A Dynamic Replica Placement Strategy in Grid Environment

Mohammad Rashedur Rahman
Department of Computer Science
University of Calgary
2500 University Drive, N.W
Calgary, Alberta, Canada T2N 1N4
{rahmanm,barker,alhajj}@cpsc.ucalgary.ca

Abstract

Grid computing emerges in part from the need to integrate a collection of distributed computing resources to offer performance unattainable by any single machine. Grid technology facilitates data sharing across many organizations in different geographical locations. 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. One of the challenges in data replication is to select the candidate sites where replicas should be placed, which is known as the allocation problem. One performance metric to determine the best place to host replicas is select for optimum average (or aggregated) response time. We use the p-median model for the replica placement problem. The p-median model has been exploited in urban planning to find locations where new facilities should be built. In our problem, the p-median model finds the locations of p candidate sites to place a replica that optimize the aggregated response time. A Grid environment is highly dynamic so user requests and network latency vary constantly. Therefore, the candidate sites currently holding replicas may not be the best sites to fetch replica on subsequent requests. We propose a dynamic replica maintenance algorithm that re-allocates to new candidate sties if a performance metric degrades significantly over last K time periods. Simulation results demonstrate that the dynamic maintenance algorithm with static placement decisions performs best in dynamic environments like Data Grids.

Keyword: Data Replication, Grid, Replica Allocation, Replica Reallocation, Replica Placement**.**

1. Introduction

The availability of powerful computers and the rapid increase of network speed are leading to an increasingly decentralized approach to providing compute infrastructures. Science today is more collaborative and multi-institutional [6] so access to computing resources and special classes of scientific devices or instruments should be made available to as many users as possible irrespective of their global location. A large number of scientific and engineering applications require a huge amount of computing time to carry out their experiments by simulation [1]. Research driven by this need has promoted the exploration of a new architecture commonly known as "The Grid" for high performance distributed application and systems.

Grid computing can coordinate resource sharing and problem solving across dynamic multi-institutional environments [3]. Large scientific initiatives such as global climate change, high energy physics, and computational genomics require large data collections which are now being curated in various diverse locations. These large data stores must be shared by researchers around the world. High performance Data Grid architectures facilitate these requirements by applying the various technologies required in a coordinated fashion to support data intensive petabyte scale applications. Data replication becomes critical as Data Grids are developed that permit data sharing across many organizations in different geographically disperse locations [10]. One such instance is the hierarchical one envisioned by GriPhyN [8]. It consists of multiple tiers with all data generating at Tier 0; Tier 1 consists of national centers; and below that there are regional centers. Each tier has its own storage capacity, but this can vary from tier to tier. Using the storage capacity at each tier, replicas can be placed at each tier to increase the data availability among different sites. The general

idea of replication is to store copies of data in different locations so data can be easily recovered if one copy at one location is lost or unavailable.

Transferring a file from a server to client consumes a substantial bandwidth. One possible way to reduce the access latency and bandwidth consumption is to replicate data across different sites. Replication also facilitates load balancing and improves reliability by creating multiple data copies. However, the files in Grid are large (*i.e.*, sizes of 500 MB-1 GB are typical) so replication to every site is infeasible. One of the challenges is to locate the candidate sites for replica placement. One approach is to place replica at sites to optimize aggregated response time. Response time is calculated by multiplying the number of request at site *i* with the distance between the nearest replication site to the requester. The sum of the response times for all sites constitute the aggregated response time. We will use the terms *total response time* and *aggregated response time* interchangeably throughout this research. We propose a *p*-median model [9] that finds the locations of *p* candidate sites to place a replica that will minimize the aggregated response time. However, the optimization problem is NP-hard so a large network requires an unacceptable computation time without directing to the optimal solution [5]. Therefore, heuristics are needed that can generate optimal/near-optimal solutions for the *p*-median model. The Lagrangean Relaxation technique is one such heuristic technique that is popular because it provides bounds on the objective function. The Lagrangean technique solves the *p*-median model by locating *p* candidate sites to place replicas optimally. The Grid environment is highly dynamic where user requests and network latency vary constantly. The candidate sites that hold replicas currently may not be the best sites to fetch replicas subsequently. Thus we propose a dynamic replica maintenance algorithm that first finds the optimal/near-optimal cumulative aggregated response time for *K* time periods by allowing relocation with a positive transportation cost and then compare it with current cumulative aggregated response time. The current response time is calculated by adding aggregated response time for *K* periods assuming that replicas are placed at sites that provide the optimal value using the *p*-median problem at period *K=1*. The relocation decision is then made based on the comparison, *i.e.*, if the difference is greater than an allowable threshold.

