Cost Optimization of Replicas in Tree Network of Data Grid with QoS and Bandwidth Constraints

Alireza Chamkoori
Department of Computer Engineering
Khormoj Branch
Islamic Azad University, Iran

Farnoosh Heidari
Department of Electrical Engineering
Bushehr Branch
Islamic Azad University, Bushehr, Iran

Naser Parhizgar
Department of Electrical Engineering
Shiraz Branch
Islamic Azad University, Shiraz, Iran

***Abstract*—Data Grid provides resources for data-intensive scientific applications that need to access a huge amount of data around the world. Since data grid is built on a wide-area network, its latency prohibits efficient access to data. This latency can be decreased by data replication in the vicinity of users who request data. Data replication can also improve data availability and decreases network bandwidth usage. It can be influenced by two imperative constraints: Quality of Service (QoS) that is locally owned by a user and bandwidth constraint that globally affects on link that might be shared by multiple users. Guaranteeing both constraints and also minimizing replication cost consisting communication and storage costs is a challenging task. To address this problem, the authors propose to use a dynamic algorithm called Optimal Placement of Replicas to minimize replication cost and coupled with meeting both mentioned constraints. It is also designed as heuristic algorithms that are competitive with optimal algorithm in performance metrics such as replication cost, network bandwidth usage and data availability. Extensive simulations show that the Optimal algorithm saves 10% cost compared to heuristic algorithms and provides local responsiveness for half of the user requests.**

Keywords*—Hierarchical data grid; replication cost; replica optimal placement; communication cost; storage cost; cost minimization; QoS and bandwidth constraints

I. Introduction

Data-intensive scientific applications are increasingly growing because of recent development in distributed systems (such as peer to peer systems, grid, cloud computing, etc.). These applications need extreme-scale repositories and computing resources. For example, astronomy projects-Virtual Observations[1] and protein simulation-Bio-Grid[2] require analysing a huge amount of data. The data generated from such an experiment, with a network of sensors or an instrument is stored at a master storage site and is moved to other sites all over the world. *Data grid* is a suitable distributed system to provide computational facilities and repositories in large scale for users. It offers a scalable infrastructure for management of massive storage resources and data that are distributed across network.

Each data grid has its own model that is the manner of resource organization such as data resources which can be either single or distributed, data size, and sharing model of data. Data grids usually follow four common models: *monadic*, *hierarchical*, *federation*, and *hybrid* [1]. In this paper, it has been focused on hierarchical model that consists of several levels (tiers). In the first level, data is generated and stored. Then, the data is distributed to other levels if it is requested. This model is usually found in current data grids [1]. One example for this model is LCG (Large Hadron Collidor Grid) project in which scientists likely need to access a huge amount of raw data that can reach several petabytes. Also, most of these data are read only; because they are the input data to the application for analysis, classification, and other purposes.

Data grid, building on wide-area network and resulting in high latency as its consequence, utilizes efficient techniques to deliver data to users with guaranteed QoS that has either local or global influence on user satisfaction. The local influence is due to QoS that is requested by user and the global influence is because of constraint on the link bandwidth may be shared by multiple users. One of the technique to guarantee QoS is data replication in multiple locations which allows user to access data from a server in her vicinity.

Clearly, in one hand, data replication not only reduces data access cost and provides the guaranteed QoS, but also promotes data availability in many applications. On the other hand, replication cost borne by user can increase if a suitable replication strategy is not taken. Thus, replication cost and access cost are two potentially conflicting objectives that should be addressed whilst the constraints in the system are satisfied.

This paper discusses the aim of addressing the above problem called *Replica Placement Optimization* with QoS and bandwidth constraints. Here some assumptions in this problem, which are compatible with the characteristics of real data grid are also discussed. It is assumed that all objects are initially stored in the root of tree and access to the objects is based on the *Closest* policy in which the user requests are served only by the closest replica in the path from request node up to the root [2]. The objective of this problem is to minimize communication and storage costs whilst QoS constraint (number of hops) and link bandwidth limitation are respected.

Several works investigate replica placement on parallel and distributed systems with regular structures such as tree, hyper cube, ring, etc. [3]. But, neither this deals with QoS guarantee nor with bandwidth constraints. Moreover, the objective function of these work is not similar to that of us. Cidon et al. [4] studied an optimal placement of replicas and minimized communication and storage costs without any constraints.

[1] www.birncommunity.org
[2] www.biogrid.jp

Karlsson et al. [5] have taken into account bandwidth limitations and proposed several heuristic algorithms to tackle NP-complete problem. But, they do not consider QoS constraint. Wu et al. [6] investigated a problem in which the objective function is to minimize the number of replicas with QoS in terms of a range limit (i.e., hops number). Rehn-Sonigo [7] proposed a major extension [6], with the same objective function but with an additional constraint, i.e., link bandwidth. Our work is different with both of these previous works in terms of the objective function.

The main objective of our algorithms is to minimize replication cost, which includes communication and storage costs whilst the desired QoS of user in terms of distance between the client (i.e., user) and server is guaranteed. It is also considered bandwidth constraint as a global QoS that belongs to the network and can influence on all clients in the data grid system. Further, proposed several heuristic algorithms and compared with the optimal one in three aspects that are important in data grid: *replication cost*, *network bandwidth usage*, and *data availability*.

The rest of the paper organized as: Section II presents related work in replica placement in data grid environments. The basic preliminaries are provided and also formulated replica placement optimization problem in Section III. Section IV dedicated to proposed algorithms. The experimental results are given in Section V. Finally, Section VI concludes the paper.

II. Related Work

Replica placement problem has been well investigated in literature. This problem can be categorized based on network structure: *general graph* and *tree*. The problem in former category is known to be a NP-hard K-center problem and has been dealt in two steps discussed as: In first step, with considering the desired objective function, a spanning tree is extracted. In the second step, the replicas are placed in the extracted tree to optimize the objective function. In this regard, many works considered QoS as a constraint whilst considering replication cost as an objective function [8], [9]. Our work does not fall into this line of research.

