Journal of Grid Computing **2:** 57–69, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim

D. G. Cameron[1], A. P. Millar[1], C. Nicholson[1], R. Carvajal-Schiaffino[2], K. Stockinger[3] and F. Zini[2]
[1]*University of Glasgow, Glasgow, G12 8QQ, Scotland, UK*
[2]*ITC-irst, Via Sommarive 18, 38050 Povo (Trento), Italy*
[3]*CERN, European Organization for Nuclear Research, 1211 Geneva, Switzerland*

Key words: data grid, data replication, optimisation, scheduling, simulator

Abstract

Many current international scientific projects are based on large scale applications that are both computationally complex and require the management of large amounts of distributed data. Grid computing is fast emerging as the solution to the problems posed by these applications. To evaluate the impact of resource optimisation algorithms, simulation of the Grid environment can be used to achieve important performance results before any algorithms are deployed on the Grid. In this paper, we study the effects of various job scheduling and data replication strategies and compare them in a variety of Grid scenarios using several performance metrics. We use the Grid simulator OptorSim, and base our simulations on a world-wide Grid testbed for data intensive high energy physics experiments.

Our results show that scheduling algorithms which take into account both the file access cost of jobs and the workload of computing resources are the most effective at optimising computing and storage resources as well as improving the job throughput. The results also show that, in most cases, the economy-based replication strategies which we have developed improve the Grid performance under changing network loads.

1. Introduction

Data Grids are predicted to be the solution to the large computational power and data storage requirements of many current projects such as the next generation of high energy physics experiments. These Data Grids will enable sharing of distributed computational and storage resources among users located all over the world. An important challenge to undertake during the construction of Data Grids is the problem of how to optimise the use of available resources.

Efficient job scheduling, i.e. the decision of when and where to run jobs submitted to the Data Grid, is important to ensure that these resources are neither over- nor under-used. Data replication – the process of creating identical copies of data files at different sites – is also an important part of maximising job throughput in a typical Data Grid. The task of *Replica Optimisation* strategies is to create, select and delete replicas taking into account both the access patterns (work loads) and the resource consumption of the jobs that are scheduled on the Grid. The combined use of scheduling and replica optimisation strategies should lead to optimal use of computational, data storage and network resources.

The European DataGrid project [19] has built computing infrastructure and middleware services for the management of large-scale data across widely distributed scientific communities. Within this project a *Resource Broker* (RB) was designed to handle job scheduling decisions and a *Replica Optimisation Ser-*

vice (ROS) [3] was developed to address the issues related to Replica Optimisation.

An efficient RB and ROS should use an optimisation strategy that must work effectively in a Data Grid under a wide range of conditions, so potential strategies should be thoroughly tested before they are employed. One way to achieve a realistic evaluation of various strategies is to define a Grid simulation environment that closely mimics a real Data Grid. This environment should be capable of simulating a number of Grid jobs using a candidate optimisation strategy, and to collect measurements on which the evaluation of the strategy is based. The authors have developed the Grid simulator OptorSim [4, 23], which has been used to evaluate several Grid Replica Optimisation strategies; in particular, an auction protocol and economic model were introduced [5] and it was shown that for certain file access patterns this model significantly outperforms more traditional models. Some early simulation results were used to select the optimisation strategy embedded in the ROS deployed in the European DataGrid project.

In this paper we are interested in continuing experimentation on optimisation strategies. We first discuss the key elements of a realistic Grid model, which forms the basis of our simulation environment. With respect to our previous work, we first consider the effects of different job scheduling strategies and improve the accuracy of the model by taking into account background (i.e. non-Grid) network traffic, which can use a sizeable proportion of the underlying network resources. Second, we analyse some metrics which are useful for the evaluation of a Data Grid. In previous experiments [5], the metric used was the total execution time of a set of jobs on the Grid. Here, we add other indicators of performance that are significant for different Grid users. Third, we present the performance of a number of scheduling and replication strategies in different operating scenarios [25].

The paper is structured as follows. Related work on Grid simulations is examined in Section 2. We discuss the features of a realistic simulation environment and briefly present the main features of OptorSim in Section 3. Section 4 describes our Scheduling and Replica Optimisation strategies in detail, and a set of performance measurements to evaluate these strategies is presented in Section 5. The specific setup we use for our simulations is described in Section 6, with results presented in Section 7. Finally, we draw some conclusions and present ideas for future work in Section 8.

2. Related Work

Various Grid simulation projects have been undertaken in recent years, among them ChicagoSim [16, 17], EDGSim [9], GridSim [7], and GridNet [11].

In [16] the authors simulated various replication and caching strategies in a tier-model Grid environment. In [17] they combined replication strategies with different scheduling algorithms and found that when using any replication strategy, taking the location of data into account when scheduling vastly improves the overall job performance. They found, however, that there was no marked difference in the choice of replication algorithm, perhaps because replication took place at the level of entire file sets (one file set defining a job) rather than individual files.