The paper is organized by presenting related work in Section 2. Optimal static replica placement strategy is discussed in Section 3. Dynamic replica maintenance strategy is presented in Section 4. The simulation model is described in Section 5. Section 6 evaluates and compares the replication strategies. Section 7 concludes and indicates possible future research directions.

2. Related Work

Decision problems arise in wide range of public and private sector environments. Different models such as the set covering model [14], *p*-center, and *p*-median models address such problems. The general problem is to locate objects (or facilities) to optimize some objective. Distance, or some measure more or less functionally related to distance (e.g., travel time or cost, demand satisfaction), is fundamental to those problems. The set covering and *p*-center models are based on maximum distance, whereas the *p*-median model is based on the total (or average) distance. The set covering model locates the minimum number of facilities required to cover all the demand nodes (i.e., requesting sites). The *p*-median model finds *p* locations to minimize the total traverse distance that customers must traverse to reach the closest facility [5]. The *p*-center model addresses the problem of minimizing the maximum distance to the closest facility. Applications for these models include locating bus stops, licensing bureaus, airports, blood bank, emergency medical services, *etc* [4, 5].

Kavitha *et al*. [12] propose a strategy for creating replicas automatically in a generic decentralized peer-to-peer network. The goal of their model is to maintain replica availability with some probabilistic measure. Although the approach may be applicable on DataGrids, each peer only utilizes partial information (the part they retain); so a more global approach is likely to achieve better results. Ranganathan and Foster [10] discuss various replication strategies for a hierarchical DataGrid architecture. They test six different replication strategies: 1) No Replication: only root node holds replicas; 2) Best Client: replica is created for the client who accesses the file the most; 3) Cascading: a replica is created on the path to the best client; 4) Plain Caching: a local copy is stored upon initial request; 5) Caching plus Cascading: combines plain caching and cascading; 6) Fast Spread: file copies are stored at each node on the path to the best client. They show that the cascading strategy reduces response time by 30% over plain caching when data access pattern contain both temporal and geographical locality. When access pattern contains some locality, Fast Spread significantly saves bandwidth over other strategies. The replication algorithms assume that popular files at one site are popular at others. The client site counts hops for each site that hold replicas, and the model selects the one with the least number of hops from the requesting client. However it does not consider current network bandwidth. Our model captures both Grid file transfer and other network traffic over the same link and it considers heterogeneous link capacities that are not necessarily hierarchical.

Kavitha *et al*. [11] develop a family of job scheduling and replication algorithms and use simulation studies to

evaluate them. Three different replica placement algorithms are considered: 1) no active replication; 2) a replica is created at random site based on a threshold; and 3) a replica is created at the site with the smallest number of waiting jobs based on a threshold. These three replication strategies are combined with the four scheduling strategies: 1) jobs are scheduled to a random site; 2) jobs go to the site with fewest waiting jobs; 3) jobs are scheduled to the site containing the required data and with the fewest waiting jobs; 4) jobs are always run locally. They show that when there is no replication, simple local scheduling performs best. However, when a replication is used scheduling jobs to sites containing the required data is better. The key lesson for our study is that dynamic replication reduces hotspots created by popular data and enables load sharing. OptorSim [2] is a simulator developed as a part of European DataGrid project to carry out different replication and scheduling algorithms. The simulator models a Grid consisting of several sites where each site has zero or more computing elements and /or zero or more storage elements. The computing element facilitates job execution whereas the storage sites are data repositories. The simulator also supports routers which forward requests to other sites but do not have any processing power or storage capacity. A resource broker acts as a meta-scheduler that controls the scheduling of jobs to different computing elements. The simulator uses an economic model in which sites buy and sell file using an auction mechanism.