In the latter structure, this problem either can be reduced to NP-problem or can be optimally solved, which depends on the defined objective function and constraints in these problems. Researchers have exhaustively studied this problem mainly for two categories as follows:

Replica Placement without Constraint: Early work on this category has done by Wolfson and Milo [10], where the multicast write policy is employed to optimally allocate k replicas in different topology. We also propsed algorithm to place k replicas in data grid tree network [11], [12]. Kalpakis et al. [13] discussed the replica placement problem when a general objective function takes into account read, write, and storage costs and uses a minimum spanning tree to propagate updates among the replicas. Cidon et al. [4] studied an instance of the problem with the objective function aimed at finding the optimal number of replicas, where the communication and storage costs are considered. Also a similar instance was studied of the problem and an algorithm with the lower time complexity in the context of data grid systems [14].

Replica Placement with Constraints: The constraints in replica placement can be on server capacity, link bandwidth, and QoS [2]. Tang et al. [15] have been the first author to consider actual QoS constraint in replica placement problem. They studied this problem in graph and tree networks. Several works considered this constraint in data grid tree network where the objective function is either replication cost minimization or load balance on servers [6], [16]. Also, Shorfuzzaman et al. [17] investigated the placement of k replicas with QoS constraint in data grid tree network such that the replication cost, which is a summation of write, read, and storage costs is minimized. Rehn-Sonigo [7] proposed a theoretical algorithm to balance load on servers such that QoS and link bandwidth constraints are satisfied.

Most work listed above do not provide both QoS and bandwidth constraints simultaneously. A dynamic algorithm is proposed to optimize replication cost including communication and storage costs such that QoS and bandwidth constraints are guaranteed. Although [7] and our work are similar in constraints (QoS and link bandwidth), our objective function is to minimize communication and storage costs whilst the objective function of [7] is aimed at balancing load on servers. Moreover, three heuristic algorithms is proposed to replicate data based on either read cost or storage cost.

III. Premilinaries and Problem Formulation

The system model is represented as a rooted undirected tree $T_r = (V, E)$, where $V(|V| = n)$ is the set of nodes. A node is a server if it has a replica of the object else it is a client. E is the set of edges, which represents links in tree. r is the root of tree and all objects i $m(1 \leq i \leq m)$ are initially stored in it. The size of each object i is denoted by O_i. Let $r_i(v)$ and $S(v)$ be two functions representing the number of read requests issued by node v for object i and storage cost of placing a replica of object at node v, respectively. Also, let $d(u, v)$ be a non-negative cost between nodes u and v and assigned to link $(u, v) \in E$. It can be interpreted as delay, link cost or number of hops. Also each link $(u, v) \in E$ owns a bandwidth limit $bw(u, v)$ that cannot be exceeded. Let $t(v)$ the sub-tree rooted by node v, and $t'(v) = t(v) - v$, i.e. the forest of trees rooted at v's children.

It is defined in two constraints: QoS and link bandwidth. The required QoS is termed by $q(v)$ that should be guaranteed. With no less of generality, it can be considered the distance (i.e., the number of communication hops) between a client and server as QoS. Thus,

$$\forall c \in clients, \forall v \in servers, d(c, v) \leq q(c) \qquad (1)$$

where distance $d(c, v)$ is the number of hops between client c and server v. According to (1) if object i is retrieved by client c from server v within distance $d(c, v) \leq q(c)$, then QoS requirement is met; otherwise it is violated.

Other constraint in our system model is bandwidth limitation and it is defined as follows. Assume that the node v_l in distance of l links is an ancestor of node c and also $S_r(t(c), v_l)$ denotes the total requests issued from $t(c)$ to node v_l for the desired object. Based on this constraint, $S_r(t(c), v_l)$ should not

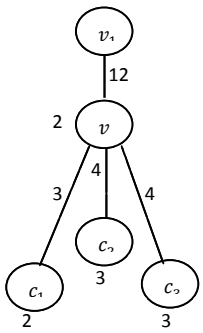


Fig. 1. An example of bandwidth constraint.

exceed the bandwidth of all links between node c and v_l. This translates into:

$$\forall c \in clients, \forall v \in servers, S_r(t(c), v_l) \leq bw(c, v_l) \quad (2)$$

where $l \in path[c, v_l]$.

An example of bandwidth constraint is illustrated in Fig. 1. Consider tree $t(v_1)$ that holds the desired object in its root (i.e, v_1). The number attached to node represents the requests per time unit of that node for the desired object. For example, node c_1 owns 2 requests for object in node v_1. Label on the link shows its bandwidth. For example, link bandwidth between v and v_1 is 12. Suppose sub-tree rooted in v making 10 (2+3+3+2) requests must be passed through the link between v and v_1 to reach object at v_1. In this case, there is not bandwidth restriction because $S_r(t(v), v_1) = 10 \leq bw(v, v_1) = 12$. Another example: Let us increase the requests of node v from 2 to 6. In this case, since $S_r(t(v), v_1)$ exceed from 10 to 14, the link bandwidth constraint is violated because $S_r(t(v), v_1) = 14 > bw(v, v_1) = 12$. A simple solution to remove this limitation is to retrieve a replica of object from v_1 and store it in node v.

In the following, is calculated the total cost and defined the replica placement optimization problem. Consider a resident set RP_i hosting replicas of object i for tree T_r. To minimize the total cost of this set, $Cost(RP_i, T)$, includes two types of costs: *communication* and *storage* cost.

