Integration of Scheduling and Replication in Data Grids

Anirban Chakrabarti, R.A. Dheepak, and Shubhashis Sengupta

Software Engineering and Technology Laboratory,
Infosys Technologies Ltd, Bangalore (India)
Tel: 91 80 852 0261
{anirban_chakrabarti, dheepak_ra,
shubhashis_sengupta}@infosys.com

Abstract. Data Grids seek to harness geographically distributed resources for large-scale data-intensive problems. Such problems involve loosely coupled jobs and large data sets distributed remotely. Data Grids have found applications in scientific research fields of high-energy physics, life sciences etc. as well as in the enterprises. The issues that need to be considered in the Data Grid research area include resource management for computation and data. Computation management comprises scheduling of jobs, scalability, and response time; while data management includes replication and movement of data at selected sites. As jobs are data intensive, data management issues often become integral to the problems of scheduling and effective resource management in the Data Grids. The paper deals with the problem of integrating the scheduling and replication strategies. As part of the solution, we have proposed an Integrated Replication and Scheduling Strategy (IRS) which aims at an iterative improvement of the performance based on the coupling between the scheduling and replication strategies. Results suggest that, in the context of our experiments, IRS performs better than several well-known replication strategies.

1 Introduction

In an increasing number of scientific and enterprise applications, large data collections are emerging as important resources that need to be shared and accessed by research teams dispersed geographically. In domains as diverse as global climate change, high energy physics, and computational genomics, the volume of interesting data will soon total petabytes[1]. The combination of large data size, geographic distribution of users and resources, diverse data sources, and computationally intensive analysis results in complex and stringent performance demands that are not satisfied by any existing data management infrastructure. The literature offers numerous point solutions that address the issues of data management, data distribution and job scheduling (e.g., see [2,3]). However, no integrating architecture exists that allows one to identify requirements and components common to different systems and hence apply different technologies in a coordinated fashion to a range of data-intensive application domains. Motivated by these considerations, researchers have launched a collaborative effort called *Data Grids* to design and produce such an integrating architecture.

L. Bougé and V.K. Prasanna (Eds.): HiPC 2004, LNCS 3296, pp. 375–385, 2004.
© Springer-Verlag Berlin Heidelberg 2004

1.1 Motivation and Objectives

Most previous scheduling work has considered data locality/storage issues as secondary to job placement. Casanova et al. [4] describe an adaptive scheduling algorithm for parameter sweep applications that uses a centralized scheduler to compute an optimal placement of data prior to job execution. Banino et. al [5] talks about scheduling in a heterogeneous scenario. Work on data replication strategies for Grids includes [6], where the authors examined dynamic replica placement strategies in a hierarchical Grid environment. Recently, some work has been carried out which combines the scheduling and replication strategies to provide better overall performance in Data Grids [7]. Paper [8] talks about combining the replication and scheduling strategies in a more organized manner. The authors assumed three components: an External Scheduler (ES), which determines where (i.e. to which site) to send jobs that originate at that site; a Local Scheduler (LS), which determines the order in which jobs that are allocated to that site are executed; and a Data Scheduler (DS), responsible for determining if and when to replicate data and/or delete local files. The Grid architecture considered in this paper is similar to one proposed in [8].

In Data Grid, both scheduling and replication aim at reducing the latency for job execution. While scheduling does that by directing the jobs to certain sites so that the latency involved in data movement and job processing is reduced, replication moves the data around so that the data access time during scheduling is reduced. The key contribution of the paper lies in the idea of the possible integration between scheduling and replication called *Integrated Replication and Scheduling (IRS)* Approach. Most of the works in this field have concentrated either on replication or scheduling aspects of the problem. Though, some hybrid strategies have been proposed in [7], the first real effort to study the combination of these two strategies was first done in [8]. In [8], the authors have assumed that at a time each job will access only a single data resource like a file. However, in practical situations one job may require multiple files. In this paper, we propose a replication-scheduling algorithm which iteratively improves the performance of the Data Grids. The main objectives of the paper are to develop and evaluate an iterative replication and scheduling strategy.