Several research efforts [2, 10, 11] assume user requests is the only parameter considered for replica placement so network latencies are ignored. However, network bandwidth plays a vital role in file transfer. We can save substantial transfer time if we place a replica of a file at a site that is connected to its neighbors with limited bandwidth and if its request for that file is above average. Earlier work [13] shows that considering both the current network state and file requests produce better results than file requests times alone. The replication algorithm begins by placing the master files at one site. The expected utility or risk index is calculated for each site that does not currently hold a replica and then one is placed on the site that optimizes the expected utility or risk. The algorithms proposed based on *utility* selects a candidate site to host a replica by assuming that future requests and current load will follow current loads and user requests. Conversely, algorithms using a *risk* index expose sites far from all other sites and assume the worst case whereby future requests will primarily originate from that distant site. One major drawback of these strategies is that the algorithms select only one site per iteration and places a replica there. The Grid environment is highly dynamic and there might be a sudden burst of requests indicating multiple sties should simultaneously get replicas to quickly satisfy the large spike of requests. Our work does not consider about the relocation of the candidate sites. In this research, we extend earlier work by selecting *p* candidate sites to host replicas as well as a dynamic maintenance using a previous static placement strategy.

3. Static Replica Placement Algorithm

On a Data Grid different jobs are submitted from various sites. Total job execution time measures effectiveness of the replication strategies. Jobs in the Data Grid may request a number of files. If the file is at a local site, response time is assumed to be zero; otherwise the file must be transferred from the nearest replication site. Thus, job execution time includes the latency required to transfer a file. The best replication strategy minimizes the total job execution time and should also minimize the total response time. Response time for a requesting site *i* is given by the product of number of requests at site *i* (*i.e.*, h_i) and the time required between the requesting site *i* and the nearest replication site. Therefore, our objective is to find the *p* best candidate (replication) sites such that total response time for all of the requesting sites is minimized. The identified problem is closely analogous to the *p-median* [9] model used extensively for facility location problems in urban planning. In the following sections we formally restate the model and provide a heuristic solution approach that leads to optimal/near-optimal solution for our replica placement problem.

3.1 P-Median Model

The *p*-median model finds the *p* replica placement sites

$$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} h_i d_{ij} y_{ij} \quad (1)$$

$$\text{Subject to} \sum_{j=1}^{n} x_j = p \quad (2)$$

$$\sum_{j=1}^{n} y_{ij} = 1,\ i = 1,....., n \quad (3)$$

$$y_{ij} - x_j \le 0,\ i = 1,....., n \quad j = 1,.....n \quad (4)$$

$$x_j \in (0,1),\ \ j = 1,.....n \quad (5)$$

$$y_{ij} \in (0,1),\ \ j = 1,.....n\ ;\ i = 1,....., n \quad (6)$$

$$y_{ij} = \begin{cases} 1 \text{ if requesting site } i \text{ is allocated to replication site } j \\ 0 \text{ otherwise} \end{cases} \quad (7)$$

to minimize the request-weighted total distance between the requesting sites and the replication sites holding copies assigned. This model is formulated with the equations presented above.