A read cost of RP_i is the cost of serving all read requests issued from $v \in V$ and defined as

$$\sum_{v \in V} r_i(v) \times O_i \times d(v, v_l) \quad (3)$$

where $v_l \in RP_i$ is the closest ancestor of node v that contains object i and $d(v, v_l)$ equals the hops number on l links.

The storage cost of RP_i is the cost of hosting object i at all nodes in RP_i and formalized as

$$\sum_{v \in RP_i} O_i \times S(v) \quad (4)$$

Thus, the total cost for the set RP_i for tree T is

$$Cost(RP_i, T) = \sum_{v \in V} r_i(v) \times O_i \times d(v, v_l) + \sum_{v \in RP_i} O_i \times S(v) \quad (5)$$

Replica Placement Optimization Problem: Given a tree network $T_r(V, E)$, with m objects, find a subset $RP_i \subseteq V$ for all objects i ($1 \leq i \leq m$) such that the total cost $Cost(RP_i, T)$ given in (5) is minimized and constraints in (1) and (2) are satisfied.

IV. Proposed Algorithms

In this section, an algorithm called Replica Placement Optimization is suggested that works in two phases in order to solve the optimization problem described in the previous section. In the first phase, the cost parameter is computed for each node and then determined whether it is potential to host a replica in itself or not. The computed cost parameter will serve phase 2 to determine the optimal placement of replicas. In the following, two phases are presented in details.

A. Phase 1: Bottom-Up Cost calculation

Let $C^i_{min}(t(v), v_l)$ denote the minimum cost (calculated based on (5)) of the sub-tree $t(v)$ with the assumption that the issued requests from $t(v)$ are processed by node v_l that is the lowest ancestor of node v. Also assume that the *candidate nodes set* associated to $C^i_{min}(t(v), v_l)$ is termed by $C^i_{set}(t(v), v_l)$. This set contains potential nodes that can host a replica of object i. By traversing the tree T_r in breadth first order from bottom to top, $C^i_{min}(.)$ and $C^i_{set}(.)$ for two distinguished cases are calculated as follows:

Case (a): As shown in Fig. 2, assume that node v is a leaf. It has two ways to access object i in v_l, which is its lowest ancestor. 1) It reads object i for one time and locally stores it. Thus, node v incurs the read cost from v_l through link (v, v_l) for one time in addition to the storage cost of object i. This cost is called *storing cost* and is computed as $C^i_{sl}(v, v_l) = O_i \times d(v, v_l) + O_i \times S(v)$ and node v is added to $C^i_{set}(v, v_l)$. Since a request from v to v_l is passed through links $l \in path[v, v_l]$ to read the desired object for one time, the value $S_r(v, v_l)$ is increased by one. This modification helps us to control link bandwidth constraint. 2) It reads the object whenever it needs, and incurs read cost for $r_{v,i}$ times. This cost is called *reading cost* and is computed as $C^i_{rl} = r_{v,i} \times O_i \times d(v, v_l)$. So, node v is not added to $C^i_{set}(v, v_l)$ and $S_r(v, v_l)$ is increased by $r_{v,i}$ because all requests $r_{v,i}$ passed through links $l \in path[v, v_l]$. By considering the defined QoS for each node and the link bandwidth, $C^i_{min}(.)$ for each leaf is calculated as follows.

If the distance between v and its ancestor v_l is more than $q(v)$ (i.e., $d(v, v_l) > q(v)$) or the issued requests $r_{v,i}$ exceeds than bandwidth of any links $l \in path[v, v_l]$, then the object i is retrieved from node v_l and replicated at node v (in fact, node v has only one way to access object in v_l). Thus, $C^i_{min}(v, v_l) = C^i_{sl}(v, v_l)$ and the remaining parameters, $C^i_{set}(v, v_l)$ and $S_r(v, v_l)$, corresponds to the parameters that are defined in the *storing cost*. Otherwise, *reading cost* and *storing cost* are calculated for node v and then the minimum cost is considered. In fact, in this case, node v has two ways to access its desired object. Thus, $C^i_{min}(v, v_l) = min(C^i_{sl}(v, v_l), C^i_{rl}(v, v_l))$, and based on the minimum cost, other parameters, $C^i_{set}(v, v_l)$ and $S_r(v, v_l)$, are calculated, respectively.

Case(b): As illustrated in Fig. 3, node v is a non-leaf. Similar to leaf v, it is with two alternatives to access object i in

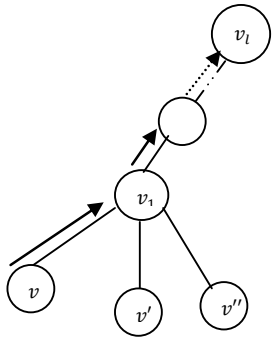


Fig. 2. The rational of dynamic algorithm for leaf nodes of tree.

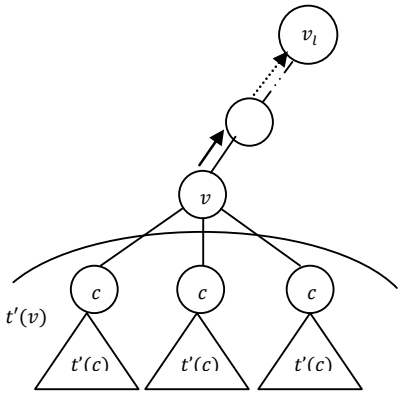


Fig. 3. The rational of dynamic algorithm for non-leaf nodes of tree.

