

Optimal Number of Replicas with QoS Assurance in Data Grid Environment*

Yasser Mansouri, Reza Monsefi
Department of Computer Engineering
Ferdowsi University of Mashhad-Iran

E-mail: ya_ma20@stu-mail.um.ac.ir, rmonsefi@ferdowsi.um.ac.ir

Abstract

Optimizing the use of grid resources is critical for users to effectively exploit a Data Grid (DG). Data replication is considered as a major technique for increasing access performance and data availability in DG systems. Current works on data replications in Grid systems focuses on infrastructure for replication mechanism for creating or deleting replicas. One of the challenges in data replication is determining Optimal Number of Replicas (ONR) with Quality of Service (QoS) assurance, as well as their Optimal Location of Replicas (OLR) in DG.

In this paper, we propose an algorithm that finds ONR of an object over DG systems, such that the overall communication and storage cost is minimized. This algorithm ensures that QoS required from the users are satisfied. In addition, we proposed a sketch of the proof for our algorithm and its integrity.

1. Introduction

Generally the field of Grid research can be divided into two large sub-domains: Computational and Data Grid (DG). Although a Computational Grid is a natural extension of cluster computers where large computing tasks have to be computed at distributed computing resources. A DG deals with the efficient management, placement and replication of large amounts of data.

In this paper, we focus on DG that connects a collection of geographically distributed computers and storage resources that may be located in different parts of a country or even across the globe, and enables the users to share data and other resources. For example, scientists working on application domains, such as High Energy Physics (HEP), earth observation, astrophysics and climate change modeling, generate large objects that are collected and stored in geographically distributed locations[1,2,3]. Since these objects have large amounts of data, the cost of maintaining a local copy of objects on each site that needs the data is extremely expensive. In addition,

these objects are mostly read-only and we seldom write in them (these are input data to the applications for various purposes, such as benchmarking, identification and classification [4]).

Data replication is excellent technique to move and cache data close to users. Replication reduces access latency and bandwidth consumption. A fair amount of work on data replication in DG systems have been reported. However, most of the existing works have focused on infrastructure for replications and mechanisms for creating/deleting replicas [5, 6, 7, and 8]. We believe that optimizing the overall access cost and reducing the cost of replication (i.e., build and maintenance of a server) are two conflicting goals. On the other hand, considering the importance of response time for users and business-oriented applications, there is an increasing demand to support QoS in DG. Thus, a strategy for determining ONR, with QoS assurance and minimizing the overall replication cost is necessary.

Some works exist on the placement of replicas in parallel and distributed systems with regular network topologies such as tree, hyper-cubes and ring, etc [9, 10, and 11]. However, these algorithms are not 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. Moreover, QoS has never been considered in these works.

A number of early works have considered placing replicas for DG environments. In [12], the author presents a heuristic algorithm, named *proportional share replication* for the placement problems. However, the algorithm does not guarantee to find optimal placement for replicas. Liu et al. [4] have suggested efficient algorithm for selecting strategic location for placing the replicas so that the workload among these replicas are balanced. In addition, they have proposed algorithm in [13] to decide the minimum number of replicas required, when the maximum workload capacity of each replica server is known. This algorithm ensures that a locality requirement (QoS) from the user is satisfied. This algorithm has been developed in [14] and they deal

*The present research work has been partially sponsored by Iranian Telecommunication Research Center (ITRC), Tehran, Iran.

Contract No:T500,12602

with the optimality of QoS and bandwidth constraints. However, the objective function of the two mentioned algorithms is different from that of our algorithm, since storage cost has not been considered in these two algorithms. Storage cost cannot be ignored because the sizes of objects in DG are too vast. In addition, our algorithm proves to have a better time complexity in comparison to these two recent works [13, 14].

Here, we present a novel 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 the communication and storage cost is minimized. Meanwhile, our algorithm ensures that locality requirements i.e. QoS requested by the users are satisfied.