The objective function (1) minimizes the request-weighted distance between each requesting site and the nearest replication site. Constraint (2) states that exactly p sites to be located to place the replica. Constraint (3) states that each requesting site should be allocated exactly one replication site from which it can fetch the replica. Constraint (4) states that requests at site i can only be assigned at replication site j if a replica is placed at site j. Constraints (5-6) are general integrity constraints. Here h_i represents the requests at site i. For a small network and small number of p any of the well-known algorithms such as branch and bound can be used [5] to solve the p-median problem optimally. However, for a large number of constraints and variables the problem is classified as NP-Hard [5]. Therefore, the problem needs to be solved heuristically that find good solutions to the problem.

3.2. Lagrangean Relaxation: A Heuristic Approach

A major benefit of the Lagrangean heuristic [7] over other heuristic approaches is it gives both upper and lower bounds for the objective function. Thus, it provides a range in which the optimal value of the solution lies. The basic idea is to relax some constraints of the original model and add those constraints, multiplied by Lagrange multiplier to the objective function. We then try to solve the relaxed problem optimally. The model uses a search technique to find a set of values for Lagrange multipliers that lead to a solution of the problem that satisfies the relaxed constraints. If the lower and upper bound of the solution coincides we have found the optimal solution, otherwise we can iterate or search for the best Lagrange multipliers until the gap between upper and lower bound is acceptably narrow. We outline the relaxation algorithm in the next section.

3.2.1 Lower Bound Calculation:

$$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} h_i d_{ij} y_{ij} + \sum_{i=1}^{n} l_i \left(1 - \sum_{i=1}^{n} y_{ij}\right) \qquad \textbf{(8)}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(h_i d_{ij} - l_i\right) y_{ij} + \sum_{i=1}^{n} l_i \qquad \textbf{(9)}$$

If we relax constraint (3) and add this one into objective function, the relaxed problem can be stated as equations (8-9).

The other constraints (2, 4-7) remain the same as the p-median problem. To minimize the objective function in (9) we would like to set $y_{ij} = 1$ if its coefficient $\left(h_i d_{ij} - l_i\right) < 0$ and $y_{ij} = 0$ otherwise. To set he value of y_{ij}, *i.e.*, $y_{ij} = 1$, the corresponding x_j's value should be 1 (by constraint 4). However, constraint (2) states that we can choose at most p replica sites for which $x_j = 1$. Therefore, we have to rank the values of V_j s, where V_j is defined by $V_j = \sum_{i=1}^{n} \min\left(0, h_i d_{ij} - l_i\right)$. Find the p smallest values of V_j that have the largest impact on the objective function. Set the corresponding $x_j = 1$. Then set $y_{ij} = 1$, if $x_j = 1$ and $\left(h_i d_{ij} - l_i\right) < 0$ otherwise set $y_{ij} = 0$. Calculate the lower bound of the solution (Z_{LB}) by finding the objective function from constraint (9) which includes y_{ij} that are set to 1.

3.2.2. Upper Bound Calculation

Recall that in the relaxed problem we relax constraint (2) which states that each requesting site must be assigned to a replication site is eased. The objective function value found by the lower bound program ignores this constraint. Therefore, this constraint may remain unsatisfied which lead to an infeasible solution to the original problem. We must find an upper bound (Z_{UB}) of the objective function by assigning each requesting site to the nearest replication site. The replication sites are found from the lower bound calculation, *i.e.*, the sites for which the corresponding $x_j = 1$.

3.2.3. Multiplier Adjustment

The multipliers are updated by subgradient optimization which maximizes the lower bound of the solution to satisfy the relaxed constraint that is relaxed. The steps for updating the Lagrange Multipliers are given below:

1. Define subgradients G_i^t for the relaxed constraint in the current iteration by:

   $G_i^t = \sum_{j=1}^{n} \left(1 - y_{ij}^t\right)$ $i = 1,..., n$.

2. Define a step size $T^t = \dfrac{p\left(Z_{UB}^t - Z_{LB}^t\right)}{\sum_{i=1}^{n} \left(G_i^t\right)^2}$ where p is initially set to 2. If there is not much improvement after a certain number of iterations p is replaced by $\frac{p}{2}$.

