http://ansinet.com/itj

ITJ

ISSN 1812-5638

# INFORMATION TECHNOLOGY JOURNAL

ANSInet

Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (4): 590-598, 2008
ISSN 1812-5638
© 2008 Asian Network for Scientific Information

# On P2P and Hybrid Approaches for Replica Placement in Grid Environment

Qaisar Rasool, Jianzhong Li, Shuo Zhang, Ehsan Ullah Munir and George S. Oreku
Harbin Institute of Technology, Harbin 150001, China

**Abstract:** In this study, we present a review of replication techniques that work under peer-to-peer, hybrid and domain-based approaches for Grid environment, with specific emphasis on replica placement. We identify that peer-to-peer technology is contributing a high proportion in Grid data management and replication. A representative set of replica placement schemes were simulated and compared to show their effectiveness.

**Key words:** Data Grid, data replication, peer-to-peer, replica placement

## INTRODUCTION

Grid technology (Foster and Kesselman, 2003) is emerging as a key enabling infrastructure for a wide range of disciplines in science and engineering, including astronomy, high energy physics, molecular biology and earth sciences. The data requirements for these scientific applications are very intensive in nature and need techniques to manage and access resources in an efficient and cost-effective manner. Replication is a technique for improving data access efficiency and fault tolerance in large distributed systems. The general idea of replication is to store copies of data in different locations so that data can be easily recovered if one copy of data is lost. Also, placing data replicas closer to user improves data access performance significantly. Data replication in Grid systems is characterized as an important optimization technique for promoting high data availability, low bandwidth consumption, increased fault tolerance and improved scalability (Abawajy, 2004). The goals of replica optimization is to minimize file access times by pointing access requests to appropriate replicas and pro-actively replicating frequently used files based on access statistics gathered. Existing replication approaches in Data Grids can be divided into four camps, namely Data replication architecture, Data replica placement, Replica selection and Replica consistency (Qin and Jiang, 2006). In this study we focus on the different strategies for dynamic replica creation and placement in Grid environment.

A replica is an exact copy of a file that is linked to the original file through some well-defined mechanism (Guy *et al.*, 2002). In Grid environment, replica management services do not enforce any semantics to (multiple) replicas of a file and thus a replica takes the form of a user-asserted correspondence between two physical files (Chervenak *et al.*, 2000). A replication strategy may be static or dynamic as shown in Fig. 1. In contrast to static replication where a replica persists until deleted or its duration is expired, dynamic replication automatically creates and deletes replicas according to changing access patterns and thus ensures that the benefits of replication continue even if user behavior changes. Dynamic replication strategies tend to reduce hotspots created by popular data and facilitate load sharing, hence suitable for Grid environment. A dynamic replication scheme may be implemented either in centralized or distributed manner.

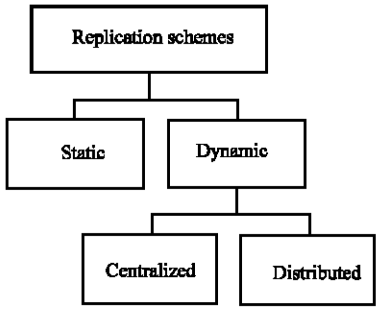

Fig. 1: Types of replication schemes

In any replication scheme, the decision of when to create a replica and where to place it is very important. By placing replicas at strategically critical locations, the maximum benefit can be achieved from the replication process. In this study we present a study of dynamic replica placement strategies used in diverse Grid environments. First, we identify the various system models for Grid and categorize them. Naturally, a replication technique follows the rules of Grid topology. For a multi-tier Grid, a hierarchical replication method is appropriate and for a peer-to-peer (P2P) Grid a different replication method would apply. As first step of our study we recognize the underlying Grid structure and the layout for replication process. Our study encompasses P2P

**Corresponding Author:** Qaisar Rasool, Harbin Institute of Technology, Harbin 150001, China

Table 1: Grid models and replication layout

| Grid model | Layout for replication |
|---|---|
| Tree | Multi-tier; top and middle-tiers possess data; lowest tier is client-tier from which data requests are generated and then replica creation and placement decision is made; clients may be either mutually disconnected or cannot act as server |
| P2P | Structured/unstructured decentralized environment; any peer node can initiate replication and can hold data replicas; alternatively some nodes may be designated as superpeers to act as replica servers |
| Tree and P2P | The Grid is multi-tier and the nodes at client-tier are mutually connected in a P2P-like manner; a client node can transfer replicas to other client nodes |
| P2P and component technology | A loosely coupled P2P Component architecture in which each component connects several peers using an overlay network; various data management aspects like replica creation, replica placement etc are cooperatively handled in separate P2P component layers |
| P2P and Agent technology | Grid nodes are arranged into a circular identifier space with routing tables, using DKS (Distributed K-ary System)-a structured P2P middleware; RTT (round trip time) based replication scheme using agent (ant metaphor) |
| Domain-based | The Grid is organized into domains; each domain have a replica server and many computing nodes; a replication master performs replication in coordination with primary replica servers in the system |