EDGSim [9] was designed to simulate the performance of the European DataGrid but concentrates on the optimisation of scheduling algorithms. Analysis showed that data location was important in the scheduling decision, but no replication of data was taken into account.

The GridSim project [7] produced a very detailed simulation of the components of a Grid and introduced an economic model to manage the use of Grid resources through the buying and selling of resources. It was designed primarily to study scheduling algorithms and did not examine the issue of data replication.

In [11] a replication algorithm was tested which uses a cost function to predict whether replicas are worth creating. It was found to be more effective in reducing average job time than the case where there was no replication. The simulation architecture used was based on a hierarchical model where leaf client nodes ran jobs but higher nodes contained all the storage resources, in contrast to the European DataGrid architecture which we describe in Section 3.

The main advantage of OptorSim with respect to the previous simulators is that it performs two-stage optimisation. Scheduling decisions are based on both the location of data and the status of network links between grid sites, while (re)optimisation during the run-time of a job takes into account dynamic variations in the distribution of data and in the behaviour of network resources.

3. A Data Grid Simulation Environment

A realistic Grid simulation environment should be based on a Grid model that represents a real Data Grid

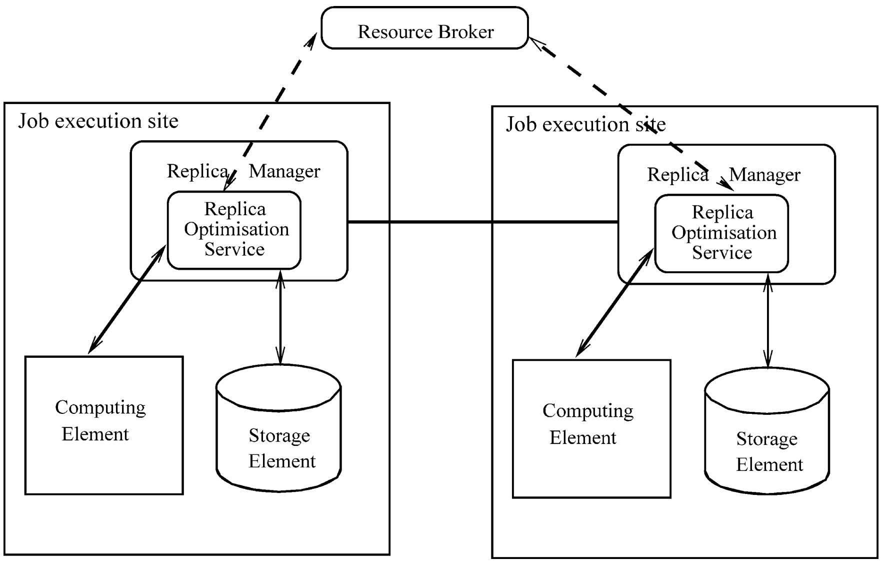


Figure 1. European Data Grid Architecture.

at the proper level of abstraction. This is the case for our model, which includes the elements for describing the Grid topology and site structure, the set of jobs to be simulated, and the parameters that regulate the dynamics of the simulation. In this section we examine its various components.

3.1. *Components of the Simulation Model*

3.1.1. *Grid Architecture*

We adopt a Grid structure based on a simplification of the architecture proposed by the European DataGrid project [19], as illustrated in Figure 1. In this model, the Grid consists of several *sites*, each of which may provide computational and data-storage resources for submitted jobs. Each site may contain a *Computing Element* (CE) and/or *Storage Element* (SE), with sites that have no Storage or Computing Elements acting as network nodes or routers. Computing Elements run jobs that use the data in files stored on Storage Elements. Our focus is on overall optimisation of Grid resources rather than performing intra-site optimisation, so in our model we assume simplified Computing Elements that have no internal structure. For instance, we do not consider scheduling of jobs to worker nodes within a Computing Element or internal queuing systems for jobs.

A *Resource Broker* controls the scheduling of jobs to Computing Elements, with the aim of improving the overall throughput of the Grid. Grid sites are connected by *Network Links*, each of which has a certain bandwidth. A *Replica Manager* at each site manages the data flow between sites and interfaces between the computing and storage resources and the Grid. The *Replica Optimisation Service* [3] inside the Replica Manager is responsible for both replica selection and the automatic creation and deletion of replicas.

3.1.2. *Files and Jobs*

In our model a data file is characterised by its name and size. A job is specified by the set of data files it needs to analyse. Files are considered to be homogeneous and only the movement of files caused by replication is simulated, not the analysis processes performed by the jobs. We also model the initial distribution of files to Data Grid sites before starting a simulation run.

3.1.3. *Dynamics*

The order in which a job requests files is determined by the *Access Pattern* used. Several access patterns are possible, owing to the different types of Grid jobs that may be run. Another important aspect is background network traffic, which can vary unpredictably over time. Any optimisation strategy must be sufficiently flexible that it can adapt to the continually changing environment, and still obtain the best performance for its users.