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 with assuring required QoS for each user; is presented in section 3. Section 4 presents sketch of the proof for our algorithm. Finally, section 5 concludes the overall work.

2. DG Systems Model and Notations for ONR Problem

First, we describe DG model where we consider hierarchical DG model in our work, due to its simplicity and close resemblance to the hierarchical (tree) management, usually found in a grid environment. For instance, LCG (World-Wide Large Hardon Collider Computing Grid) [15] and GriPhyN [16]. Therefore, this paper focuses on tree topology.

In this model, a user of a local site at the leaf, accesses an object as follows: First (s)he tries to locate the object replica locally. If the object replica is not present, (s)he goes to the parent node up the tree to find if a replica exists. 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 path, the hub (i.e. the root of tree) will serve this request. However, in this paper, we assume that all of the tree nodes (including the internal nodes) could request data. We also assume that each client's request has a QoS (i.e. locality assurance or sometimes-said range limit). No less of generality, we can also consider the distance (i.e. number of hops) between a client and its server as QoS. That is, if the object can be retrieved by client c from server v within distance $d(c,v) \leq q(c)$, then the QoS requirement is satisfied. Otherwise, it is violated. For example, in

Fig.1 the user at node n_1 tries to access an object within $q(n_1)=3$. Whence, the user could not find the object locally. Therefore, (s)he tries the parent node n_2 , which contains no data. Then, the request reaches node n_3 where it is served by the replica in there. Since $d(n_1,n_3)=d(n_1,n_2)+d(n_2,n_3) = 1+2 = 3$. Therefore, QoS requirement of node n_1 is met. However, if $q(n_1)=2$, then we fail to serve the request by this $q(n_1)$, because, there is no replica placed at node n_2 . Therefore, QoS requirement of node n_1 is not satisfied, and data must be retrieved from node n_3 and be replicated in node n_1 . Formally, we define that a client's request can reach a server if sum of distance over links between the client and the nearest replica along the path to the root is no more than its QoS (i.e. range limit).

Now, we introduce the notations and definitions used in this article. A connected and undirected Tree models the DG systems $T_r = (V, E)$, where V is the set of nodes (i.e. server or client) and E is the set of physical/logical 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 within it. Additionally, $Anc(v)$ is the set of ancestors of node v .

We also consider for each object $i(1 \leq i \leq M)$, 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 ($T=5msec.$), where node v requests object i and $q(v)$ is QoS of node v . 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 server building and maintenance cost) in node v . No less of generality, we can also consider the

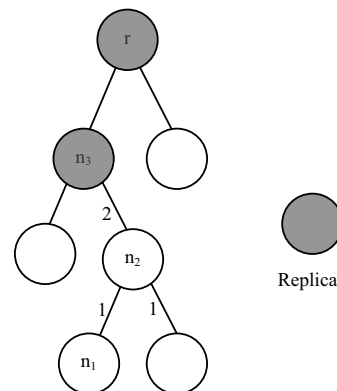


Figure.1. A DG tree

distance (i.e., number of hops) between a client and its server as QoS. So, assume that QoS constraint is defined as follows:

$$\forall c \in \text{client}, \forall v \in \text{server}, d(c, v) \leq q_i(c) \quad (1)$$

Where, the distance $d(c, v)$ is the sum of distances over links between client c and server v ; $q_i(c)$ is the QoS requirement for client c , which has requested object i .