model, domain-base model and the models in which P2P integrated with other technologies such as tree, agent or component. We then explain in detail how replicas are being placed and managed in each of the Grid models. To analyze the different replica placement policies, we summarize the main points of each policy in a table along with pros and cons. The integration of various paradigms used in different replication techniques are depicted in a diagram. We find that P2P technology is a major contributor towards implementing replication in Grid. Further, the agent and component technologies are becoming prevalent for supporting scalability and autonomicity in large Grid systems. Finally, three replica placement techniques were simulated using GridNet simulator and the results presented in a graph showing the effectiveness of these replication strategies.

## GRID SYSTEM MODELS

In research, various system models have been proposed for the data-intensive scientific Grid applications. These include multi-tier, P2P, domain-based and hybrid as shown in Table 1. We term a model as hybrid if it integrates two or more paradigms in its structure and/or its replication layout. A paradigm may be one of the following: tree, P2P, Agent, Component, or domain. The problem of replica placement in tree topology Data Grid has been studied intensively, such as by Ranganathan and Faster (2001), Abawajy (2004), Tang *et al.* (2005), Lin *et al.* (2006), Liu *et al.* (2006) and Rahman *et al.* (2005, 2006). In this study we are concerned with the P2P and hybrid approaches including the domain-based approach for the replica placement in Grid systems.

A multi-tier model, as shown in Fig. 2a, is used for Grid where there is a single source for data and the data has to be distributed among collaborations worldwide. Current Data Grid implementations in the fields of high energy physics (LCG, 2005) and health sciences (EGEE, 2005) use this model. This tiered model has motivated the

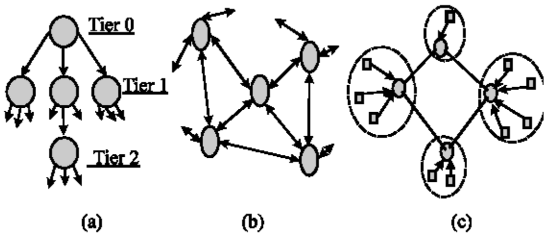

Fig. 2: Multi-tier, P2P and Domain-based architectures for Grid

development of tree-structured replication mechanisms in which replicas are placed either top-down or bottom-up fashion. In such a hierarchical structure, however, data search and movement remain confined to specific paths. With the needs for researchers to collaborate and share products of their analysis efficiently, new models are emerging for Grids which utilize the combined benefits of tree and other architectures such as P2P. A P2P system is characterized by the applications that employ distributed resources to perform functions in a decentralized manner. The key characteristics that distinguishes a P2P system from other resource sharing systems is its symmetric communication model between peers, each of which acts as both a server and a client (Mao *et al.*, 2004). Along with P2P, Agent and Component technologies are also being employed for replication in Grid environment (Table 1).

## P2P APPROACHES FOR REPLICA PLACEMENT

Figure 3 shows generic types of P2P systems. In the centralized model of P2P, data sharing is based around the use of a central server system that keeps directories of shared files whereas the decentralized model of P2P does not use a central server to keep track of files. Instead, the request for a file is spread from one peer to neighboring peers, then to other neighbors and so on. In both cases, a connection between the requester and the owner of the

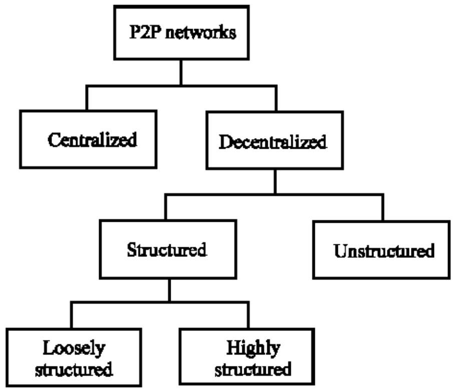

Fig. 3: Categories of P2P systems

file is directly established and the file is transferred (Crowcroft *et al*., 2003). A decentralized P2P network may be structured or unstructured based on whether the system has precise control over the network topology or file placement. The decentralized unstructured P2P networks require no centralized directories and no precise control over network topology or data placement. In loosely structured systems the placement of files is based on hints while in highly structured systems both the P2P network topology and the placement of files are precisely determined (Lv *et al*., 2002). From the viewpoint of resource sharing a P2P system overlaps a Grid system.