3.2. *The Grid Simulator OptorSim*

To evaluate optimisation strategies in a realistic simulation scenario, we have developed the Grid simulator

OptorSim. Given (a) a Grid topology and resources, (b) a set of jobs that the Grid must execute and (c) an optimisation strategy, OptorSim simulates what would happen in the Grid if that optimisation strategy were in use. It provides us with the set of measurements described in Section 5 in order to quantify the effectiveness of the strategy.

OptorSim is written in Java™ and it is designed to follow the model described in Section 3.1, using a number of threads that mimic the components shown in Figure 1. It also models peer-to-peer communication between Grid sites. This is used by the auction protocol for file selection, which we presented in [5].

OptorSim adopts time-based simulation. As we are interested in evaluating replication strategies, we simulate data transfer among Grid sites and neglect their processing by jobs. The total time needed for the simulation of a Grid scenario is proportional to the time that data transfer would take in a real Grid. This means that any time that would be spent performing optimisation calculations in the real Grid is automatically taken into account in the simulation. The adoption of a time-based simulation certainly has the disadvantage of a longer simulation time with respect to an event-driven model, especially if we want to simulate the execution of thousands of jobs on large-scale Grids; however, we are able to reduce simulation time by scaling down certain parameters, which decreases the simulated data transfer time.

4. Scheduling and Replica Optimisation Strategies

In this paper we will consider two stages of optimisation, namely *Scheduling Optimisation* (deciding where a job should be executed) and runtime *Replica Optimisation* (deciding which is the best replica for a file requested by a running job and how best to position the data).

4.1. *Scheduling Strategies*

In the past, much work has been done on traditional CPU-based scheduling algorithms in a Grid (see Section 2). In this paper we evaluate scheduling algorithms that take into account both computational and data resources for selecting the best job location. A scheduling algorithm calculates the cost of running a job on each site from a group of candidate sites. It then submits the job to the site with the minimum estimated cost. The algorithms we test are based on one or more of the following cost metrics:

- *Access Cost.* The estimated cost, based on the current network status, for obtaining all the files required by the job. This metric uses a *Replica Catalogue* (a Grid service, implemented in OptorSim as a table containing the locations of every copy of every file) to look up all the replicas for each required file. The access time for each replica is calculated and thus the best replica can be found for each file. The combined access time for the best replicas is used to rank candidate sites.
- *Queue Size.* The number of jobs waiting in the queue at the candidate site. We assume only one job at a time can run on each CE.
- *Queue Access Cost.* For each job in the queue the access cost is calculated as for the *Access Cost* algorithm. The access costs for all jobs are summed to give a total estimated access cost for all the jobs in the queue.

4.2. *Replica Optimisation Strategies*

We assume that Replica Optimisation is performed in a distributed way by a number of *Replica Optimisation Agents* (or *Optimisers*), one for each Grid site. An Optimiser performs local Replica Optimisation; the aim is to achieve global optimisation as the emergent result of local optimisation and every Optimiser therefore has two goals:

- *To minimise a single job's execution cost.* Users want their jobs to be executed at as low a cost as possible; an Optimiser therefore aims to minimise the execution cost of every job that is run on its Grid site. In this paper we define the cost of executing a job to be the total running time of the job.
- *To maximise the usefulness of locally stored files.* Good utilisation of available data resources is another goal of optimisation. An Optimiser should also, therefore, aim to keep locally those files that are most useful for jobs that are executed either locally or at neighbouring sites with good network connectivity. This also reduces the running times and hence the costs of running jobs.

Whenever an Optimiser is considering a file request, it performs the following tasks in order to achieve the goals above:

- *Replication Decision.* If a requested file is not present on a site's SE, this process decides whether

local replication of this file should take place. If the Optimiser decides not to replicate a file then the job must access that file remotely.
- *Replica Selection.* When considering which replica to read or replicate locally, this process selects the best of those available. In general, the selection criterion depends on the chosen evaluation metric.
- *File Replacement.* When a remote replica has been selected for replication to the site's SE, the SE might not have sufficient spare capacity. In this case, one or more local replicas must be deleted. The selection criteria for deciding which locally stored replicas to delete depend on some estimate of future usefulness.

A specific combination of algorithms for each stage defines a runtime replica optimisation strategy. We consider three specific optimisation strategies: one based on the traditional LFU (Least Frequently Used) algorithm, and two economic strategies.

The LFU-based strategy will always replicate files to storage local to the Computing Element on which the job is running. Replica Selection is achieved using a Replica Catalogue to locate all replicas. The replica that can be accessed in the shortest time under the current network conditions is chosen. If the local storage is full, the file that has been accessed the fewest times in the preceding period of time is deleted, creating space for the new replica.

The two economy-based strategies are similar to each other, but use two different prediction functions, one binomial-based [8] and the other Zipf-based [25], to calculate file values used in the replication and file replacement decisions. If the potential replica under consideration has a higher value than the lowest-valued file currently in the local storage, that file is deleted and the new replica is 'bought.' If local storage is not yet full, the economic models will always replicate.