The assumptions made are: (i) Data Grid is considered to be an undirected graph. Hence, the transfer cost is same both ways, (ii) a two-stage scheduling as mentioned in [8] is assumed, (iii) the Grid is more or less stable i.e., the chances of link and node failures and rare, (iv) the data is mostly handled in a read-only mode, (vi) the jobs are non-preemptable. The rest of the paper is organized as followed. In Section 2 we outline our IRS algorithms in detail with suitable examples. In Section 3, we present and discuss the performance test results vis-à-vis some other approaches. We conclude in Section 4 by pointing out the salient contributions and future work.

1.2 Data Replication (DR) and Job Scheduling (JS) Problem

We model a job request as a 3-tuple J = $< S, \tilde{F}, \tilde{C} >$, where Sj is the site at which the job is fired, $\tilde{F}$ is the list of files needed by the job and $\tilde{C}$ is the computation time required by the job J at site S_j. A site is modeled as a 3-tuple S = $< \hat{F}, V, P_s >$, where

$\hat{F}$ is the set of files stored in the site S, V is the storage capacity at that site and P_s is the computation capacity at that site. It is to be noted that P_s is expressed in sec/GB. In [8], the authors have stated that P_s varies between 10 sec/GB to 50 sec/GB. The *Job Scheduling (JS)* problem states that: Let J_i be a job, and $\hat{S} = \{ S_1, S_2, \ldots S_n \}$ be the set of sites, then the problem is to schedule the job J_i to a site S_j, where $S_j \in \hat{S}$, such that the latency between submitting the job and job execution is minimized. A Demand Matrix ${D_{F_i S_j}}^T \; \forall i = 1 \ldots n, j = 1 \ldots m$, is created based on a set of jobs J within a time interval T. The replication involves creation of identical copies of data files and their distribution over the nodes in a Grid. The Data Replication (DR) problem states that: Let ${D_{F_i S_j}}^T$ be a demand matrix and $\hat{S}$ be a set of sites; the aim is to distribute a set of files to the sites, so that the latency is minimized based on the demand matrix and the volume constraint at each site is maintained. In this paper, an *Integrated Replication and Scheduling (IRS)* approach is proposed which combines the replication and scheduling schemes. Data Replication (DR) algorithm is a centralized algorithm running at certain interval of time. After the arrival of jobs, each External Schedulers take the help of replication information and schedule so that the job scheduled has the least latency in terms of execution.

2 Integrated Replication and Scheduling (IRS) Approach

We start by defining some operational terms.

Normalized Demand (η_{F_i}): Ratio of the demand for file F_i to the demand of all files.

$$\eta_{F_i} = \frac{\sum_{j=1}^{n} D_{F_i S_j}}{\sum_{j=1}^{n} \sum_{i=1}^{m} D_{F_i S_j}} \tag{1}$$

File Latency (Δ^k_{ij}): Latency for a file F_k to be moved from site S_i to S_j.

Computational Latency (ω_{ij}): Latency for a job i to be executed at site S_j

$$\omega_{ij} = \frac{\sum_{i=1}^{m} \tau_i}{P_j} \tag{2}$$

Queuing Latency (Q_{ij}): Latency for a job i due to the queuing at the site S_j (queue size (q_i). In case of assumption that all the jobs take the same time for execution, then

$$Q_{ij} = \frac{q_j \sum_{i=1}^{m} \tau_i}{P_j} \tag{3}$$

Slots Available (γ_j): Average number of files that can be stored in site S_j. Thus,

$$\gamma_j = \frac{V_j}{\bar{\tau}}, \; \bar{\tau} = \text{average file size} \tag{4}$$

2.1 Job Scheduling (JS) Algorithm

The JS algorithm has two parts – (a) Job Scheduling and (b) Matrix Updating.