In large decentralized P2P systems where the reliability of storage elements is low and bandwidth is limited, data availability is a serious challenge. Replication can be an effective technique for improving data availability in such unreliable systems. A strategy for creating replicas automatically in a generic decentralized P2P network is proposed by Ranganathan *et al.* (2002). The goal of this model-driven approach is to maintain a specific level of availability at all times. To achieve this goal, the optimal number of replicas for each file is calculated and then best nodes are identified to host these replicas. This is done for files that have no replica or have less number of replicas than an availability threshold. The approach relies on a resource discovery service to find available storage and the Network Weather Service (Wolski, 1997) to get network status information. The availability of a file depends on the node failure rate and the accuracy of the Replica Location Service (RLS). The amount of replica availability needed for a given file is modeled as in Eq. 1:

$$RL_{acc} * \left(1-(1-p)^{r}\right) \geq \text{Avail} \qquad (1)$$

where, r is the total number of replicas for a given file, p is the probability of a node to be up, $RL_{acc}$ is the accuracy of the replica location service and Avail is the required amount of availability for the file. For a certain availability threshold, the value of r can be calculated from the above function. For example, if the probability of a node to be up is 30% and the RLS accuracy is 80% then for the availability threshold of 75%, the model recommends a minimum of 8 replicas. After calculating ideal number r of replicas, the system consults RLS to know how many replicas M are actually present. If M is less than r the system creates (r-M) copies of file and distribute them to best candidate nodes. The best nodes are those which maximize the output from replication, i.e., replicaBenefits-replicaCosts. The benefit is the reduction in transfer time to the potential users. The costs are the storage costs at the remote site and the transfer time from the current location to the new location. Let F be a file stored at a location $N_1$ then cost of creating its replica at a new location $N_2$ is given in Eq. 2:

$$\text{replicaCosts} = \text{trans}(F, N_1, N_2) + s(F, N_2) \qquad (2)$$

where, function trans() shows the transfer cost of F from $N_1$ to $N_2$ and function s() represents the storage cost of F at $N_2$. The benefit of creating a replica of F at $N_2$ is given by:

$$\text{replicaBenefits} = \text{trans}(F, N_1, \text{User}) - \text{trans}(F, N_2, \text{User}) \qquad (3)$$

In Eq. 3, User is the location from which we expect the most number of future requests. In this way, the net benefits of replicating F at $N_2$ are calculated and the best candidate identified to place replica.

Another scheme for P2P replication in a decentralized unstructured Grid environment is proposed by Mao *et al.* (2004). The main objective is to decrease network communication cost in the Grid by dynamically placing replicas at the Grid nodes. In this scheme, the concept of superpeer or broker is introduced to control the system scale. These superpeers act as replica servers. The Grid topology is modeled as an undirected graph G(V, E) where V refers to the collection of peers and E represent the edges or links between the peers. Edges are labeled with a cost metric such as bandwidth, distance, or delay. Further it is assumed there are N superpeers in the system and each superpeer can hold at most 1 replica for certain data. The replication policy works as follows. Initially the replicas are distributed randomly. Over the time interval, each superpeer gathers the parameter information such as current system topology status C and access mode A. The system topology status is maintained with the help of a graph from which a minimum cost path matrix C is constructed using dynamic Ford algorithm. The matrix A is used for storing access modes and a sensitivity factor S is used to control the replica movement action. Each node calculates its best position alone according to current access mode A and network system status C.

The algorithm steps are as follows:

```
Step 1: Check and update C, the system cost graph;
        C = Ford(E)
Step 2: Check current access mode A
Step 3: Calculate the current average path and best
        position P for the replica; P = C * A
Step 4: Calculate S, the system sensitivity factor
Step 5: If S>S0 then    //system sensitivity triggers to do
        replica placement/movement
    begin
        List the best N positions from P
        Broadcast move requests to N peers
        Gather the N positions and rank peers without the
        redundant replica by calculating the average cost
        in P
        If there is no such peer then break else Move
        replica to the peer
    end
    Wait for an interval t
go to Step 1.
```

Though all the status info is obtained dynamically in a decentralized way and all decision-making of replica movement is independent among the peer nodes, the data locating and storage resource management are centralized. To make the effective use of grid storage resources, the lifetimes of replicas are set. A replica will be deleted from the system if it is not accessed for a certain time period.