node v_l. 1) Node v reads object i from v_l and stores in itself. Thus, *storing cost* is calculated as described for leaves. In contrast to leaf, we have $t'(v) \neq \emptyset$ and replication cost of $t'(v)$ is the minimum replication cost for children $c \in t'(v)$ with this assumption that node v contains replica of object i. Therefore, *storing cost* for non-leaf is calculated as $C^i_s(v, v_l) = O_i \times (d(v, v_l)+S(v))+\sum_{c\in t'(v)}(C^i_{min}(c, v))$. Similar to the *storing cost* incurred for leaf, $C^i_{set}(v, v_l)$ contains node v and $S_r(v, v_l)$ is increased by one. Thus: $C^i_{set}(v, v_l) = \cup_{c\in t'(v)} C^i_{set}(c, v) \cup v$ and $S_r(v, v_l) = S_r(v, v_l) + 1$, where $\cup_{c\in t'(v)} C^i_{set}(c, v)$ indicates the optimal candidate nodes set of $t'(v)$ if v contains replica of object i. 2) Node v reads object from v_l instead of storing in itself and its cost equals to *reading cost* described for a leaf. Since, in this case, node v does not contain replica of object i, the replication cost of $t'(v)$ is the minimum replication cost of children $c \in t'(v)$ with the assumption that v_l has a replica. Therefore, reading cost for non-leaf is $C^i_r(v, v_l) = r_{v,i} \times O_i \times d(v, v_l) + \sum_{c\in t'(v)}(C^i_{min}(c, v_l))$, and its corresponding parameters $C^i_{set}(v, v_l)$ and $S_r(v, v_l)$ are given by $C^i_{set}(v, v_l) = \cup_{c\in t'(v)} C^i_{set}(c, v_l)$ and $S_r(v, v_l) = S_r(v, v_l) + r_{v,i}$ (see how these parameters are calculated for a leaf that incurs *reading cost*).

Thus, based on the defined constraints for nodes and links, $C_{min}(v, v_l)$ for non-leaf nodes is calculated as follows. If one of constraints is violated ($q_v < d(v, v_l)$ or $bw(v, v_l) < r_{v,i}$), then v has to read object and replicate in itself, and $C_{min}(v, v_l) = C_s(v, v_l)$. If $d(v, v_l) \leq q(v)$, $C_{min}(v, v_l)$ is the minimum of *storing* and *reading* costs. If ($C_{min}(v, v_l) = C_s(v, v_l)$), it is not required to investigate the link constraint because in storing cost case, only one request goes up through links $l \in path(v, v_l)$ that can process at least one request. Otherwise, if ($C_{min}(v, v_l) = C_r(v, v_l)$), it is possible the link constraint violation happens. To handle this violation, in the following subsection, an algorithm called Bandwidth Constraint Handling with Minimal Cost (BCHMC) and also three heuristic algorithms are suggested. Based on the above discussions, Algorithm 1 details Phase 1.

Algorithm 1: Cost Calculation: Phase 1

Input : T_r
Output: $C_{min}(v, v_l)$, $C_{set}(v, v_l), S_r(v, v_l)$

forall $v \in T_r$ *traversed in breadth first order* **do**
  if v *is a leaf* **then**
    if $q_v < d(v, v_l)$ *or* $bw(v, v_l) < r_{v,i}$ **then**
      $S_r(v, v_l) + +$, $C_{min}(v, v_l) \leftarrow C_{sl}(v, v_l)$
      $C_{set}(v, v_l) \leftarrow v$
    else
      Calculate $C_{min} \leftarrow min(C_{sl}(.), C_{rl}(.))$
      if $C_{min}(.) = C_{sl}(.)$ **then**
        $S_r(v, v_l) + +$, $C_{set}(v, v_l) \leftarrow v$
      else
        $S_r(v, v_l) = S_r(v, v_l) + r_{v,i}$
      end
    end
  else
    if $q(v) \leq d(v, v_l)$ *or* $bw(v, v_l) < r_{v,i}$ **then**
      $S_r(v, v_l) + +$, $C_{min}(v, v_l) \leftarrow C_s(v, v_l)$
      $C_{set}(v, v_l) \leftarrow \cup_{c\in child(v)}(c, v) \cup v$
    end
    if $(d(v, v_l) \leq q(v))$ **then**
      Calculate $C_{min} \leftarrow min(C_s(.), C_r(.))$
      if $C_{min}(.) = C_s(.)$ **then**
        $S_r(v, v_l) + +$,
        $C_{set}(v, v_l) \leftarrow \cup_{c\in child(v)}(c, v) \cup v$
      else
        $S_r(v, v_l) = \sum_{c\in child(v)} S_r(c, v_l) + r_{v,i}$
        if $bw(v, v_l) < S_r(v, v_l)$ **then**
          call Bandwidth Constraint Handling Algorithm
        else
          $C_{set}(v, v_l) \leftarrow \cup_{c\in child(v)}(c, v)$
        end
      end
    end
  end
end
Return $C_{min}(v, v_l)$, $C_{set}(v, v_l), S_r(v, v_l)$

B. Bandwidth Constraint Handling

In this section, at first introduced new terminologies and then propose an algorithm to solve bandwidth constraint violation for a non-leaf node v. For clarity in notations, node v is termed by v_f. As discussed above, bandwidth constraint happens when node v_f reads object from v_l and and its requests $S_r(v_f, v_l) = r_{v_f,i}+\sum_{c\in child(v_f)}(r_{c,i})$ increases more than the bandwidth of at least one of the links $l \in path[v_f, v_l]$.