Job Scheduling Strategies: Two different Job Scheduling Strategies have been proposed: Matching based Job Scheduling (MJS) and Cost Based Job Scheduling (CJS).
Matching based Job Scheduling (MJS): In MJS, the jobs are scheduled to those sites which have the highest match in terms of data (maximum number of files for the job available at the site). Any tie is broken by reducing the latency involved in moving the data which is not present in the scheduled site from the site(s) containing the data. It is possible that MJS may distribute the jobs to the same site resulting in the queue size increase in that site. To distribute the jobs to different sites the scheduling is done based on $v = m.\frac{\overline{q}}{q_i}$ factor, where m is the maximum match and $\overline{q}$ is the average queue size and q_i is the queue size at the site. MJS schedules based on the maximum v value. Figure 1 shows the topology of a Data Grid. S1, S2, S3 and S4 are the different sites in the Data Grid. The numbers and the arrows show the latency to move a file from one data site to the other. The elements in each site indicate the files that are present in each of those sites. Let a job come which requires files D1, D3 and D6. According to the MJS algorithm, both S2 and S4 are candidate sites where the job can be scheduled. If the job is scheduled in S2, then it takes 7 seconds to move the file D6 from S3 (File Originating Site) to S2. On the other hand, if the job is scheduled onto S4, then it takes 4 seconds. Therefore, the job is scheduled onto site S4.

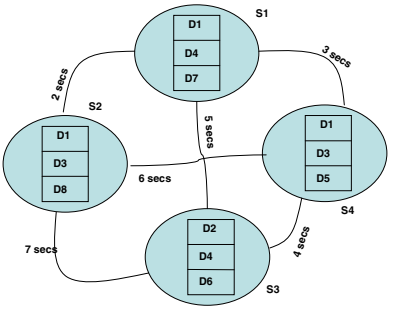

Fig. 1. Topology of an example Data Grid

Cost Based Job Scheduling (CJS): Another alternative to matching based job scheduling, a cost based job scheduling strategy is proposed. Cost ($C_{ij}{}^{s}$) of scheduling a job J_i onto a site s_j is defined as the combined cost of moving the data into the site s_j, latency to compute the job J_i in the site S_j and the wait time in the queue in the site S_j. The job is scheduled onto the site which has the minimum $C_{ij}{}^{s}$. Referring back to the example shown in Figure 1, we assume that in this case the computational time is 0 and queues at each site is also 0. Therefore $C_{ij}{}^{s}$ is composed of only the data latency. The values of

$C_{ij}{}^{s}$ for j=1,2,3,4 are: $C_{i1}{}^{s} = 7$ secs, $C_{i2}{}^{s} = 7$ secs, $C_{i3}{}^{s} = 8$ secs, $C_{i4}{}^{s} = 4$ secs. Therefore, the job is scheduled onto site S4, same as MJS. Though both the algorithms provide similar performance in this example, generally CJS will be better if instantaneous queue information is available. However, in case of partial information the comparison between these algorithms can be an interesting future study.

Updating the Demand Matrix: In this step, the Demand Matrix is updated as illustrated below. The example is based on the topology shown in Fige 1. Let the files required by job J_i be $\tilde{F}_i$. The data files required are: $\tilde{F}_1 = (D1, D3, D4)$, $\tilde{F}_2 = (D1, D2, D5)$, $\tilde{F}_3 = (D2, D3, D8)$, $\tilde{F}_4 = (D1, D3, D7)$, $\tilde{F}_5 = (D4, D5, D8)$, $\tilde{F}_6 = (D1, D5, D7)$, $\tilde{F}_7 = (D3, D4, D5)$, $\tilde{F}_8 = (D1, D7, D8)$. Based on the job requests, the given topology and the MJS; the following Demand Matrix can be constructed:

Table 1. Demand Matrix for the topology in Figure 1 based on a job pattern

Files\Sites	S1	S2	S3	S4	Total
D1	0	0	0	1	1
D2	0	0	4	0	4
D3	0	2	1	1	4
D4	0	1	1	0	2
D5	0	0	0	1	1
D6	0	1	3	2	6
D7	0	0	1	1	2
D8	0	2	2	0	4
Total	0	6	12	6	24

Scheduling of Jobs with Data Ordering: Till now we have assumed that each job requires data all at the same time i.e., at the time of starting the job. However, the cost of scheduling can be modified in case of order of data files. By order of data files we mean that say the job requires files (f_1, f_2, f_3) at the start of the job and requires (f_4, f_5) later. Then files f_4, f_5 can be obtained later than files f_1, f_2, f_3. Therefore,

$$Latency = \max\left(\sum_{j=1}^{3} {}^{s}\Delta_{ij} + \sum_{i=1}^{3}\omega_{ij}, \sum_{j=4}^{5}\Delta^{s}{}_{ij}\right) + \sum_{i=4}^{5}\omega_{ij}, \tag{5}$$

Let $f_{11}, f_{12} \cdots f_{1k_1}$ be the set of files required initially (Step 1), $f_{21}, f_{22} \cdots f_{2k_2}$ be the number of files required in Step 2, $f_{j1}, f_{j2} \cdots f_{jk_j}$ be the files required in Step j, and $f_{p1}, f_{p2} \cdots f_{pk_p}$ be the files required in Step p (last step), then

$$\begin{aligned} L(i) &= \max\left(L(i-1), \sum_{j=1}^{K_i}\Delta_{ij}\right) + \sum_{j=1}^{K_i}\omega_{ij} \\ L(1) &= \sum_{j=1}^{K_1}\Delta^{s}{}_{ij} + \sum_{j=1}^{K_1}\varpi_{ij} \end{aligned} \tag{6}$$

Where L(i) is the latency at the i[th] step, K files are required at step i and L(p) is the total latency.

2.2 Data Replication Strategy

Data Replication Strategy has two steps: (i) Allocation of Replication Limits to each file and (ii) Replication.

Allocation of Replication Limits: We define Replication Limit (χ_i) of file F_i as the number of sites where the file F_i should be replicated. It is to be noted that each file should be replicated at least once, therefore the χ_i is defined as:

$$\chi_i = \min(\#sites, 1 + (ceiling(\eta_{F_i} . (\sum_{j=1}^{n} \gamma_j - \Phi))) \tag{7}$$

In the Equation 7, the minimum of the ceiling value and the #of sites is taken as a file could not be replicated more than the number of available sites. During the allocation of χ_i, priority is given to the files having the highest Normalized Demand (η_{F_i}). Based on the Demand Matrix shown in table 1, the following η_{F_i} values are calculated for each of the different files: $\eta_{D1} = 1/24, \eta_{D2} = 1/6, \eta_{D3} = 1/6, \eta_{D4} = 1/12, \eta_{D5} = 1/24, \eta_{D6} = 1/4, \eta_{D7} = 1/12, \eta_{D8} = 1/6.$ Based on the η_{F_i} values calculated above, the Replication Limit allocations are: $\chi_{D1} = 1, \chi_{D2} = 2, \chi_{D3} = 2, \chi_{D4} = 1, \chi_{D5} = 1, \chi_{D6} = 2, \chi_{D7} = 1, \chi_{D8} = 2.$