## HYBRID SOLUTIONS FOR REPLICA PLACEMENT

For a multi-tier Data Grid similar to Fig. 2a, Ranganathan and Foster (2001) discussed a bunch of replication techniques, among which we focus on Caching plus Cascading replication that uses P2P concepts. Caching means a client requests a copy from a server and upon receiving that file caches it locally. In cascading, server periodically identifies popular files and transfers their replicas to lower-tier nodes along the path to the node that is generating max requests for that file. Whenever the number of requests for a file gets larger than a threshold, the replica of that file moves down one tier. For example, if a client c at tier-3 generates max requests for a file f then the replica of f will be brought from tier-0 to a node on tier-1 along the path to c. The interesting phenomenon of cascading is that clients are linked together and a client node can transfer replica to its sibling nodes. In other words, a client acts as server for its siblings. This dual behavior makes this technique unique among other replication techniques in multi-tier Grid environment.

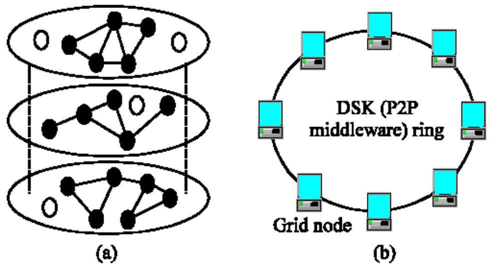

Fig. 4: (a) Three interacting P2P components and (b) DKS ring structure for grid

For a Data Grid tree, a decentralized solution in the form of an adaptive and scalable middleware is presented by Lamehamedi and Szymanski (2007). The approach is inspired by P2P techniques that require no centralized management and advocate self-organization. The components of the proposed middleware include a communication layer, a resource layer and a replica management layer. The replica management layer is responsible for transferring data between Grid nodes and the creation of new replicas. Whenever required, a replica is created by Replica Creation Service at a local site after accumulating access statistics and evaluating incurred cost.

Schintke *et al.* (2003) proposed the use of P2P and component paradigms for Grid data management. Here, various management aspects like replica creation, placement and access etc are cooperatively performed in separate P2P component layers. Within a P2P component system, participating peers connect to form an overlay network. In Fig. 4a, these participating peers are shown by dark grey nodes. The white nodes in the Fig. 4a represent servers that do not run peer algorithm. Each P2P component consists of a peer interface for internal communication, a peer algorithm for system tasks and a component interface to interact with other components. Among various data management components, the 'Availability Manager' component performs the task of replica creation and placement based on a user-defined availability threshold in the following manner:

```
Step 1: For each file, calculate the number of necessary
        replicas
Step 2: Observe the neighborhood and count the number
        of replicas for each file
Step 3: If there are not enough replicas then create new
        copies and place them
Step 4: After a time interval t, repeat the process i.e. go to
        step 1.
```

Vlassov *et al.* (2006) present an approach to build a replica management service for Grids constructed with GT4. The framework utilizes Distributed K-ary System (DKS) which is a P2P middleware that allows Grid nodes to emerge into self-organizing overlay networks with DHT functionality. The DHT is used as an index service with multiple access points to locate replicas given a filename; whereas the DKS routing table is used to direct ant agents to find positions for new replicas. Here the replica managers are connected in a structured P2P network built using the DKS middleware. The framework includes a proactive statistics service that optimizes replica placement based on observed access patterns and replica Round Trip Time (RTT) as a client-specific QoS requirement. In this scheme, RTT is the basic criteria for selecting sites to hold a replica. A node is selected for replica placement if its RTT matches or is smaller than the user specified RTT requirements.

## REPLICA PLACEMENT IN DOMAIN-BASED GRIDS

For a domain-based Grid, Tang *et al.* (2006) present work to analyze the impact of data replication over job scheduling. The Data Grid is envisioned as a set of domains with each domain containing one (or more) replica server node and many computing site nodes, similar to Fig. 2c. The replica server, on behalf of its domain's computing sites, arranges replicas which in turn distributed to the respective computing sites. The author presented both centralized and distributed dynamic replication schemes. The main points of these schemes along with replica placement procedure are summarized below.

The time is divided into sessions and at the start of each session, the replication algorithm is invoked to adjust the replica placement based on the placement in the previous session. If no storage space available in the replica servers then some replicas are evicted to make room for new ones using Least Recently Used (LRU) policy.

**Centralized dynamic replication (CDR):** At the time of replication, replica servers collect the file access info of the past session and submit to a replication master who aggregates this records in an history table H(FileID, NoA). Here NoA stands for Number of Accesses. Replication master then calculates average on NoA and deletes all records from H below this average (i.e., the average number of accesses is used as the threshold to distinguish popular data files.) The records are sorted based on NoA in descending order. The table H now contains the files to be replicated in the Data Grid domains. The CDR algorithm determines where to place the file replicas in one of the following two manners:

**Response-time oriented replica placement (RTPlace):** Let $C_i$ be the computing capability of a computing site I then computing capability of a domain g will be $C_g = \Sigma\,(g)C_i$. Similarly, the computing capability of the sum G of all domains is $C_G = \Sigma C_g$. It is assumed that the data request rate from any domain is proportional to domain's computing capability. Let $\theta$ be the factor that measures this proportion then data request rate for domain g is denoted by $Q(g) = \theta\ C_g$. We can write the request rate from domain g for a data file f as Q (g,f) = Probf. Q(g) = $\theta$. Probf. $C_g$, where $Prob_f$ represents the request probability of file f.