The file value is approximated by the number of times it is expected to be accessed in a future time window $\delta T'$, based on the file access history for the previous time window δT. The binomial prediction function constructs a binomial distribution of file popularity, centred on the mean number of file accesses in δT. The value of the file in question is then found by checking where it lies on that distribution. This prediction function is described in more detail in [8]. The Zipf prediction function orders the files into a Zipf distribution according to their popularity in δT, and takes the value from there. A description of the Zipf distribution function is given in Section 6.

Replica Selection is based on the auction protocol for buying and selling files, as described in [5]. By means of the auction protocol, file requests are propagated, using a peer-to-peer infrastructure, over the neighbourhood of the site from which the file request originates. If the file request reaches a site where the file is available locally, the Optimiser at that site will calculate the transfer cost to the requesting site and reply with a corresponding bid. If the file is not present on the site, the Optimiser might start a *nested auction* in order to create a replica of the requested file locally. Once this auction is complete the site can reply with a bid to the initial auction. When all bids have been collected, the winner is the site which bid the lowest price, and it is paid the price of the second lowest bid. This is known as a reverse Vickrey auction [22].

5. Grid Performance Metrics

In order to evaluate the effectiveness of the various optimisation strategies implemented in OptorSim, we have identified the following metrics:
- Mean job execution time;
- Network usage;
- SE usage;
- CE usage.

5.1. *Mean Job Execution Time*

The mean job execution time is defined as the total time to execute all the jobs, divided by the number of jobs completed. This is related to the metric we used in [4, 5] and a typical Grid user would probably consider it to be the most important metric of how the algorithm is performing.

5.2. *Network Usage*

Replicating a file takes time and uses network bandwidth. However, performing no replication has been shown [4] to be ineffective compared to even the simplest replicating optimsation algorithm. Thus, a good balance must be found, where any replication is in the interest of reducing future network traffic. We define *effective network usage* r_{ENU}:

$$r_{\mathrm{ENU}} = \frac{N_{\text{remote file accesses}} + N_{\text{file replications}}}{N_{\text{local file accesses}}},$$

where $N_{\text{remote file accesses}}$ is the number of times the CE reads a file from a SE on a different site, $N_{\text{file replications}}$ is the total number of file replications, and $N_{\text{local file accesses}}$ is the number of times a CE reads a file from a SE on the same site (we assume infinite bandwidth within a site).

For a given network topology, a lower value of r_{ENU} indicates that more files are accessed locally, fewer network resources are used and hence the optimisation strategy is better at replicating files to the correct location.

5.3. *SE Usage*

Monitoring the use of storage resources in Grid sites can also be a valuable source of information and thus we measure the percentage of storage used during the simulation. This can help in evaluating a strategy from two opposite points of view: on the one hand, the goal could be the minimisation of storage usage, perhaps because the resource cost is proportional to the amount being used; on the other hand, its cost might be fixed and one would then aim at maximising the use of storage space.

5.4. *CE Usage*

Another resource which is of interest is the computational power usage, which we define as the percentage of time that a CE is running jobs or otherwise active. The metric used in this paper is the total computational power usage for all the CEs on the Grid, which we call the CE usage. A good scheduler should be able to maximise the CE usage by spreading the workload, avoiding the situation where some sites lie idle while others have long queues of jobs.

6. Simulation Setup

In this section we describe the topology of a realistic Grid environment. We also provide a description of the simulated jobs and discuss various access patterns (work loads) implemented in OptorSim.

6.1. *Grid Topology*

In this paper we base all simulation studies on the testbed used during a large scale production effort for the