Data Replication: Data Replication Strategy is based on the principle of choosing sites based on the expected latency that the site is going to provide. We assume that the data is placed at various storage elements as a result of replication strategy and no further caching takes place. Let the probability of a job scheduled in site S_k requiring the file F_i be p_{ik}. Then the expected file latency (δ_{ij}) of the file F_i scheduled in site S_j is defined as:

$$\delta_{ij} = \sum_{k=1}^{n} p_{ik} . l_{kj} \tag{8}$$

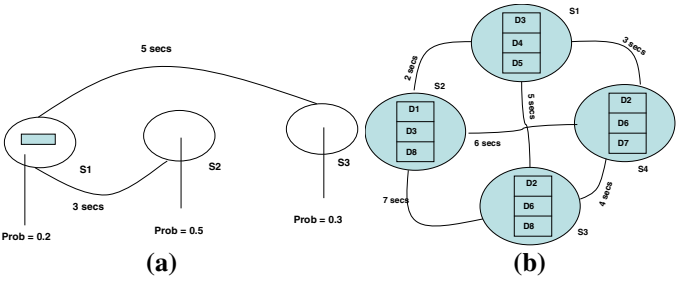

Fig. 2. Illustration of the (a) Expected Latency, (b) Final Config

Figure 2(a) illustrates the concept of expected latency. Let a file be replicated in Site 1. Probabilities of using the file by a job scheduled in site S1, S2 and S3 are 0.2, 0.5 and 0.3 respectively. Therefore the expected latency of the file in site S1 becomes (0*0.2 + 3*0.5 + 5*0.3) = 3 seconds. It is to be noted that the latency of the job requiring the file in site S1 is 0, as the site contains the file. Similarly, Expected Computation (θ_{ij}) and Queuing Latencies (α_{ij}) are given by:

$$\theta_{ij} = \sum_{k=1}^{n} p_{ij}.(\tau_j / P_j), \quad \alpha_{ij} = q_j.\theta_{ij} \tag{9}$$

Cost function for data replication strategy is $C^R_{ij} = \delta_{ij} + \theta_{ij} + \alpha_{ij}$, the algorithm aims at minimizing C^R_{ij}. It is to be noted that this is an NP-Complete problem as it can be reduced to a K-Median problem [9]. The Data Replication Strategy is based on a simple greedy approach. Based on the Demand Matrix, a Normalized Demand is calculated (η_{F_i}) (See Equation 1). Replication algorithm starts with the node having the maximum η_{F_i} value. Then Replication Limit (χ_i) is calculated, where χ_i indicates how many times the file F_i is replicated in the Data Grid. More the value of η_{F_i}, higher is χ_i. The best χ_i sites are selected among n sites which has the lowest C^R_{ij}. Then replication is carried out by the file having the second highest η_{F_i} value. This process is repeated until all the files are exhausted.

An Example: The data replication strategy is shown in Table1 and Figure 1. The cost of the strategy is provided in Table 2.

Table 2. Cost table based on the data replication strategy

Files\ Sites	S1	S2	S3	S4
D6	3.83	5.50	**2.50**	**3.00**
D8	3.50	**3.50**	**3.50**	5.00
D3	**3.00**	**3.25**	4.50	4.00
D2	5.00	7.00	**0.00**	**4.00**
D4	**3.50**	3.50	3.50	5.00
D7	4.00	6.50	2.00	**2.00**
D5	**3.00**	6.00	4.00	0.00
D1	3.00	**6.00**	4.00	0.00

The final replication is shown in Figure 2(b). Based on the replication the latency is improved from 5.38 seconds to 2.25 seconds, an improvement of 58%.

3 Performance Studies

To evaluate the performance of the proposed strategies, the OptorSim [10] simulator was used, which is a commonly used simulator for the simulation of Data Grid environment. The simulation runs are taken with jobs arriving average exponential inter-arrival time of 0.25 seconds, the processing speed at the nodes are considered constant at 10 second/Gb of data. Number of jobs requesting a particular file is distributed exponentially. This gives an elliptical file distribution per job with an average of 7 and total files in the system (ϕ) as 20. The initial file