Further assume that $R_f$ be the set of replica servers that contains the replicas or original copy of data file f and let $B_{gj}$ is the bandwidth capacity from any node in domain g to replica server j, then the bandwidth for any node in domain g accessing data file f is given in Eq. 4:

$$AB(g,f) = \max_{j\in R_f} B_{g,j} \tag{4}$$

The average response time of all requests for data f is

$$\begin{aligned} AvgRespTime(f) &= \frac{\sum_{g\in G} \frac{Q(g,f).Size_f}{AB(g,f)}}{\sum_{g\in G} Q(g,f)} \\ &= \frac{Size_f}{C_G}.\sum_{g\in G}\frac{C_g}{\max_{j\in R_f} B_{g,j}} \end{aligned} \tag{5}$$

Since the file size and computing capacity of nodes is relatively constant hence we need to consider only the second, i.e., $\Sigma$ part of Eq. 5 to get minimal average response time for data f. To create a new replica for data file f, the replica placement strategy evaluates every candidate replica server x by calculating function Y as given in Eq. 6. A replica server is considered candidate for hosting a replica if it has enough storage space and does not already possess that replica.

$$Y(f,x) = \sum_{g\in G}\frac{C_g}{\max_{j\in R_f \bigcup x} B_{g,j}} \tag{6}$$

To place the replica, a replica server is selected among candidate replica servers that minimize the value of function Y. The replica is transferred from a replica server that offers the highest transfer bandwidth.

**Server merit oriented replica placement (SMPlace):** Here the assumption is that the selection of replica server can be based on its location relative to all domains in the Data Grid. The location merit of a replica server j is defined as in Eq. 7.

$$\text{Merit}(j) = \sum_{g \in G} \frac{C_g}{B_{g,j}} \tag{7}$$

A small value of Merit(j) means that replica server j is in a more crucial location. Placing a replica in the server that has a small merit value can result in a short average data access time. The merits for all replica servers are calculated and sorted in ascending order to make decision for replica placement.

**Distributed dynamic replication (DDR):** For every data access request from a computing site, the data access records are gathered and exchanged among all replica servers. Every replica server aggregates NoA over all domains for the same data file and creates the overall data access history of the system. At intervals, each replica server will use the replication algorithm to analyze the history and decide for replication. The replication algorithm calculates average on NoA and deletes all records from H below this average. The records are sorted based on NoA in descending order and a replica is created for each file at the respective local replica server.

## RESULTS AND DISCUSSION

It is interesting to measure what is the contributed proportion of a paradigm for a Grid model. We see that P2P paradigm contributes significantly much than any other paradigm, as shown in Fig. 5. The P2P approach has a number of distinguishing features that make it ideally suitable for large Grid systems. P2P systems are simple in functionality and provide basic system resilience. There is no single point of failure. They offer flatness in the sense that all peer execute the same algorithm; no hierarchy and hence no bottleneck. The overall task in the system is done by many collaborative peers. Different peers act based on their locality view and hence promote scalability.

Besides these benefits of P2P approach, there may be pitfalls in the system, e.g., the issue of node availability. As pointed by Anirban and Kitsuregawa (2005), the replication in P2P/Grid system is facing new challenges in terms number of replicas required and the proper placement of these replicas because of dynamic nature of peers to enter and leave the system. When it is decided to place a replica at a particular node, either the requester node or a replica server, this decision is merely a request to that node by the replication manager for hosting the replica. From the point of view of QoS, the best place to put a replica would be the requester's own storage, though requester may refuse to host replica due to e.g., insufficient storage space.

Replica placement issue involves the transfer or movement of replica from one node to another consuming the network bandwidth and thus increasing the network transfer cost. Assuming that all data is in the form of files then transfer speed for Grid data can be increased by parting the files into transferable chunks and sending them on the destination in parallel as suggested by Fan *et al.* (2006) in their file-parted replication scheme.