At this time, we are going to calculate the total cost of the system, which is the sum of the overall read and the 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 placed 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:

$$\sum_{v \in V} r_{v,i} \cdot d(v, c(v, OLR_i)) \quad (2)$$

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 the request 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 (that is, $c(v, OLR_i)$ and $d(v, v_l)$ equal to the sum of the distances 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:

$$\sum_{v \in OLR_i} S_i(v) \quad (3)$$

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

$$\text{cost}^i(OLR_i, V) = \sum_{v \in V} r_{v,i} \cdot d(v, v_l) + \sum_{v \in OLR_i} S_i(v) \quad (4)$$

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

$$\sum_{i=1}^M \text{cost}^i(OLR_i, V) \quad (5)$$

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

OP: Given a tree network of DG $T_r = (V, E)$ and M objects. Find ONR_i and $OLR_i \subseteq V$ and $r \in OLR_i$, for all $1 \leq i \leq M$, such that to minimize $\sum_{i=1}^M \text{cost}^i(OLR_i, V)$, where $\text{cost}^i(OLR_i, V)$ is given by (4). Meanwhile, QoS for each user should be met.

3. Optimal Number of Replicas (ONR) with QoS Assurance

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 (4) is minimized and QoS for each client is satisfied. No less 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.

At this time, we propose a new algorithm for ONR. As shown in Fig. (2.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 in 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 the total cost i.e., (4), to be minimum; two cases for calculation $\min_cost(v, v_l)$ -value for all nodes $v \in T_r$ and for each $v_l \in \text{Anc}(v)$ 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 the following costs:

- 1) Replication cost in v 's children, while we have optimum number of replica and minimum replication cost. In this case, it is clear that the lowest ancestor of v 's children is v_l , which contains object i and data are read from it. Therefore, replication cost of v 's children is $\sum_{z \in Z(v)} \min_cost(z, v_l)$ where $Z(v)$ is the set of v 's children.

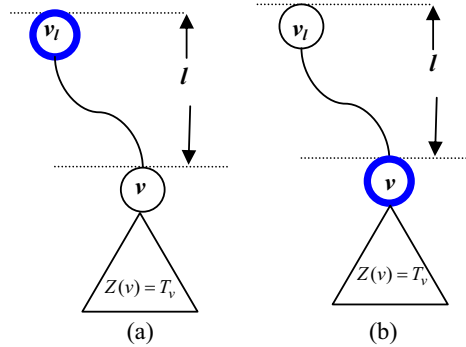


Figure.2. Description of algorithm for $\min_cost(v, v_l)$ in a DG Tree: (a). No replica at node v , (b) replica at node v .

- 2) Reading cost of node v from the lowest of its ancestor (i.e., v_l) which contains object i .

Therefore, reading cost of node v is $r_{v,i} \cdot d(v, v_l)$.

Therefore, in this case recurrence function can be as follows (see fig. 2.a):

$$\begin{aligned} \min_cost(v, v_l) &= c_1(v, v_l) \\ &= r_{v,i} \cdot d(v, v_l) + \sum_{z \in Z(v)} \min_cost(z, v_l) \quad (6) \end{aligned}$$

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 the following costs:

- 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., object is read from parent). That is, $\sum_{z \in Z(v)} \min_cost(z, v)$.
- 2) Reading cost of node v from the lowest of its ancestor (i.e., v_l node) which contains object i . In this case, 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 , i.e., $S_i(v)$.

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

$$\begin{aligned} \min_cost(v, v_l) &= c_1(v, v_l) \\ &= \sum_{z \in Z(v)} \min_cost(z, v) + d(v, v_l) + S_i(v) \quad (7) \end{aligned}$$

Now, we consider general DG tree with n node, and the traversal of all the nodes in reverse post-order. That is, starting from tree's leaf, we calculate all $\min_cost(v, v_l)$ -values. Note that, these values for all DG tree nodes considering all ancestors of node, is calculated. Therefore, $OLR(v, v_l)$ is trivial if v is a leaf in T_v . In this case, if $d(v, v_l) \leq q(v)$ **and** $S_i(v) + d(v, v_l) > r_{v,i} \cdot d(v, v_l)$, no replica need to be placed at v . Otherwise, if $d(v, v_l) > q(v)$ **or** $S_i(v) + d(v, v_l) < r_{v,i} \cdot d(v, v_l)$, a replica should be placed at v . For each non-leaf (i.e. internal node) v in T_v , we compare two $\min_cost(v, v_l)$ in (6) and (7). Thus, if $d(v, v_l) \leq q(v)$ **and** $c_1(v, v_l) < c_2(v, v_l)$, no replica need to be placed at v (see Case1); otherwise, if $d(v, v_l) > q(v)$ **or** $c_1(v, v_l) > c_2(v, v_l)$, a replica should be placed at v . Thus, the recurrences for algorithm are given by:

$$\min_cost(v, v_l) = \left\{ \begin{array}{ll} r_{v,i} \cdot d(v, v_l) & \text{if } v \text{ is a leaf and } [d(v, v_l) \leq q(v) \text{ and } S_i(v) + d(v, v_l) \geq r_{v,i} \cdot d(v, v_l)] \\ S_i(v) + d(v, v_l) & \text{if } v \text{ is a leaf and } [d(v, v_l) > q(v) \text{ or } S_i(v) + d(v, v_l) < r_{v,i} \cdot d(v, v_l)] \\ c_1(v, v_l) & \text{if } v \text{ is not a leaf and } [d(v, v_l) \leq q(v) \text{ and } c_1(v, v_l) \leq c_2(v, v_l)] \\ c_2(v, v_l) & \text{if } v \text{ is not a leaf and } [d(v, v_l) > q(v) \text{ or } c_1(v, v_l) > c_2(v, v_l)] \end{array} \right\} \quad (8)$$

$$OLR(v, v_l) = \left\{ \begin{array}{ll} \phi & \text{if } v \text{ is a leaf and } [d(v, v_l) \leq q(v) \text{ and } S_i(v) + d(v, v_l) \geq r_{v,i} \cdot d(v, v_l)] \\ v & \text{if } v \text{ is a leaf and } [d(v, v_l) > q(v) \text{ or } S_i(v) + d(v, v_l) < r_{v,i} \cdot d(v, v_l)] \\ \bigcup_{z \in Z(v)} OLR(z, v_l) & \text{if } v \text{ is not a leaf and } [d(v, v_l) \leq q(v) \text{ and } c_1(v, v_l) \leq c_2(v, v_l)] \\ v \bigcup_{z \in Z(v)} OLR(z, v) & \text{if } v \text{ is not a leaf and } [d(v, v_l) > q(v) \text{ or } c_1(v, v_l) > c_2(v, v_l)] \end{array} \right\} \quad (9)$$

4. Sketch of Proof and Complexity to the ONR Algorithm with QoS Assurance

In this section, we present a sketch of proof to the ONR algorithm with QoS assurance. As show in Fig.3, we consider a more generalized problem of placing replicas in a sub-tree rooted at node v , with a string length connected to it. The string consist of the nodes (v_1, v_2, \dots, v_l) and the edges are $\{e_v = v_{v-l} = (v_v, v_{v-l}) \mid v=1, 2, \dots, l\}$ such that $v_0=v$ is the original root of T_v . The string length is l and on side of it is connected to the root T_v . At node v_l , there is a server with a replica of the object. Meanwhile, the storage cost of an object in the string S_l is not included in the total cost of tree T_v .

However, the communication cost from the string is included. These assumptions are completing in accordance with general state of the problem. Therefore, in our algorithm (see section 3) treatment of a sub-tree T_v corresponds to the treatment of (its root) v , and a string of length corresponds to the initial distance of sub-tree from the server, which contains a replica of the object.

Moreover, we assume minimum string length that is $l=1$ to prove the problem because the algorithm with $l=1$ has finished and it is only required to calculate the value of (v, v_l) for the root.

Our proof of ONR algorithm with QoS assurance is based on induction and assumption that ONR

algorithm has been terminated (i.e. lengths of string is zero=0). Lemma 1 is used as induction base.

Lemma1: For all string length, ONR algorithm with QoS assurance optimally determines replicas number of object in the Tree T_v when v is leaf of T , such that ONR cost is minimized.