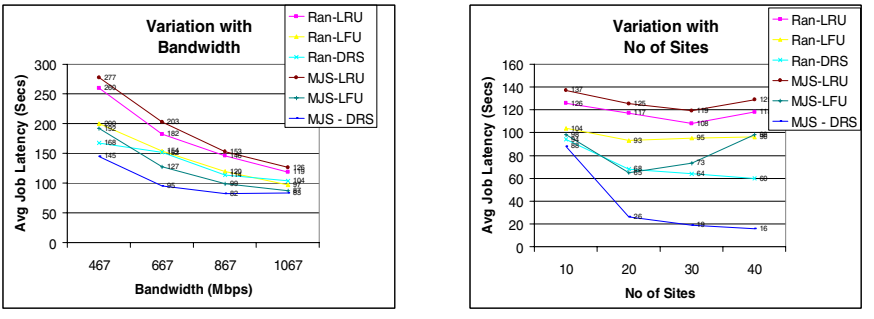

Fig. 3. Performance of schemes with (a) Bandwidth variation, (b) No. of Sites

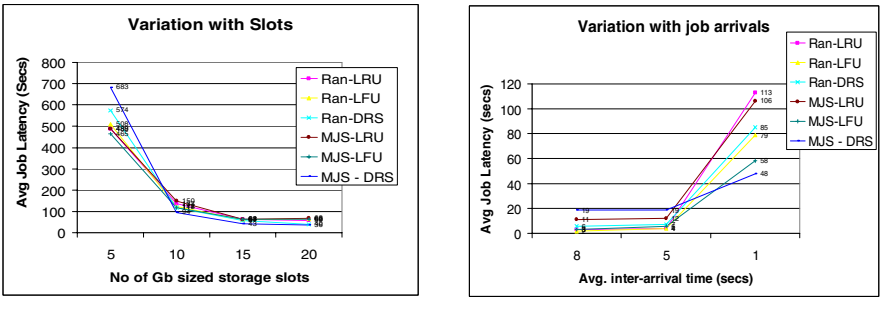

Fig. 4. Performance of schemes with (a) no. of slots, (b) Job Arrival

distribution in the Grid is random. The data replication was carried out at pre-determined intervals during the runs. The parameters that were varied in simulation runs are network bandwidth (467 to 1067 Mbps); number of computing and storage nodes (10 to 40); and storage capacity V_j (5 to 20 Gb). A separate set of runs were taken by varying the average job inter-arrival times (1 seconds to 8 seconds). We have compared performance of our data replication strategy against no replication, and commonly used Least Recently Used (LRU), Least Frequently Used (LFU) replication strategies. For job scheduling, the MJS version of our

scheduling algorithm is used (with α value in the average queue size factor taken as 1). Essentially, during evaluation, the following combinations are considered : Ran-NoRep – random scheduling with no replication, Ran-LRU – random scheduling with LRU replication strategy, RAN- LFU – random with LFU, Ran-DRS – random with our replication strategy, MJS-NoRep – our scheduling with no replication, MJS-LRU,MJS-LFU and MJS-DRS respectively.

3.1 Discussions

Figures 3 and 4 highlight the performance of the schemes in terms of variations in average job latency with respect to the variations in bandwidth, number of sites, SE size, and job arrival distributions. It is clear that the MJS scheduling scheme with DRS performs best in most of the cases. The cases with no data replication strategies perform far worse than other cases, and therefore, the results are omitted. For example, average job latency with Ran-NoRep strategy with 10 sites is 1512 seconds – 10 to 12 times the average job latencies observed in other schemes.