In our view, the proposal of domain-based topology for Grid and the superpeer structure for Grid has

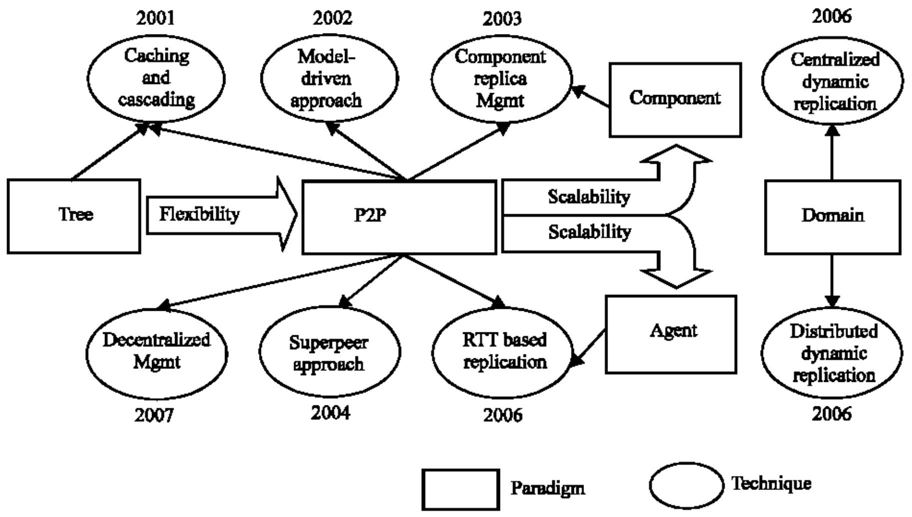

Fig. 5: Integrated paradigms for data replication in Grid

Table 2: Pros and cons of replica placement strategies in P2P, hybrid and domain topology grids

| Replication scheme | Replica placement methodology | Pros (+) and cons (-) | Performance metric |
|---|---|---|---|
| **P2P** | | | |
| Dynamic Model-driven approach (Ranganathan *et al.*, 2002) | Computing ideal number of replicas required for a file by doing cost/benefit analysis and placing replicas at nodes from which max future benefit is expected | + no single point of failure<br>- possibility of excessive replicas as each peer takes his decision independently | Replica availability |
| Superpeer approach (Mao *et al.*, 2004) | Modeling costs between superpeers using an adjacency graph and 0/1 vector for replica placement sites | + scalable<br>+ better performance over static approach | Network communication cost |
| **Hybrid** | | | |
| Caching and cascading (Ranganathan and Foster, 2001) | in a multi-tier Grid, clients requesting data get a copy to be stored in their local cache (caching) and periodically replicas of popular files are sent down to lower tiers (cascading) | + Client can act as Server for siblings<br>+ better performance than cascading or caching alone<br>- No workload balancing | Response time and bandwidth savings |
| Component replica management (Schintke *et al.*, 2003) | an 'availability manager' Component creates and places replicas of a file when number of replicas are falling below than a threshold | + combined use of P2P and Component paradigms<br>+ self-management and self-optimization | Replica availability and system scalability |
| RTT-based replication (Vlassov *et al.*, 2006) | P2P middleware using DKS in which replicas are placed according to access pattern and QoS (RTT) requirements | + improved scalability using P2P concepts and agent technology | Response time and system scalability |
| Decentralized Middleware Framework (Lamehamedi and Szymanski, 2007) | a Replica Creation Service creates replicas based on access pattern and evaluating incurred costs using a cost model | + no centralized management<br>+ top-down and bottom-up data placement models evaluated | Response time |
| **Domain-based** | | | |
| Centralized dynamic replication (Tang *et al.*, 2006) | response-time oriented and server merit oriented replica placement methods | + simple, straight-forward solution<br>- individual sites not taken into consideration | Response time |

architectural resemblance if we assume that each superpeer serves replicas to a specific set of neighboring peer nodes. The differences between two approaches also exist. For example, in domain topology Grid, each domain have one virtual replica server (in actual they may be one, two or more) to facilitate the domain's computing sites whereas in superpeer approach, there is no such distinction of storage and computing nodes and a normal peer can take the form of superpeer as soon as it stores the replica. The promising features of these two approaches can be used for integrating the Grids around the globe, especially those which share a common application area such as high energy physics.

On the scalability end, the loosely coupled P2P Component paradigm and structured DKS P2P middleware may be very effective. In P2P Component layer model, many complex tasks are divided into subtasks and distributed to various component layers in the system. All components run concurrently and the tasks are cooperatively solved by the separate component layers with each layer optimizing the system with respect to its goals. This yields an easy-to-manage environment where system performance is optimized gradually rather than in one step. And the analogy we detect as in DKS system where the role of component is performed by agents. Agents bring ease in management and different Grid middleware for data and storage resource management can be implemented by using agent technology. The DKS self-organizes itself as nodes join and leave and uses symmetric replication for better load balancing and reliability. For replica placement, it employs ants which are complex adaptive systems often used to study and engineer the behavior of large-scale biological, social and computer systems.