Proof: There is string with length $l \geq 1$ where ONR cost is the minimum between:

- a) Allocating a replica in node v , the cost is the sum of: **1)** storage cost of replica in node v .
- 2)** Communication cost: this cost is the sum of links distance on the string with length l , since the object is read from the server, which is of length l away from node v , by only one.
- b) Not allocating a replica in node v . Therefore the ONR cost is communication only. This cost equals to sum of the link distance on string with length l multiplied by frequency of reading object by node v .

Note that According to (8), it is clear that in sub-tree T_v , which is $d(v, v_i) > q(v)$, one replica must be located in node v . That is option **a)** is selected. Otherwise, the minimum amount of options **a)** and **b)** is selected.

Lemma2 is the induction step. In this lemma2, we prove that all strings with length l , ONR optimally allocate the replicas of objects in the tree T_v when v is a non-leaf of T .

Lemma2: Assume that the ONR algorithm optimally determines number of replicas to nodes in every tree that is rooted at the child of node v (i.e., $T_{v_1}, T_{v_2}, \dots, T_{v_k}$, see Fig.3), for all string length, such that ONR cost is minimized and QoS for the user should be met. Then, ONR algorithm with QoS assurance determine numbers of replicas in tree rooted at node v optimally for all string length, such that ONR cost is minimized.

Proof: Either one of two possibilities holds:

- 1) If there is one string connected (i.e., $l=1$), ONR cost is the minimum between:
 - a) Allocating a replica in node v . The ONR cost is the sum of: **1)** the cost of sub-trees that are rooted at the children of node v with string length one connected to them. **2)** The storage cost at node v and communication cost which equals to the distance over link (i.e. distance between node v and v_i (see (7))).
 - b) Not allocating a replica in node v . The ONR cost is the sum of: **1)** the costs of k sub-trees that rooted at the children of node v with a

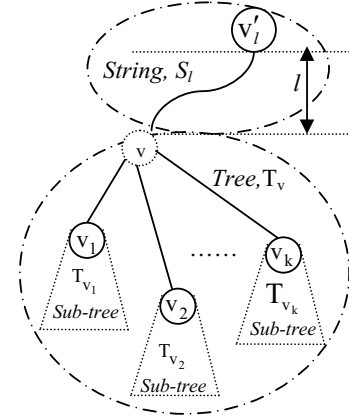


Figure.3. Tree with string S_l

string of length two connected to them Since the distance between the children of v (v_1, v_2, \dots, v_k) and the node v_1 ($l=1$) is equal to two links. **2)** Communication cost, which equals to the sum of links distance on string with length two multiplied by frequency of reading object by node v (see (6)).

If $d(v, v_i) > q(v)$; then option **a)** is selected, only to replicate the object in the node v so that the QoS in node v is satisfied. Otherwise, ($d(v, v_i) \leq q(v)$) these are the only two possibilities when there is one string connected. The minimum is optimal allocation¹ for k sub-trees that rooted at children of node v . If this is not the optimal allocation then clearly one of the allocations for k sub-trees that rooted at node v , is not optimal. This contradicts the assumption that all k sub-trees that are rooted at children of node v have optimal allocation for replicas, for all string length.

2) If a string of length l is connected, ONR cost is the minimum between:

- a) Allocating a replica in node v . The ONR cost is the sum of: **1)** trees is cost that are rooted at childrens of node v with string length one connected to them. **2)** Storage cost at node v .
- 3)** Communication cost, which equals to sum of the links distance on string with length l (see (7)).
- b) Not allocating a replica in node v , and using the replica from the string. The ONR cost is the sum of: **1)** The costs of k sub-trees which are rooted at children of node v , with a string of $l+1$ connected to them. **2)** Communication cost of node v , that equals to the sum of links

¹ That is, determining ONR with QoS assurance, such that ONR cost is minimized.

distance on string with length l multiplied by frequency of reading object by node v (see (6)).