3. With this step size the values of l_i are updated by the following relationship

$$l_i^{t+1} = \max\left(0,\ l_i^n + T^n G_i^t\right) i = 1,...,n$$

The algorithm terminates either after a specified number of iterations or the value of p becomes sufficiently small. A more detailed discussion about this Lagrangean relaxation technique and its application to *p*-median problem can be found elsewhere [4, 5]

4. Dynamic Maintainability of Static Placement

The Lagrangean relaxation technique assures the optimal or near-optimal solution based on the user requests and network characteristics for the current period. However, the candidate sites that hold replicas currently may not be the best sites to fetch replica if the user requests and network latency changes. Therefore, relocation needs to be considered if the performance is to be maintained. However, the relocation is costly. Files in Grid are typically in the magnitude of 500MB-1GB, so before relocating a file replica to another site, we must consider the file transfer cost for this evolution. To determine the performance degradation occurring in last *K* time periods, we must determine the optimal cumulative average response time for *K* time periods if reallocation is permitted while accounting for the transfer costs. Fortunately, the solution of this aspect of the problem also finds the replica placement needed to achieve an optimal /near optimal cumulative average response time. Wesolowsky and Truscott [15] analyze a multi-period facility location-allocation problem that allows facilities to move. They propose a dynamic model that minimizes three factors, (1) distributing cost, (2) construction and removal cost for a given time period, and (3) determines possible facility allocation to achieve the optimal/near-optimal cumulative cost.

In Data Grid system, the performance monitoring is often done by a meta-scheduler or resource broker. To remove a file from a site's local storage the resource broker must send a message, a small overhead message to initiate the much larger file transfer. We will also ignore the cost of removing a file from local storage. We use their dynamic model [15] to find the optimal cumulative total response time for *K* periods:

$$\text{Minimize} \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} h_i^k d_{ij}^k y_{ij}^k + \sum_{k=2}^{K} \sum_{j=1}^{N} c_j^k a_j^k \quad \textbf{(10)}$$

$$\text{Subject to} \sum_{j=1}^{n} x_j^k = p \quad \text{for } k=1.,2...,K \quad \textbf{(11)}$$

$$\sum_{j=1}^{n} y_{ij}^k = 1,\ i = 1,....., n\ ;\ k=1,2,...,K \quad \textbf{(12)}$$

$$y_{ij}^k - x_j^k \le 0,\ i = 1,....., n\ ;\ j = 1,.....,\ n\ ;\ k=1,2,...,K \quad \textbf{(13)}$$

$$x_j^k \in (0,1),\ j = 1,....., n\ ;\ k=1,2,...,K \quad \textbf{(14)}$$

$$y_{ij}^k \in (0,1),\ j = 1,.....,\ n\ ;\ i = 1,....., n\ ;\ k=1,2,...,K \quad \textbf{(15)}$$

$$y_{ij}^k = \begin{cases} 1 \text{ if requesting site } i \text{ is allocated to replication site } j \text{ at period } k \\ 0 \text{ otherwise} \end{cases} \quad \textbf{(16)}$$

$$a_j^k = \begin{cases} 1 \text{ if a replica is relocated at site } j \text{ in period k} \\ 0 \text{ otherwise} \end{cases} \quad \textbf{(17)}$$

$$c_j^k = \text{the transportation cost of the file from nearest replication site to node } j \text{ at period } k \quad \textbf{(18)}$$

Equations (10-16) are the multiperiod versions of (1-7) respectively. The constraint (17) ensures that we can consider the reallocation cost if a replica is relocated on that site. Wesolowsky *et al.* [15] use dynamic programming to solve the mathematical model optimally for a small network size and limited number of periods. For a large network and large value of K, the dynamic programming generates huge state spaces and stages, therefore the authors suggest heuristics to generate good solutions.