The pros and cons of different replica placement techniques are shown in the Table 2.

In the presence of diverse assumptions made by different replication techniques regarding Grid topology, number of nodes, link bandwidth and layout for replication, it is challenging to create a common ground for comparing these techniques. For the sake of convenience, we take a small set of representative techniques and compare them with the base case of no replication. Among several P2P techniques we select Superpeer approach (Mao *et al.*, 2004) and for the hybrid case, we choose decentralized middleware technique (Lamehamedi and Szymanski, 2007).

To make a quantitative analysis of these replica placement techniques, we have used GridNet simulator (Lamehamedi *et al.*, 2003). This simulator is written in C++ and built on top of the ns (Network Simulator). We have considered a 12-node Data Grid topology and distinguish between two types of nodes. One type of nodes is client nodes; they generate data requests and possess a very small storage capacity to hold maximum two data replicas. And the others are storage nodes with relatively very large storage capacity; they act as replica servers. The details of link bandwidth among Grid nodes are given below:

**Scenario 1-Decentralized middleware:** The topology is multi-tier. Bandwidth among all nodes is 100 MB except between clients and other nodes which is 10 MB.

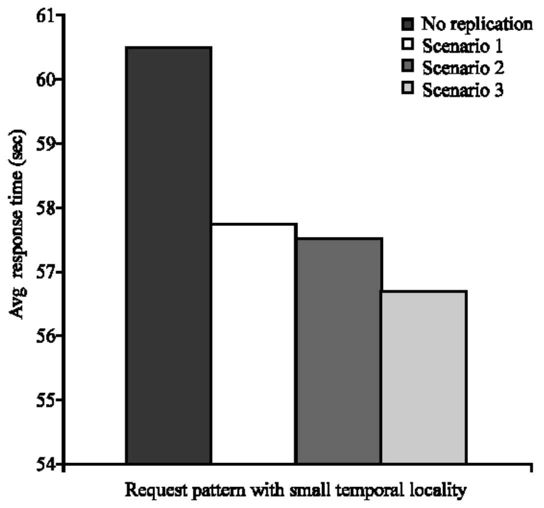

Fig. 6: Performance analysis of replication techniques

**Scenario 2-Superpeer approach:** The structure is P2P. Superpeers, or the replica servers, have bandwidth from 100 to 300 MB among them. Peers act as clients and are linked to superpeers with 10 MB bandwidth.

**Scenario 3-Domain-based technique:** We consider a replica server and its associated client nodes collectively as a domain. The domains are connected to each other through their respective replica servers and the link bandwidth among domains ranges from 100 to 300 MB.

In each of these scenarios, the data is considered to be read-only and so there are no consistency issues involved. The simulation starts by specifying the grid topology. Initially, it is assumed that no replica exists in the system and all data is held at a main replica server. When a client generates a request for a file, the replica of that file is fetched from the main replica server and transported and saved to the local cache of the client. After an initial run, the access statistics are gathered and are used for the decision of replica creation and placement. When a replica is being transferred from one node to another, the link is considered busy for the duration of transfer and cannot be used for any other transfer simultaneously. We used an access pattern with a small degree of temporal locality. That is, some files were requested more frequently than others and such requests were 25% of the whole. For each simulation run, we use 6 files of same size and gradually increase the size starting 100 MB to 1 GB in subsequent runs. For each client node, we keep a record of how much time it took for each file that it requested to be transported to it. The average of this time for various simulation runs is calculated and used as a basis to compare the replication techniques.

The results of the simulation are presented in the Fig. 6. For most of the simulation output, we found that all scenarios produce almost the similar results with very small difference in the response time. The graph shows a relatively low response time from scenario 3 (domain-based technique). This is because the time for replica transfer from domain's replica server to domain's clients is considerably low due to high-speed link for intra-domain. Moreover, the replica is fetched from the remote replica server that offers a high bandwidth transfer. In contrast, the scenario 1 and scenario 2 both observe a low bandwidth between clients and the other storage nodes and thus incurred a high response time.

## CONCLUSIONS

The research community is recognizing the need of new techniques and paradigms for replication to ensure efficient access and distribution of data based on real-time application demands. For our study, we concentrate on replica placement and provide a survey of different replication schemes that utilize new paradigms and techniques. P2P technology is now mature enough to allow for a better design of large-scale grid systems. Despite many exciting features that P2P systems present, still there are challenges to face in the form of scalability and data availability issues in Grid environment. We conclude that dynamic replica placement techniques under integrated Grid paradigm can manage high data availability. The quantitative analysis under a simulated Grid scenario shows the promising results of various replication schemes.