Algorithm 2: Bandwidth Constraint Handling with Minimal Cost (BCHMC)

Input : T_r, $S_r(v, v_l)$
Output: $S_r(v, v_l), C_{min}(v, v_l)$, $C_{set}(v, v_l)$

1 **forall** $v \in R$ **do**
2 | $C_{over}(v) = C_s(v, v_l) - C_r(v, v_l)$
3 **end**
4 $R_{sort} \leftarrow$ sort node $v \in R$ based on $C_{over}(v)$
5 **while** $S_r(v_f, v_l) \leq bw(v_f, v_l)$ **do**
6 | select $v \in R_{sort}$
7 | **if** $(v = v_f)$ **then**
8 | | $S_r(v_f, v_l) = 1$
9 | | $C_{min}(v_f, v_l) = C_s(v_f, v_l)$
10 | | $C_{set}(v_f, v_l) = \cup_{c \in child(v)} C^i_{set}(c, v_f) \cup v_f$
11 | **else**
12 | | $S_r(v_f, v_l) = S_r(v_f, v_l) - (r_{v,i} - 1)$
13 | | $C_{min}(v, v_l) = C_s(v, v_l)$
14 | | $C_{set}(v, v_l) = \cup_{c \in child(v)} C^i_{set}(c, v) \cup v$
15 | **end**
16 **end**
17 Return $C_{min}(v, v_l)$, $C_{set}(v, v_l)$

Algorithm 3: Replica placement: Phase 2

Input : $C_{min}(.), C_{set}(.)$
Output: RP

1 RP$\leftarrow \emptyset$, $lev \leftarrow 1$, $v_{inv} \leftarrow c \in child(r)$
2 Procedure Replica Placement (v_{inv}, lev)
3 **forall** $v \in T_r$ *traversed in level order* **do**
4 | **if** $v_{inv} = Null$ **then**
5 | | Return;
6 | **else**
7 | | **if** $v_{inv} \in C_{set}(v_{inv}, v_{lev})$ *or* $C_{min}(v_{inv}, v_{lev}) = C_s(v_{inv}, v_{lev})$ *or* $C_{min}(v_{inv}, v_{lev}) = C_{sl}(v_{inv}, v_{lev})$ **then**
8 | | | RP$\leftarrow$RP$\cup v$
9 | | | Replica Placement($c \in child(v_{inv})$,1)
10 | | **end**
11 | | **if** $C_{min}(v_{inv}, v_{lev}) = C_r(v_{inv}, v_{lev})$ *or* $C_{min}(v_{inv}, v_{lev}) = C_{rl}(v_{inv}, v_{lev})$ **then**
12 | | | Replica Placement(RP)
13 | | **end**
14 | **end**
15 **end**
16 Return $S_r(v, v_l), C_{min}(v, v_l)$, $C_{set}(v, v_l)$

Note $S_r(v_f, v_l)$ is the summation of requests of node v_f (i.e. $r_{v_f,i}$) and requests of its children that do not contain the replica of object (i.e. $\sum_{c \in child(v_f)}(r_{c,i})$). Let R be a set that includes these nodes and defined as $R = v_f \cup c \in childe(v_f) \wedge c \notin C_{set}(v_f, v_l)$. Also, let $C_{ove}(v)$ be the *overhead cost* of replication for node $v \in R$, which is the difference between *storing cost* ($C_s(v, v_l)$) and *reading cost* ($C_r(v, v_l)$) of node v (i.e., $C_{ove}(v) = C_s(v, v_l) - C_r(v, v_l)$). To eliminate link bandwidth constraint, the BCHMC algorithm is proposed to select nodes from R to host a replica of object i. The rationale behind this algorithm is to select nodes $v \in R$ such that the summation of the overhead cost (i.e., $\sum_{v \in R}(C_{over}(v))$) is minimized and the constraint $S_r(v, v_l) \leq bw(v, v_l), l \in path[v, v_l]$ is satisfied.

To do so, as illustrated in Algorithm 2, overhead cost for all nodes $v \in R$ is first calculated and then these nodes are sorted based on $C_{over}(v)$ on ascending order and denoted by R_{sort}(Lines 1-4). To remove bandwidth constraint, nodes $v \in R_{sort}$ are selected until the condition $S_r(v_f, v_l) \leq bw(v_f, v_l)$ is satisfied for all links $l \in path[v_f, v_l]$. Clearly, if $v = v_f (v \in R)$, then a replica placed at v_f and the already selected nodes (i.e., $v \in R_{sor} \wedge v \neq v_f$) are not considered because by creating a replica in v_f, all requests made by v_f and its children $c \in child(v_f) \wedge c \notin C^i_{set}(v_f, v_l)$ are satisfied with this replica. Thus: $S_r(v_f, v_l) = 1$, $C_{min}(v_f, v_l) = C_s(v_f, v_l)$ and $C_{set}(v_f, v_l) = \cup_{c \in child(v_f)} C^i_{set}(c, v_f) \cup v_f$, where $c \in child(v_f) \wedge c \notin C^i_{set}(v_f, v_l)$ (Lines 5-10). Otherwise, if $v \neq v_f$[3], then a replica should be placed at v. As a result, $S_r(v_f, v_l) = S_r(v_f, v_l) - (r_{v,i} - 1)$ because at least one request from the selected node v goes up through links $l \in path(v, v_l)$, $C_{min}(v, v_l) = C_s(v, v_l)$ and $C_{set}(v, v_l) = \cup_{c \in child(v)} C^i_{set}(c, v) \cup v$ (Lines 11-14).

[3]Note in this condition, node v is a child of v_f that does not contain the replica of object i

C. Phase 2: Top-Down Replica Placement

Phase 2: as illustrated in Algorithm 3, it is a recursive approach that is fed by $C_{min}(.)$ and $C_{set}(.)$ computed in Phase 1. In this phase, an algorithm called *Replica Placement* is suggested to determine which node $v \in T_r$ contains a replica of the object. This algorithm begins at the root of tree and ends at leaves. In the proposed algorithm, assumed that lev is the distance between node v and node v_{inv}. Here, node v has a replica of the object and node v_{inv} is the node that should be investigated whether to host a replica or not, such that (5) is minimized and constraints are satisfied. Sine *Replica Placement* Algorithm starts from root r, we set $lev = 1$, $v = r$ and $v_{inv} = c \in child(r)$.