We can use the Lagrangean relaxation technique to generate the optimal or near-optimal solutions to the dynamic model. Once achieved, we compare this result with the current one. The current result is calculated by adding total response time for *K* periods assuming that replicas are placed to the sites that gave the optimal value for the *p*-median problem at period *K=1*. We must then decide if relocation is appropriate. Table 1 presents 4 cases to consider when determine if relocation should occur and it also identifies a candidate target. For simplicity, we consider a 2-median problem and 3 time periods for the dynamic model. We compare the current result (CR) with optimal result (OR) and check whether the difference is more than the allowable threshold (T). For example, the solution found by the *p*-median problem at period *K=1* suggests that the replica should be placed at Site A and Site B. We must analyze the performance of this placement decision with respect to three consecutive periods that include the first period when the static optimal decision was made and the subsequent two periods.

Table 1: Replica Reallocation Decision

Case	(CR–OR)>T	Period 1	Period 2	Period 3	Decision
1	No	A, B	A, B	A, B	No Relocation
2	No	A, B	A, B	C, D	Relocate at C, D
3	Yes	X, Y	X, Y	X,Y	Relocate at X, Y
4	Yes	P, Q	R, S	M, N	Re-optimize by *p*- median with aver-age requests and average bandwidth for last 3 periods

In Case 1, we find that the optimal solution complies with our early decision so replicas are placed correctly. In Case 2, we find an optimal solution that suggests replica should be placed at Site A and Site B for time period 1 and 2 but to get the optimal value we should consider a relocation to Site C and D for time period 3. Therefore, we can relocate at C and D at the end of time period 3. In Case 3, we found that site (X, Y) is giving the optimal cumulative response time suggesting that Site(X,Y) show a consistent performance since last three periods even we consider the relocation cost, *i.e*., transportation time of a file to site X, Y from the best candidates (which are A and B currently). Moreover, the difference between current and optimal solution is above the prescribed threshold. The Case 4 addresses a random situation where we are not able to find a set of sites that perform satisfactory throughout the last three time periods. Moreover, the tolerance level is above the threshold, so we must consider relocation. Unfortunately, the sites must now be found by solving the *p*-median problem that takes average request and average network latency as parameters. The averages are calculated by averaging the request and latency for last three time periods.

5. Simulation

Replica placement algorithms must be tested thoroughly before deploying them in real Data Grid environments. One way to achieve a realistic evaluation of the various strategies is through simulation that carefully reflects real Data Grids. On a Data Grid different jobs are submitted from various sites. Mean job execution time is a good measure of effectiveness of the replication strategies. Jobs in the Data Grid request a number of files. If the file is at a local site, response time is assumed to be zero; otherwise the file must be transferred from the nearest replication site. Thus, job execution time incorporates the response time required to transport a file. The best replication strategy minimizes the mean job execution time and also minimizes the average response time. We validate our replica placement algorithms with average response times. We will validate our placement algorithms with mean job execution time in the future. Our replica placement algorithms are evaluated with a simulator written in Java. The simulation generates random background traffic and grid data requests.

.

5.1 Grid Configuration

The study of our replica placement algorithms was carried out using a model of the *EU Data Grid Testbed 1* [2] sites and their associated network geometry. Site 0 is the CERN (European Organization for Nuclear Research) location. Initially all master files are distributed to CERN. A master file contains the original copy of some data samples and cannot be deleted. Each circle in Figure 2 represents a testbed site and a star represents a router. Each link between two sites shows the available network bandwidth. The network bandwidth is expressed in Mbits/sec (M) or Gbits/sec (G). We include the storage capacity at each router, *i.e*., intermediate nodes. The intermediate nodes have higher storage capacity than the testbed sites but smaller capacity than CERN. Placing data at intermediate nodes moves it closer and hence more accessible to testbed sites. The file requests are generated from the testbed sites.