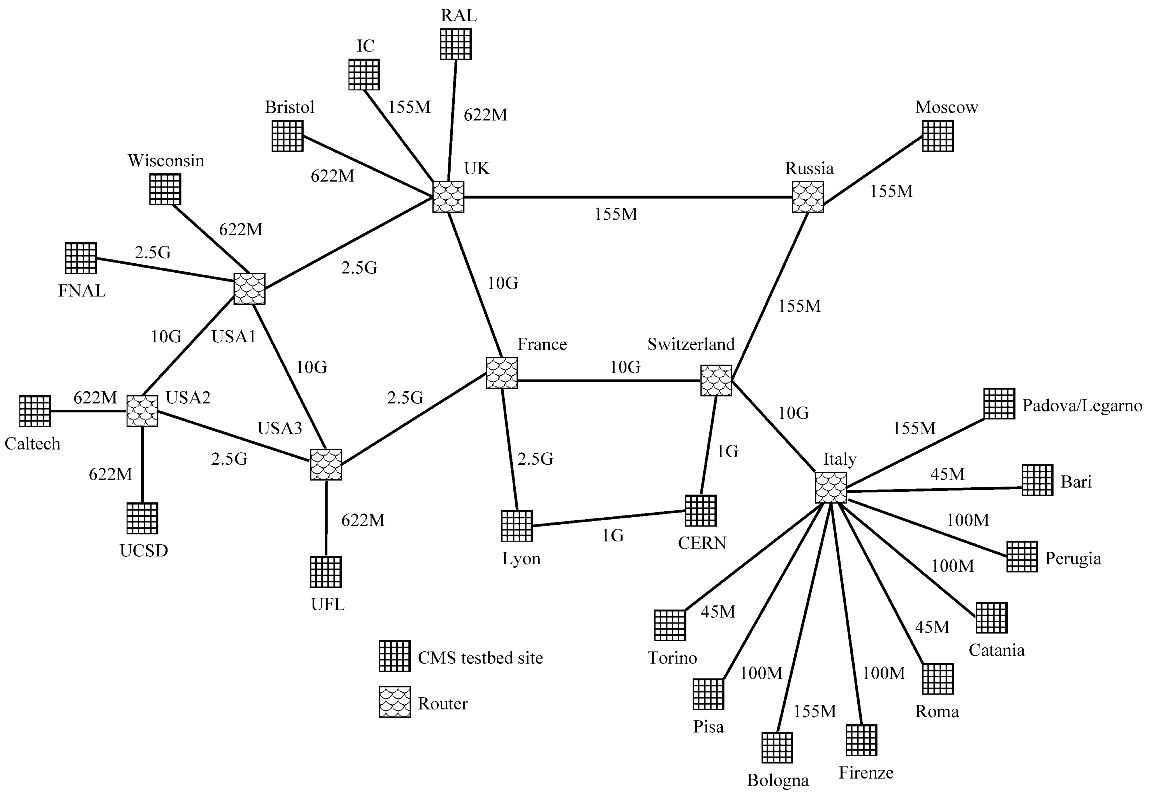


Figure 2. Grid topology for CMS data production challenge in Spring 2002.

high energy physics experiment CMS [12]. We used a similar testbed configuration in our previous work [5]. The Grid topology (see Figure 2) comprises 20 sites in Europe and the USA. CERN and FNAL (where the data are originally produced) have a storage capacity of 100 GB each and a master copy of each file is stored at one of these sites. Every other site has a Computing Element and initially empty storage of capacity 50 GB. The storage capacity values used are representative, being scaled down from the actual resources at these sites. In Figure 2, the labels over network links represent the available bandwidth in Gb/sec or Mb/sec. During our simulation studies we include contending network traffic which is based on network monitoring data between some of the sites shown in Figure 2 (see Section 7.3).

6.2. *Simulated Jobs*

The simulated work loads are based on a scaled down set of high energy physics analysis jobs from the CDF experiment use case (as described in [10]). In this case each file has a size of 1 GB and the total size of the whole file set on the Grid is 97 GB. Each CE takes one second to process each file.

6.3. *Access Patterns*

Access patterns determine the order in which files for a particular job are requested by a CE. In this paper we will consider the following access patterns: *sequential* [4] (all files are requested in a predetermined order), *Gaussian random walk* [4] (successive files are selected from a Gaussian distribution centered on the previous file) and *Zipf.*

A Zipf-like distribution, which is an inverse power law distribution, is defined as $P_i \propto i^{-\alpha}$, where P_i is the frequency of occurrence of the ith ranked item and $\alpha \leqslant 1$. In other words, a few items in the observed set occur very often while many others occur rarely. Due to the increasing importance of the web as an Internet application, recent research [1, 2] has investigated how to model and reproduce typical web workloads and shown that they generally follow a Zipf-like distribution. Web proxy caching techniques especially have received considerable interest due to the importance of reducing web traffic [6]. As there is a conceptual similarity between web workloads and Data Grid file access patterns, we have decided to investigate the effects of such a distribution in a Grid environment. Values of α are typically between 0.7 and 1; for the results presented in this paper, a value of 0.85 was used.

7. Results

In this section we present simulation results, using OptorSim to evaluate and compare different strategies for both scheduling and replica optimisation. We use the metrics described in Section 5 as indicators of how well the strategies perform. The simulation was run on a farm of dual processor Pentium IIIs, taking the average of several simulation runs for each set of results.

7.1. *Scheduling and Replica Optimisation Strategies*

We start our evaluation by studying the impact of the scheduling algorithm used by the Resource Broker on a given replica optimisation strategy. The following scheduling algorithms are analysed (see Section 4.1):

– *Random:* schedule jobs to a random CE;
– *Shortest Queue:* schedule to the CE with the shortest job queue;
– *Access Cost:* schedule to the CE where the job has lowest file access cost;
– *Queue Access Cost:* schedule to the CE where the sum of the access cost for the job itself and the access costs of all jobs in the queue is lowest.

The simulation was run with 1000 jobs submitted at 5 second intervals and for each scheduling algorithm, each of the three access patterns described previously (*sequential*, *Gaussian random walk* and *Zipf*) was considered.

Figures 3, 4, and 5 show the mean job execution time and CE usage for the three different access patterns.