By starting from the right most child of r, if node $v_{inv} \in C_{set}(v_{inv}, v_{lev})$, then a replica is placed at v_{inv} and it is added to RP, i.e., the set of optimal placement of replicas. Also *Replica Placement* Algorithm is called with $lev = 1$, and $v_{inv} = c \in child(v_{inv})$. That is, Replica Placement ($child(v_{inv}), 1$) (lines 7-10). Otherwise, if $v_{inv} \notin C_{set}(v_{inv}, v_{lev})$, algorithm investigates the value of $C_{min}(v_{inv}, v_{lev})$ and based on this value two cases are considered: 1) If ($C_{min}(v_{inv}, v_{lev}) = C_r(v_{inv}, v_{lev})$), then node v_{inv} does not host a replica and the *Replica Placement Algorithm* is called with $lev = lev + 1$ and $v_{inv} = c \in child(v_{inv})$(lines 11-13). 2) Otherwise, if ($C_{min}(v_{inv}, v_{lev}) = C_s(v_{inv}, v_{lev})$), Replica Placement Algorithm works similar to the case as discussed above where $v_{inv} \in C_{set}(v_{inv}, v_{lev})$.

The time complexity of the algorithm to find an optimal placement of replicas is as follows. The algorithm works in two phases. In Phase 1, for each node $v \in T_r$, the values of $C_{min}(.)$ and $C_{set}(.)$ are calculated. So, the computation requires $O(n)$ if bandwidth constraint is not violated. If this

constraint happens (Lines 27-28), then Algorithm 2 is called and takes time complexity of $O(n)$. Thus, in worst case, the time complexity of Phase 1 is $O(n^2)$. In regard to Phase 2, for all nodes, the computed parameters in Phase 1 are compared and optimal placement of replicas is determined. Thus, this phase takes $O(n)$, and the total computation complexity of proposed algorithm is $O(n^2 + n) = O(n^2)$.

D. Heuristic Algorithms for handling Bandwidth Constraint

In this section, the authors propose three algorithms to address bandwidth constraint and then compare with Algorithm 2 in performance parameters in the next section. These algorithms are as follows:

Bandwidth Constraint Handling Based On Nodes Requests (**BCHNR**): Whenever link bandwidth constraint occurs for node v_f, it is handled as follows: First, all children of v_f not containing replica of object (i.e., $c \in childe(v_f) \wedge c \notin C_{set}(v_f, v_l)$) and node v_f are sorted based on their requests in descending order. Second, the first node in the sorted nodes list is selected and a replica of object is placed at this node. The node selection process is repeated until the bandwidth constraint is removed (i.e., $S_r(v_f, v_l) \geq bw(v_f, v_l)$).

Bandwidth Constraint Handling Based on Nodes Storage Cost (**BCHNSC**): When link bandwidth constraint happens for node v_f, this algorithm works as follows. First the storage cost of replication in all children of node v_f is calculated and then the children of v_f are sorted in ascending order of storage cost. Second, the nodes corresponding to these sorted values are chosen to host replica of the object. The node selection (to host replica) continues until the constraint $S_r(v_f, v_l) \geq bw(v_f, v_l)$ is removed.

Bandwidth Constraint Handling Based on Node Random Selection (**BCHNRS**): Similar to the above discussed heuristic algorithms, whenever the link bandwidth constraint is violated for node v_f, the children of v_f not containing replica are randomly selected to host object replica until the violation is omitted.

Bandwidth Constraint Handling Based on Node v_f (**BCHNV**): In this simple algorithm, if link bandwidth constraint happens for node v_f, then the desired object is replicated in node v_f; as a result the constraint is discarded.

V. Performance Evaluation

Extensive experiments have been done to evaluate optimal placement of replica Algorithm with QoS and bandwidth constraints, using different proposed algorithms that handle bandwidth constraint, with several criterion such as replication cost, network bandwidth usage and local availability of objects.

A. Simulation Setup

In this simulation, the tree is randomly generated and controlled by two parameters: number of nodes and the children of each node that effects the proposed algorithms that handle the bandwidth constraint. The number of nodes n ranges from 100 to 5000 and the number children of each node follows a uniform distribution in (1-5) for $n \leq 1000$ and [1-10] for $n > 1000$. It is set q with a fraction of the tree height h. That is, $q = 1/4h$, $q = 1/2h$, $q = 3/4h$ and $q = h + 1$. The last value of q implies the absence of QoS. The link bandwidth constraint is assigned as follows. Tree links in level 0 connecting a leaf to its father is assigned with a uniform distribution in the range (1, 10), and the link bandwidth of an immediate higher level of that leaf ($level1$) is set to (10, 30) for $n \leq 1000$ and (30-70) for $n > 1000$. Each higher levels links ($level \geq 2$) is assigned between 2 and 4 times the immediate corresponding lower level links. The value of other parameters following a uniform distribution is according to Table I.

TABLE I. Default System Parameter Settings

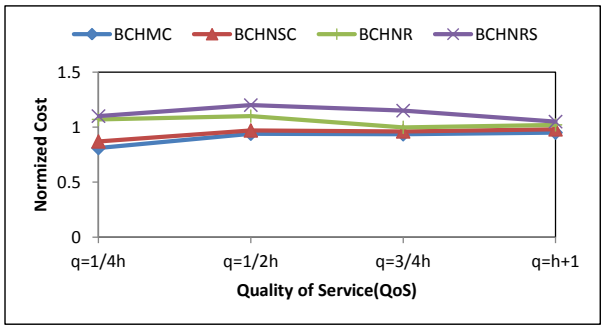
Parameter	Setting	Parameter	Setting
n	1-5000	$r_{i,v}$	1-10
m	100	$S(v)$	1-20
O_i	1MB-100MB	$d(u, v)$	1-5