The idle times for the processors were evenly distributed at high loads following MJS-DRS scheme with a standard deviation of 3. The average job queue sizes at high loads were also evenly distributed. The results show that with variation of bandwidth (figure 3(a)), the Ran-DRS scheme achieves a performance improvement of up-to 19% over Ran-LFU and up-to 54% over Ran-LRU schemes. The MJS-DRS scheme results in a job latency improvement of an average of 56% over Random scheduling schemes and of 23% over other data replication schemes with our scheduling. With increasing numbers of computing elements *n,* the MJS-DRS scheme performs significantly better than other schemes (figure 3(b)). At 40 nodes the MJS-DRS scheme is almost two times better than Ran-DRS scheme and even better than other replication schemes. The results of the job latency with variations of storage capacity, and hence, the number of gigabyte slots are interesting (figure 4(a)). While at lower storage capacity (5 GB), other schemes perform marginally better than MJS-DRS; with increase in capacity and γ_j , MJS-DRS fares better than other schemes by an average of 17%. As evident from figure 4(b), the MJS-DRS scheme scales up better than other schemes with increase in load. With a decrease of average job inter-arrival time from 5 seconds to 1 seconds, the average job latency increases by 2.3 times while in case of the other schemes the performance deteriorates by 10-12 times.

We have also compared our scheduling – replication scheme against economy based replication strategy proposed in [11]. The preliminary results, as given in Table 3, suggest that the MJS-DRS scheme improves job latency over EcoModel optimizer (EO) with ZipF file distribution.

Table 3. Performance comparison of MJS-DRS with EcoModel optimizer

Scheduler + Replication	RS+EO	RS+DRS	MJS+EO	MJS +DRS
Job Latency (secs)	106	104	69	48

4 Conclusions

In this paper an interaction between replication and scheduling strategy called the Integrated Replication and Scheduling (IRS) strategy, has been proposed. The data replication is carried out in an asynchronous timer-controlled process that takes into account history of jobs and data access patterns and is primarily based on the notion of expected data file latency and a greedy optimization approach. The scheduling is carried out in a matching-based or a cost-based manner with view of transient system-state data like queue length. The approach MJS-DRS has shown promising results with respect to the popular and commonly used data replication strategy, while the cost-based scheduling approach is yet to be tested. Contrary to [8], our experience shows that it is better to consider the interactions of replication and scheduling while scheduling in a Data Grid and replication strategy works well even if the jobs are not scheduled locally. In this paper, we have considered a centralized external scheduler which may prove costly with increase in Data Grid size. In subsequent works, we propose to extend this scheduling and replication scheme to a decentralized and hierarchical environment. Further, we propose to analyze the sensitivity of the schemes with respect to variations *viz* in file sizes, processor speeds, effect of data arrival pattern in job execution etc.

References

1. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, vol. 23, pp. 187-200, 2001.
2. M. Beck and T. Moore, "The Internet2 distributed storage infrastructure project: An architecture for internet content channels," *Computer Networking and ISDN Systems*,1998.
3. Foster and C. Kasselman, "The Grid 2: Blueprint for a new Computing Infrastructure," *Morgan Kaufman,* 2004.
4. H. Casanova, G. Obertelli, F. Berman and R. Wolski, "The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid," in *Proc. SuperComputing'00*, 2000.
5. C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert, "Scheduling Strategies for Master-Slave tasking for Heterogeneous Processor Platforms," in *IEEE Trans. On Parallel and Distributed Systems,* vol. 15, no. 4, Apr. 2004.
6. K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid," in *Proc. Second IWGC*, 2001.
7. D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau and M. Livny, "Gathering at the Well: Creating Communities for Grid I/O," in *Proc. SuperComputing 2001*, 2001.
8. K. Ranganathan and I. Foster, "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids," in *Journal of Grid Computing*, vol. 1, no. 2, Apr. 2003.
9. R.R. Mettu and K.G. Plaxton, "The Online Median Problem", in *SIAM Journal on Computing, Vol. 32, No. 3,* pp 816- 832, 2003

10. W.H. Bell, D.G. Cameron et al., "Simulation of Dynamic Grid Replication Strategies in OptorSim," in *Proc. Third Int'l Workshop on Grid Computing*, 2002.
11. W.H. Bell, D.G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, K. Stockinger and F. Zini, "Evaluation of an Economy-Based File Replication Strategy for a Data Grid", in *Proc. CCGrid*, May 2003.