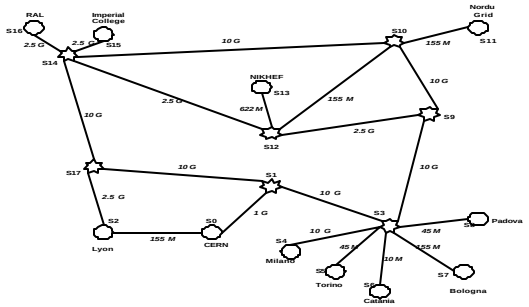

Figure 1: The EU Data Grid Testbed 1 Sites and the Approximate Network Geometry.

5.2 Simulation Input

Our program's input is from two configuration files. One file describes the network topology, *i.e.*, the link between different sites, the available network bandwidth between sites, and the size of disk storage of each site. The second configuration file contains information about the number of requests generated by each testbed site and the current network load. The network load is varied to test the impact on our replication algorithm. We consider low, medium or heavy traffic. File requests may either follow uniform distribution or normal distribution. We set three maximum values for uniform file requests where each testbed site can generate requests that are uniformly distributed with a maximum of 10, 30, or 50.

We also consider ten random normal requests with different mean and variance. The testbed site that generates each of those random requests is chosen arbitrarily. We consider uniform and normal requests with diverse variances to analyze how well the replication algorithms performs when there is no co-relation among previous requests, that is., the requests are totally random.

6. Simulation Results

Each site records the time taken for each file requested to be transferred to it. This time record forms the basis to compare various replication strategies. We compare our replication algorithm with respect to average response time. Response time is the time that elapses from a request for a file until it receives the complete file. The average of all response times for the length of the simulation is calculated. The best replication strategy will have lowest response time. Each file is 100 MB in size. After some initial runs, we place a replica at sites that will optimize either one of the objectives, *i.e.*, the request objective (place the replicas to the *p* sites that request most of the files), static *p*-median, and dynamic *p*-median. The Best_Client strategy considers the request objective. After six (K=6) time periods we consider relocation. For simplicity we set the value of Threshold to zero (T=0), *i.*,*e.*, we consider relocation when we can find an optimum aggregated response for the last six periods that is better than the current accumulated response time. We plan to work with variable threshold in future. We test our algorithm for two different values of *p*, i.e., *p*=5 or *p*=7. We calculate the average response time for future requests in different network load by assuming the replicas are now at the candidate sites.

Table 2. Average response time for different models, network loads, user requests when p=5

Traffic	Request	Best_ Client	Static P-Median	Dynamic P-Median
Low	Uniform (10)	2883	896	714
Medium	Uniform (10)	8765	1669	1368
High	Uniform (10)	9906	2975	1737
Low	Uniform (30)	8216	2734	1753
Medium	Uniform (30)	17140	4205	3310
High	Uniform (30)	25189	7862	5320
Low	Uniform (50)	15483	3434	3184
Medium	Uniform (50)	25115	7243	6054
High	Uniform (50)	28649	12585	6798
Low	Normal	40970	7925	7076
Medium	Normal	86454	16381	11714
High	Normal	57585	16547	12232

We accumulate the average response time for the next sixty runs to analyze the performance of the replica placement algorithms. We also vary the network load with other background traffic to see its impact on the replication algorithm. The results of accumulated average response time (in seconds) are shown in Table 2 and 3.

Table 2, 3 show that the response time increases with increasing requests. There is a strong correlation between response time and user requests as one would expect. We have highest average response time in peak period, *i.e.*, when user requests reach the maximum, as well the background traffic is highest. We include the dynamic traffic condition and random requests to see the impact on the dynamic model. The dynamic model that considers the relocation shows significant performance improvement compared to static and best-client model in different background traffic conditions as well when user requests vary randomly, that is, no relation with previous requests (uniform random), or the future requests are normally distributed and centered on previous requests. We can get a significant performance improvement with dynamic model if the previous best paths become congested because of high background traffic or if current user requests vary significantly.