In general the three access patterns show very similar relative performance for each scheduling and replication strategy. The mean job time for Gaussian random walk access patterns, however, is roughly half that for sequential access patterns and for Zipf access patterns it is half as much again. This is because with sequential access patterns, every file is accessed once whereas for the other two, some files may be accessed more than once and others not at all. The access pattern has very little effect on the CE usage, but with Zipf access patterns it is around 10% higher than with the other two.

The scheduling strategies *Random* and *Shortest Queue* show similar performance and generally have

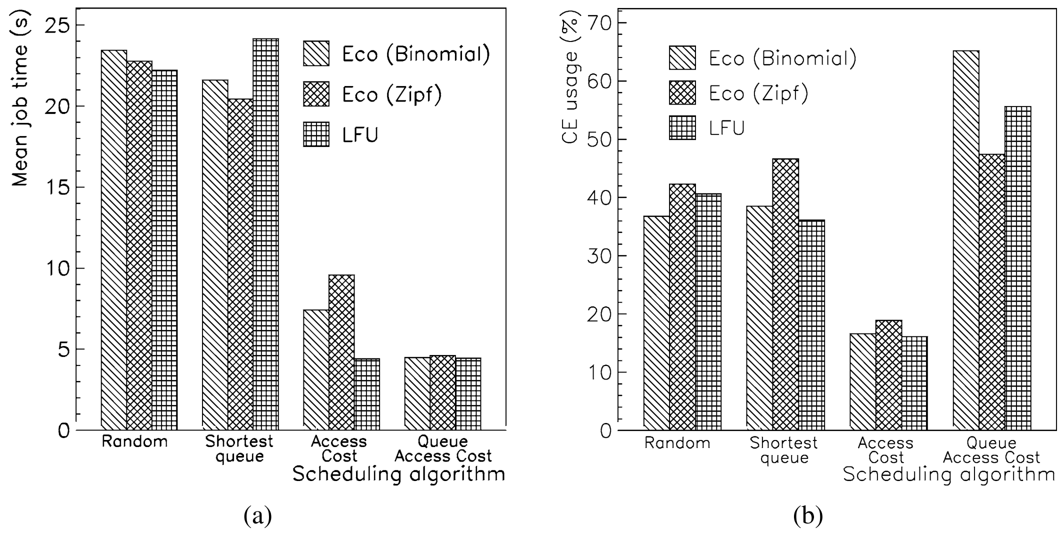


Figure 3. (a) Mean job time and (b) CE usage for various replica optimisation strategies and sequential access pattern.

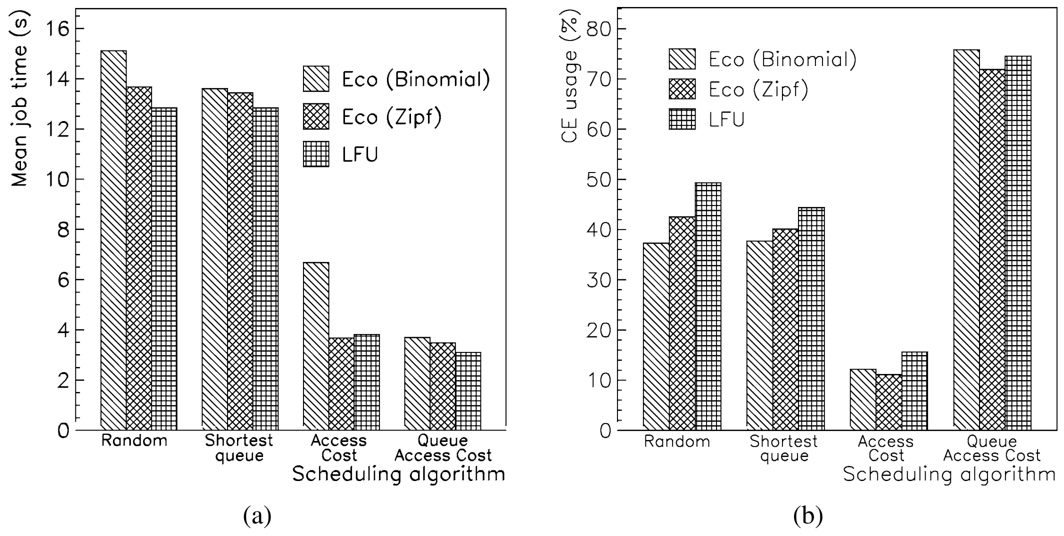


Figure 4. (a) Mean job time and (b) CE usage for various replica optimisation strategies and Gaussian random walk access pattern.

the longest mean job execution times. The scheduling strategy *Access Cost* has a lower mean job execution time but has the lowest CE usage, due to the fact that jobs are only scheduled to sites with high network connectivity.

The mean job execution time is lowest and CE usage is highest when we use the scheduling strategy *Queue Access Cost.* This algorithm has the tendency to schedule data intensive jobs close to the location of the data, whilst ensuring that sites with high network connectivity are not overloaded and sites with poor connectivity are not idle.

