP2003), March 2003.

[3] A. Chervenak et. al, Giggle: A framework for constructing scalable replica location services,Proc. of the ACM/ IEEE SuperComputing Con-ference, November 2002.

[4] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe. Wide area data replication for scientific collaborations. In In Proceedings of the 6th International Workshop on Grid Computing, November 2005.

[5] W. B. David, D. G. Cameron, L. Capozza, A. P. Millar, K. Stocklinger, and F. Zini. Simulation of dynamic grid rdeplication strategies in optorsim. In In Proceedings of 3rd Intl IEEEWorkshop on Grid Computing, pages 46–57, 2002.

[6] W. B. David. Evaluation of an economy-based file replication strategy for a data grid. In International Workshop on Agent based Cluster and Grid Computing, pages 120–126, 2003.

[7] M. Deris, A. J.H., and H. Suzuri. An efficient replicated data access approach for large-scale distributed systems. In IEEE International Symposium on Cluster Computing and the Grid, April 2004.

[8] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman. Data replication strategies in grid environments. In In Proceedings of 5th International Conference on Algorithms andArchitecture for Parallel Processing, pages 378–383, 2002.

[9] K. Ranganathan, A. Iamnitchi, and I. Foste. Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In In 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 376–381, 2002.

[10] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and object eplication in data grids.In In 10th IEEE Symposium on High Performance and Distributed Computing, pages 305–314, 2001.

[11] M. M. Bae and B. Bose. Resource placement in torus-based networks. IEEE Transactions on Computers, 46(10):1083–1092, October 1997.
[12] K. Kalpakis, K. Dasgupta, and O. Wolfson. Optimal placement of replicas in trees with read, write, and storage costs. IEEE Transactions on Parallel and Distributed Systems, 12(6):628–637, June 2001.
[13] N.-F. Tzeng and G.-L. Feng. Resource allocation in cube network systems based on the covering radius. IEEE Transactions on Parallel and Distributed Systems, 7(4):328–342, April 1996.
[14] J. H. Abawajy. Placement of file replicas in data grid environments. In ICCS 2004, Lecture Notes in Computer Science3038, pages 66–73, 2004.
[15] P. Liu and j.j. Wu. Optimal Replica Placement Strategy for Hierarchical Data Grid Systems. Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), Volume 00,2006
[16] P. Liu, Y.-F. Lin, and J.-J. Wu. Optimal placement of replicas in data grid environments with locality assurance. In International Conference on Parallel and Distributed Systems (ICPADS). IEEE Computer Society Press, 2006.
[17] World wide LHC computing Grid .http//lcg.web.cern.ch./lcg/.
[18] The GriPhyN Project, http://www.Griphyn.org.