B. Results

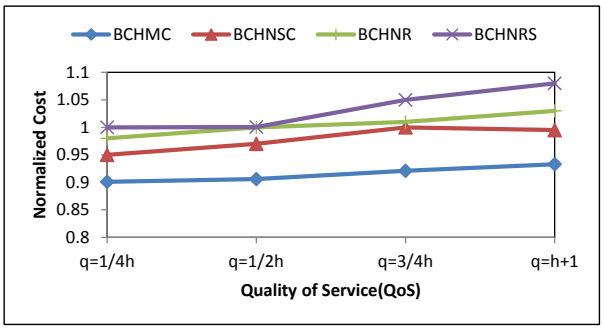
1) Normalized Replication Cost: It indicates the ratio of replication cost of the proposed algorithms to the **BCHNV** Algorithm as a benchmark. If this ratio is lower, the algorithm works better in finding a placement for the replicas in the tree network. From Fig. 4, observed that the BCHMC Algorithm has minimal normalized replication cost for all q and n values compared with other algorithms. Three observations we can make are as follows: 1) As n increases the normalized replication cost of the BCHMC algorithm decreases. The reason is that when tree size (i.e., n) increases, QoS and link bandwidth constraint violations happen more and the proposed optimal algorithm works better than other algorithms. As an example, for $q = 1/2$, the normalized cost replication decreases from 95% to below 90% when n increases from 100 to 5000. 2) As the requested QoS of nodes is tight (that is q is low), the normalized cost of replication in all proposed algorithms is small in comparison with the case in which QoS tend to be relaxed. This is because, in former case the QoS violation occurs more than the latter case. 3) As expected, the hierarchy between other proposed algorithms in normalized replication cost is respected, i.e., BCHNSC is better than BCHNR which in turn is better than BCHNRS especially n increases. The reason is that the BCHNSC algorithm works based on the storage cost whilst the other two algorithms act based on requests number.

2) Effective Network Usage: It is the ratio of the total data transferred through links to the total requested data for serving user requests. As this parameter decreases, the algorithm performs better in placing replicas in the tree. As shown in Fig. 5, data transferred below 50% through the links in the BCHMC algorithm for all n and q whilst this value for other heuristic algorithms is between 55% and 78%. BCHNR comes after BCHMC such that less than 60% of data is remotely read. In fact, it performs better than other heuristic algorithms. The reason is that the priority of this algorithm is to store replicas in nodes that have more requests. As results show, in regards to remaining heuristic algorithms therefore BCHNSC works better than BCHNRS, which is in turn better than BCHNV. Also, from Fig. 5 It is found that as q decreases, the network usage percentage improves because of more QoS violations, that results in more replicas being placed in the tree.

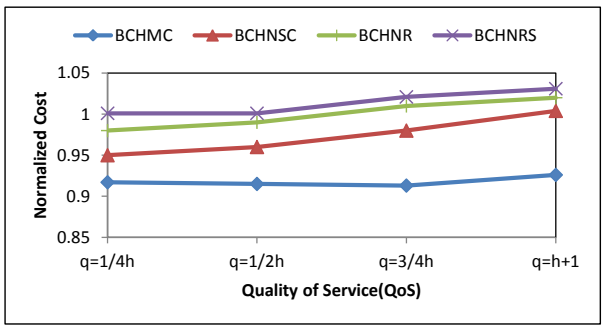
3) Local Access Percentage of Objects: It exhibits the percentages of user requests in the tree that are locally satisfied. As shown in Table II, by using BCHMC algorithm, more than



(a)

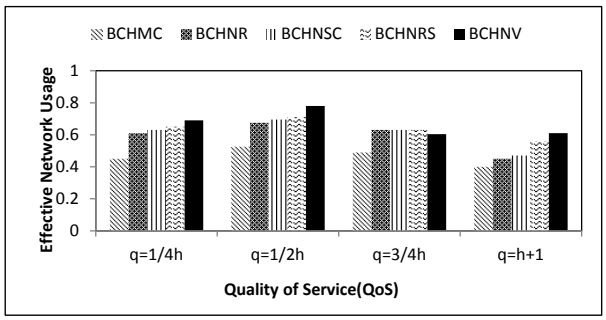


(b)

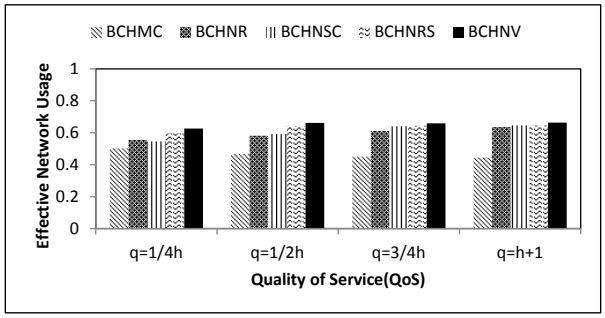


(c)

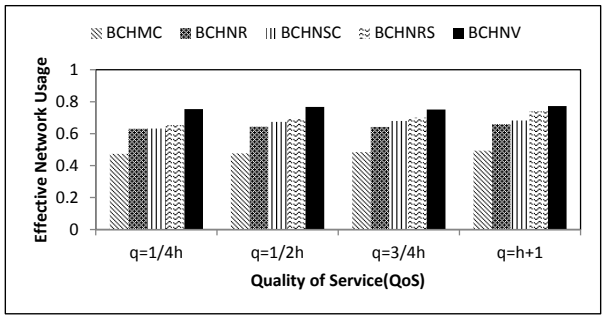
Fig. 4. Normalized Cost vs. Quality of service(QoS). (a) nodes number=100. (b) nodes number=1000. (c) nodes number=5000.



(a)



(b)



(c)

Fig. 5. Network Usage vs. Quality of service(QoS).(a) nodes number=100. (b) nodes number=1000. (c) nodes number=5000.