Table 3. Average response time for different models, network loads, user requests when p=7

Traffic	Request	Best_ Client	Static P-Median	Dynamic P-Median
Low	Uniform (10)	3297	149	146
Medium	Uniform (10)	2865	246	218
High	Uniform(10)	7670	382	381
Low	Uniform (30)	5995	433	409
Medium	Uniform (30)	16642	838	709
High	Uniform (30)	19237	1052	951
Low	Uniform (50)	9549	723	706
Medium	Uniform (50)	30699	1249	1204
High	Uniform (50)	26136	1245	1220
Low	Normal	22243	1312	1312
Medium	Normal	68454	3196	3022
High	Normal	100073	4032	3674

The proposed mathematical models require little computational time to reach the solution by both static and dynamic *p*-median models. The simulation was carried out on a Pentium 4 processor 2GHz with 512 MB RAM. With current network size, the computational time is only 10 seconds on average to reach a solution using static or dynamic *p*-median model.

7. Conclusions

We consider a *p*-median model for the replica placement problem. The model finds the locations of *p* candidate sites to place replica that will minimize the aggregated response time. Due to dynamic nature of Grid, the placement decision may not be optimal for subsequent periods. Therefore, we need to decide about relocation. However relocation needs transportation cost for transferring the file to the relocated sites. We propose a dynamic replica maintenance algorithm that suggest for a relocation of candidate sites by considering the relocation cost. The decision of relocation is made when

the performance metric degrades significantly in last *K* time periods. We validate our model by using a model of the EU Data Grid testbed 1 sites and their associated network geometry. However, we need to decide about the value of *p* for our *p*-median problem that gives a satisfactory response time to the requesting sites. Moreover, the term Threshold (T) needs to be calculated before using the dynamic maintenance algorithm. Its value should not be too small or too large. One of the choices may be to use a value that changes proportionally with respect to the average response time in each time period.

References:

[1] Buyya, R., Abramson, D., and Giddy, J. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid, HPC Asia 2000, May14-17, 2000, pp 283-289, Beijing, China.

[2] Bell, W., D. G. Cameron, L.Capozza, A., P. Millar, K. Stockinger, and F. Zini. OptorSim- A Grid Simulator for Studying Dynamic Data Replication Strategies. International Journal of High Performance Computing Applications, 17(4), 2003.

[3] Chervenak, A., I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards and Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. Journal of Network and Computer Applications, 23(3):187-200, 2000.

[4] Daskin., M.S, Network and Discrete Location Models: Algorithms and Applications, John Wiley & Sons, 1995.

[5] Drezner, Z., and H. W. Hamacher. Facilty Location Applications and Theory, Springer Verlag, Berlin, Germany, 2002.

[6] Foster, I. Internet Computing and the Emerging Grid, Nature Web Matters, 2000.

[7] Fisher, M.L. The Lagrangian relaxation method for solving integer programming problems, Management Science, 27, 1-18

[8] The GriPhyN Project, http://www.griphyn.org

[9] Hakami, S. Optimum location of switching centers and the absolute centers and medians of a graph, Operations Research, 12, 450-459

[10] Kavitha, R., and I. Foster. Design and Evaluation of Replication Strategies for a High Performance Data Grid, in Computing and High Energy and Nuclear Physics 2001 (CHEP'01) Conference.

[11] Kavitha, R., and I. Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. Proceedings of 11th. IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002

[12] Kavitha, R., A. Iamnitchi, and I. Foster. Improving Data Availability through Dynamic Model Driven Replication in Large Peer-to-Peer Communities. Proceedings of Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop, Berlin, Germany, May 2002.

[13] Rahman, R. M., K. Barker and R. Alhajj. Replica Placement on Data Grid: Considering Utility and Risk. IEEE International Conference on Coding and Computing (ITCC), April, 2005, pp. 354-359.

[14] Toregas, C., R. Swain, C. Revelle and L. Bergman, The location of emergency service facileties, *Operations Research*, 19, 1363-1373.

[15] Wesolowsky, G.O., and W.G. Truscott. The multiperiod location-allocation problem with relocation of facilities. Management Science, 22, September, 1975.