The SE usage was also monitored as the simulation progressed, and the same scheduling strategies evaluated with the three replica optimisation strategies. Sequential access patterns were used. Figure 6(a) shows that the scheduling strategies *Random* and *Shortest Queue* quickly fill up all the available SEs to reach the maximum of 90% (100% SE usage is never reached due to the Grid configuration used, in which CERN and FNAL serve as data repositories for master files, to which no jobs are sent and which thus remain unaffected by data replication). The strategy *Access Cost* uses only the sites with high network connectivity, re-

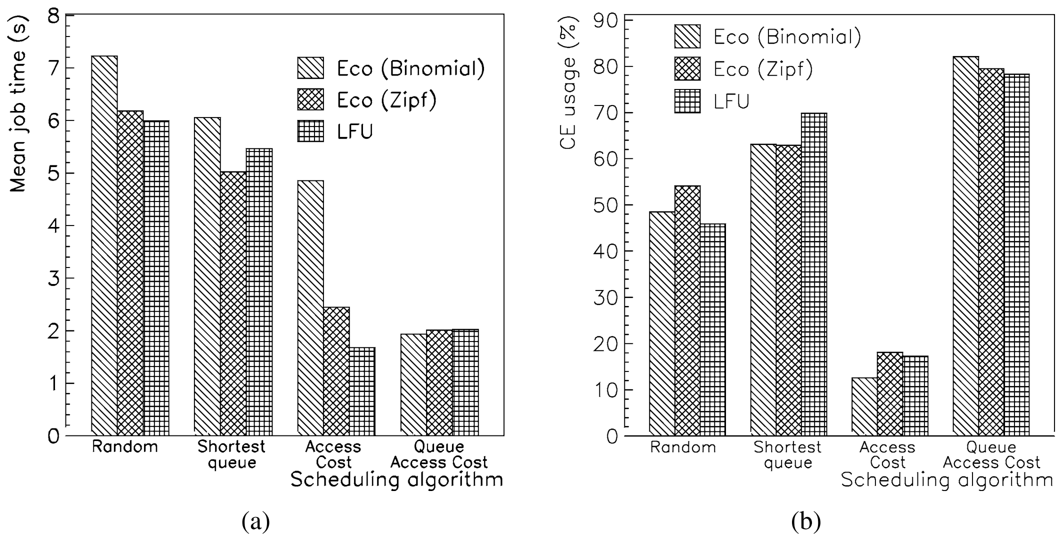


Figure 5. (a) Mean job time and (b) CE usage for various replica optimisation strategies and Zipf access pattern.

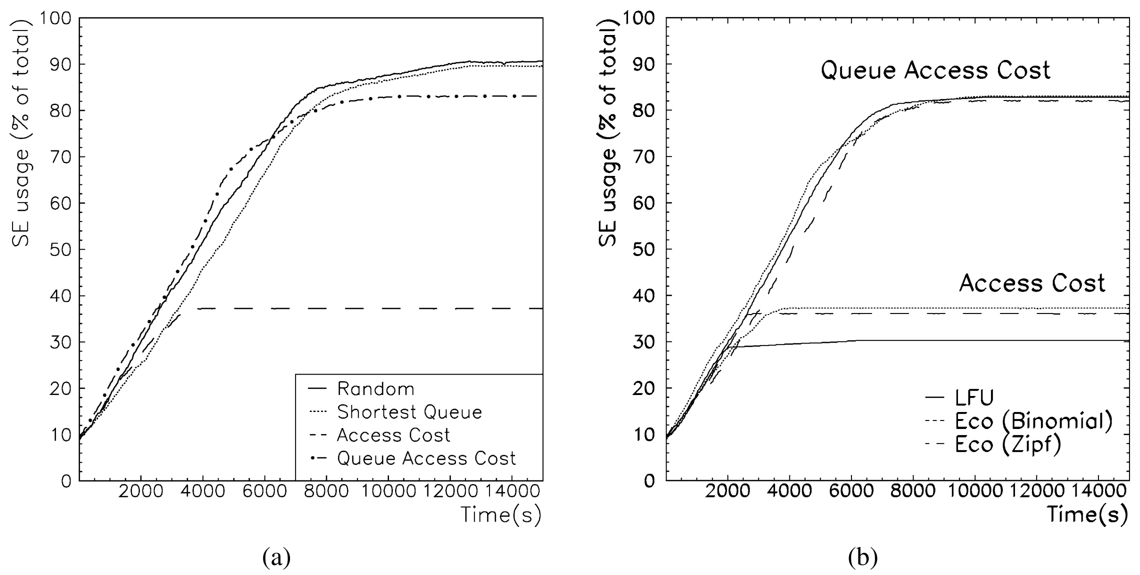


Figure 6. SE usage: (a) all schedulers, binomial economic model; (b) all replica optimisation strategies, *Queue Access Cost* and *Access Cost* schedulers.