50% of requests are locally served and it always outperforms other proposed algorithms for all n and q. But the BCHNV algorithm has the worst performance because in this algorithm the link bandwidth constraint is removed by replication of data in the node v_f instead of its children. As a result, more than 70% of requests remotely access objects. It is also observed the BCHNR algorithm comes after BCHMC with regards to local responsiveness, where BCHNR services 35%-40% of requests locally. The reason is that, to remove bandwidth constraint, BCHNR selects nodes based on their read rate to replicates objects in those nodes. The other algorithms, BCHNSC and BCHNRS, rank next in this metric, respectively.

VI. Conclusions and Future Works

This paper studied a new replica placement algorithm in hierarchical data grid, that amid at replication cost optimization whilst QoS and bandwidth constraints take into consideration. This algorithm has low time complexity which makes it well-suited for data grid environment. The simulation showed that this algorithm compared with heuristic algorithms has suitable performance in terms of access cost, network bandwidth usage and data availability. The BCHMC algorithm replicates data in nodes with guaranteed constraints such that it saves cost by at least 10% compared to other heuristic algorithms. It also locally serve 50% of requests whereas at most only 40% of requests could be locally satisfied in other algorithms. As a future work, we plan to consider updating cost in the proposed algorithms and evaluate the effects of this cost on the performance criterion that are important in data grid environment.

Acknowledgment

The authors would like to thank Deepak C Poola for his helpful suggestions to improve our paper. This work has been done when the second author was with Islamic Azad University. Now, he is a PhD student at Melbourne University in CLOUDS laboratory.

TABLE II. LOCAL ACCESS PERCENTAGE OF OBJECTS

Algorithm	q=1/4h	q=1/2h	q=3/4h	q=h+1
BCHCMC	54.5%	87.4%	50.3%	50%
BCHNR	42.8%	41.4%	35.6%	38.4%
BCHNSC	31.8%	40%	31.8%	40.4%
BCHNRS	30.8%	39.4%	30%	33.7%
BCHNV	30%	31.4%	29%	30.4%

(a) Number of Nodes, n=100

Algorithm	q=1/4h	q=1/2h	q=3/4h	q=h+1
BCHCMC	52%	53.1%	50.8%	50%
BCHNR	38.4%	40.1%	36.6%	36.1%
BCHNSC	34.4%	35.7%	33%	31.2%
BCHNRS	32.4%	33.9%	31.6%	30.4%
BCHNV	30.4%	31.7%	28.9%	29.9%

(b) Number of Nodes, n=1000

Algorithm	q=1/4h	q=1/2h	q=3/4h	q=h+1
BCHCMC	56.7%	58.3%	52.4%	53.4%
BCHNR	40.1%	42.5%	37.7%	38.9%
BCHNSC	39.7%	41.7%	36.6%	35.6%
BCHNRS	38%	41%	30%	34.3%
BCHNV	30.1%	35.6%	34%	34.2%

(c) Number of Nodes, n=5000

REFERENCES

[1] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006.

[2] A. Benoit, V. Rehn-Sonigo, and Y. Robert, "Replica placement and access policies in tree networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 12, pp. 1614–1627, 2008.

[3] O.Kariv and S. Hakimi, "An algorithmic approach to location problems. ii: The p-medians," *SIAM J.Applied Math*, vol. 37, no. 2, pp. 539–560, 1979.

[4] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1773–1780 vol.3.

[5] M. Karlsson and C. Karamanolis, "Choosing replica placement heuristics for wide-area systems," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, ser. ICDCS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 350–359.

[6] J.-J. Wu, Y.-F. Lin, and P. Liu, "Optimal replica placement in hierarchical data grids with locality assurance," *Journal of Parallel and Distributed Computing*, vol. 68, no. 12, pp. 1517 – 1538, 2008.

[7] V. Rehn-Sonigo, "Optimal replica placement in tree networks with qos and bandwidth constraints and the closest allocation policy," Tech. Rep.

[8] J.-J. Wu, S.-F. Shih, H. Wang, P. Liu, and C.-M. Wang, "Qos-aware replica placement for grid computing," *Concurr. Comput. : Pract. Exper.*, vol. 24, no. 3, pp. 193–213, Mar. 2011.

[9] Z. Du, J. Hu, Y. Chen, Z. Cheng, and X. Wang, "Optimized qos-aware replica placement heuristics and applications in astronomy data grid," *Journal of Systems and Software*, vol. 84, no. 7, pp. 1224 – 1232, 2011.

[10] O. Wolfson and A. Milo, "The multicast policy and its relationship to replicated data placement," *ACM Trans. Database Syst.*, vol. 16, no. 1, pp. 181–205, Mar. 1991.

[11] M. Garmehi and Y. Mansouri, "Optimal placement replication on data grid environments," in *Information Technology, (ICIT 2007). 10th International Conference on*, Dec 2007, pp. 190–195.

[12] Y. Mansouri, S. T. Azad, and A. Chamkori, "Minimizing cost of k-replica in hierarchical data grid environment," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, May 2014, pp. 1073–1080.

[13] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Steiner-optimal data replication in tree networks with storage costs," in *Database Engineering and Applications, 2001 International Symposium on.*, 2001, pp. 285–293.

[14] Y. Mansouri, M. Garmehi, M. Sargolzaei, and M. Shadi, "Optimal number of replicas in data grid environment," in *Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on*, Oct 2008, pp. 96–101.

[15] X. Tang and J. Xu, "Qos-aware replica placement for content distribution," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 10, pp. 921–932, 2005.

[16] Y. Mansouri and R. Monsefi, "Optimal number of replicas with qos assurance in data grid environment," in *Asia International Conference on Modelling and Simulation*, 2008, pp. 168–173.

[17] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, "Qos-aware distributed replica placement in hierarchical data grids," in *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, 2011, pp. 291–299.