If $d(v, v_i) > q(v)$ then option **a**) is selected. Else, there are the only two possibilities when there is string of length l connected to node v . The minimum is the optimal allocation for k sub-trees rooted at node v . If this is not optimal allocation then clearly one of the allocations for k sub-trees rooted at childrens of node v , is not optimal. This contradicts the assumption that k sub-trees that are rooted at the children of node v have optimal allocation for the replica, for all string length. Thus, the allocation of replicas is optimal for the T_v rooted at root v , for every length of strings connected to it.

Theorem1. When the algorithm terminate, that is $l=0$, optimal number of replicas is determined. So that ONR cost is minimized and the user's QoS is assured.

Proof: The proof is conducted by an induction where lemmal is the base and lemma2 is the induction step.

Finally, we analyse the complexity of ONR algorithm with QoS assurance for a DG Tree in the following theorem.

Theorem2. Let T_r be a DG Tree with n nodes. We can find ONR with QoS assurance for T_r in $O(n.q(v))$ time and space complexity, such that (5) is minimized.

Proof: ONR algorithm computes each $\min_cost(v, v_i)$ and $OLR(v, v_i)$ in $O(Z(v))$, where $Z(v)$ is the number of v 's children. Moreover, it is needed to calculate $O(q(v))$ in the forms of $OLR(v, v_i)$ and $\min_cost(v, v_i)$ for each $v \in V$, where $q(v)$ is the QoS of node v . Therefore, the total time complexity of ONR algorithm is given by:

$$O(\sum_{v \in V} (Z(v).q(v))) \leq O(\sum_{v \in V} Z(v).q(v)) = O(n.q(v))$$

Therefore, time complexity of our algorithm is $O(n.q(v))$. As we noted, our algorithm has a better time complexity than the two other ones. The coefficients and $\log^2 n$ are too large since n the number of nodes in DG, is large.

5. Conclusion

In this paper we have introduced a new algorithm for determining Optimal Number of Replicas (ONR) and Optimal Location of Replicas (OLR) in a DG Tree system, such that the overall cost (i.e., communication and storage cost) is minimized and QoS of users are being met. Meanwhile, our algorithm takes time complexity at worst-case $O(n.q(v))$, which n and $q(v)$

are number of nodes and user's QoS in the DG Tree. Although QoS constraint is applied to both of these two algorithms [13, 14] but the objective function of our algorithm are different to these two algorithms and this is in accordance with the actual properties of DG. On top of that, our algorithm has a better time complexity than the said two algorithms.

References

- [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 Computing Journal 28 (2002), no. 3, 749{771.
- [2] D. Bosio, J. Casey, A. Frohner, and L. Guy et al, Next generation data grid data management services, computing in high-energy physics (CHEP2003), March 2003.
- [3] A. Chervenak et. al, Giggie: A framework for constructing scalable replica location services, Proc. of the ACM/ IEEE Super Computing Conference, November 2002.
- [4] 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.
- [5] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe. Wide area data replication for scientific collaborations. In Proceedings of the 6th International Workshop on Grid Computing, November 2005.
- [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] K. Ranganathan, A. Iamnitchi, and I. Foster. Improving data availability through dynamic model-driven replication in large peer-to-peer 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 376–381, 2002.
- [9] M. M. Bae and B. Bose. Resource placement in tours-based networks. IEEE Transactions on Computers, 46(10):1083– 1092, October 1997.
- [10] 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.
- [11] 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.
- [12] J. H. Abawajy. Placement of file replicas in data grid environments. In ICCS 2004, Lecture Notes in Computer Science3038, pages 66–73, 2004.
- [13]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.
- [14] V. Rehn-Sonigo, Optimal Replica Placement in Tree Networks with QoS and Bandwidth Constraints and the Closest Allocation Policy, Proceedings of the CoreGRID Symposium 2007.
- [15] World wide LHC computing Grid .http://lcg.web.cern.ch/lcg/.
- [16] The GriPhyN Project, http://www.Griphyn.org.