## REFERENCES

Abawajy, J.H., 2004. Placement of file replicas in data grid environments. In: Proceedings of International Conference on Computational Science, LNCS 3038, pp: 66-73.

Anirban, M. and M. Kitsuregawa, 2005. Effective dynamic replication in wide-area network environments: A perspective. In: Proceedings of International Workshop on Database and Expert Systems Applied (DEXA'05), pp: 287-291.

Chervenak, A., I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, 2000. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. J. Network Comput. Appl., 23 (3): 187-200.

Crowcroft, J., T. Moreton, I. Pratt and A. Twigg, 2003. Chapter Peer-to-Peer Technologies. In: The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann.

EGEE (Enabling Grids for E-sciencE), 2005. http://www.eu-egee.org/.

Fan, Q., Q. Wu, Y. He and J. Huang, 2006. Transportation strategies of the data grid. In: Proceedings of the 1st International Conference on Semantics, Knowledge and Grid (SKG'05).

Foster, I. and C. Kesselman, 2003. The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann.

Guy, L., P. Kunszt, E. Laure, H. Stockinger and K. Stockinger, 2002. Replica Management in Data Grids. Technical Report, Global Grid Forum Info Document, GG5, Edinburgh, Scotland.

Lamehamedi, H., B.K. Szymanski, Z. Shentu and E. Deelman, 2003. Simulation of dynamic data replication strategies in data grids. In: Proceedings of International Parallel and Distributed Processing Symposium (IPDPS'03).

Lamehamedi, H. and B.K. Szymanski, 2007. Decentralized data management framework for data grids. Future Generation Computer Systems (FGCS-Elesvier), 23 (1): 109-115.

LCG (LHC Computing Grid), 2005. http://www.cern.ch/LHCgrid/.

Lin, Y.F., P. Liu and J.J. Wu, 2006. Optimal placement of replicas in data grid environment with locality assurance. In: International Conference on Parallel and Distributed Systems (ICPADS'06).

Liu, P., Y.F. Lin and J.J. Wu, 2006. Optimal replica placement strategy for hierarchical data grid systems. In: International Symposium on Cluster Computing and the Grid (CCGrid'06).

Lv, Q., P. Cao, E. Cohen, K. Li and S. Shenker, 2002. Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th Annual ACM International Conference on Supercomputing (ICS'02), New York.

Mao, F., H. Jin, H. Chen, S. Wu and D. Zou, 2004. P2P based decentralized dynamic replica placement strategy in grid environment. Technical Report, Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan, China.

Qin, X. and H. Jiang, 2006. Chapter Data Grids: Supporting Data-Intensive Applications in Wide-Area Networks. In: High Performance Computing: Paradigm and Infrastructure. John Wiley and Sons, pp: 481-494.

Rahman, M.R., K. Barker and R. Alhajj, 2005. Replica placement in data grid: A multi-objective Approach. In: Grid and Cooperative Computing (GCC'05), pp: 645-656.

Rahman, M.R., K. Barker and R. Alhajj, 2006. Replica placement design with static optimality and dynamic maintainability. In: International Symposium on Cluster Computing and the Grid (CCGrid'06).

Ranganathan, K. and I. Foster, 2001. Identifying dynamic replication strategies for a high-performance data grid. In: Proceedings of International Grid Computing Workshop, LNCS 2242, pp: 75-86.

Ranganathan, K., A. Lamnitchi and I. Foster, 2002. Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In: Proceedings of 2nd International IEEE/ACM Symposium on Cluster Computing and the Grid (CCGrid'02), pp: 376-381.

Schintke, F., T. Schutt and A. Reinefeld, 2003. A framework for self-optimizing grids using P2P components. In: Proceedings of 14th International Workshop on Database and Expert Systems Applied (DEXA'03), pp: 689-693.

Tang, M., Bu-Sung Lee, Chai-Kiat Yeo and Xueyan Tang, 2005. Dynamic replication algorithms for the multi-tier data grid. Future Generation Computer Systems (FGCS-Elesvier), 21: 775-790.

Tang, M., Bu-Sung Lee, Xueyan Tang and Chai-Kiat Yeo, 2006. The impact of data replication on job scheduling performance in the data grid. Future Generation Computer Systems. FGCS-Elesvier, 22 (3): 254-268.

Vlassov, V., D. Li, K. Popov and S. Haridi, 2006. A scalable autonomous replica management framework for grids. In: Proceedings of IEEE John Vincent Atanasoff (JVA'06), International Symposium on Modern Computing, pp: 33-40.

Wolski, R., 1997. Forecasting network performance to support dynamic scheduling using the network weather service. In: Proceedings of the 6th IEEE International Symposium on High Performance Distr. Computing, pp: 316-325.