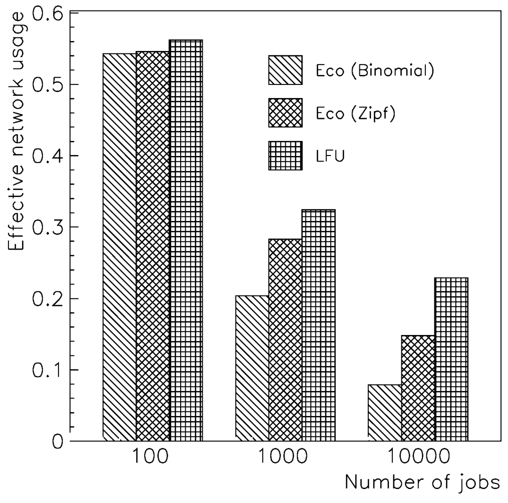
sulting in slower execution time and a final SE usage level of only 37%. *Queue Access Cost*, on the other hand, takes network connections into account and also avoids long queues of jobs, resulting in good SE usage and fast execution time.

All Replica Optimisation strategies gave very similar results, as can be seen for two of the schedulers in Figure 6(b). The difference in SE usage between these two schedulers is comparable to the difference in CE usage shown above.

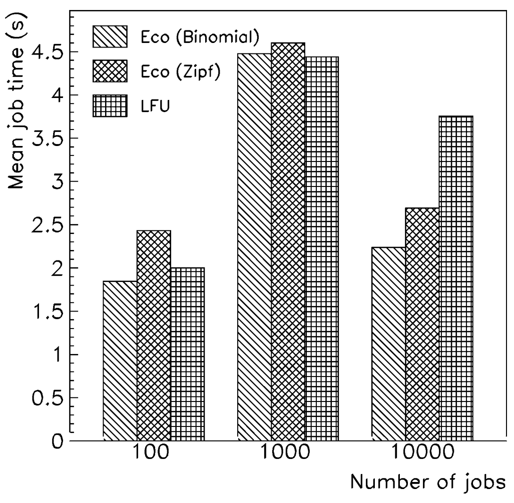
Given that the *Queue Access Cost* scheduling algorithm had given the best results in all the experiments above, it was chosen for all further tests, with sequential access patterns taken to be the closest available approximation to physics analysis jobs.

7.2. *Scalability of Replica Optimisation Algorithms*

In the next set of tests we study the scalability of the optimisation algorithms by varying the number of jobs from 100 to 1,000 to 10,000. To set a scale for these



(a)



(b)

Figure 7. Effective network usage (a) and mean job time (b) for different number of submitted jobs.

tests, the number of jobs required to fill the SEs to ∼75% was measured. Using the binomial economic model, sequential access patterns and *Queue Access Cost* scheduler, it was found that this level was reached after ∼ 500 jobs had been completed and hence that the above range was reasonable.

The effective network usage for 100, 1,000 and 10,000 jobs, shown in Figure 7(a), decreases with the number of jobs submitted. This is as might be expected, since the access histories used by the optimisation strategies to make replication decisions take time to build up and stabilise. The economic strategies, though, show much lower usage with an increased number of jobs, with a factor of 3 difference between the binomial based economic model and the LFU strategy. In short, the main advantage of the economic strategies is that they use up considerably less network bandwidth than the LFU strategy.

This scalability can also be seen in the mean job times (Figure 7(b)), with the economic strategies becoming more effective with an increased job load. From 1000 jobs to 10000 jobs, the mean job time for the binomial based economic strategy fell by 2 s and for the Zipf based economic strategy by 1.2 s, whereas for the LFU strategy it only fell by 0.6 s.

7.3. *Effects of Non-grid Network Traffic*

In all the previous evaluations, non-Grid background traffic was included in the network model. This non-Grid background traffic consists of any data transfers that can be observed on the network throughout the day; here, we examine the effect this has on Grid performance by comparing results with and without the inclusion of background.

To build up a profile of the underlying network traffic for the testbed links, Iperf [20] monitoring data from various sources was used, including EDG WP7 [24], SLAC [14] and FNAL [18] WAN Bandwidth Measurement Tests, UK e-Science Grid Network Monitoring [21] and GridNM [13]. The sizes of the data samples varied from a period of a few days to about two months. The mean value for each half-hour period of the day was found and a profile of mean available bandwidth as a function of the time of day compiled for each link for which data was available. Where data for a particular link was not available, it was substituted by using data from as similar a link as possible, e.g., a site on the same router.

A selection of these profiles is shown in Figure 8, with the uncertainty in the mean for each point; it can be seen that some links exhibit a clear diurnal variation in the available bandwidth. There is also a large variation in the actual bandwidth available from link to link. The data are stored as fractional values of the maximum bandwidth on each link, so when it is necessary to calculate a bandwidth, the mean value for the given time and link is taken, a random jitter is added and the maximum is scaled by the result.

A comparison of the results with and without background traffic is shown in Figure 9. As would be expected, there is a large increase in mean job time when we simulate the background network traffic. For

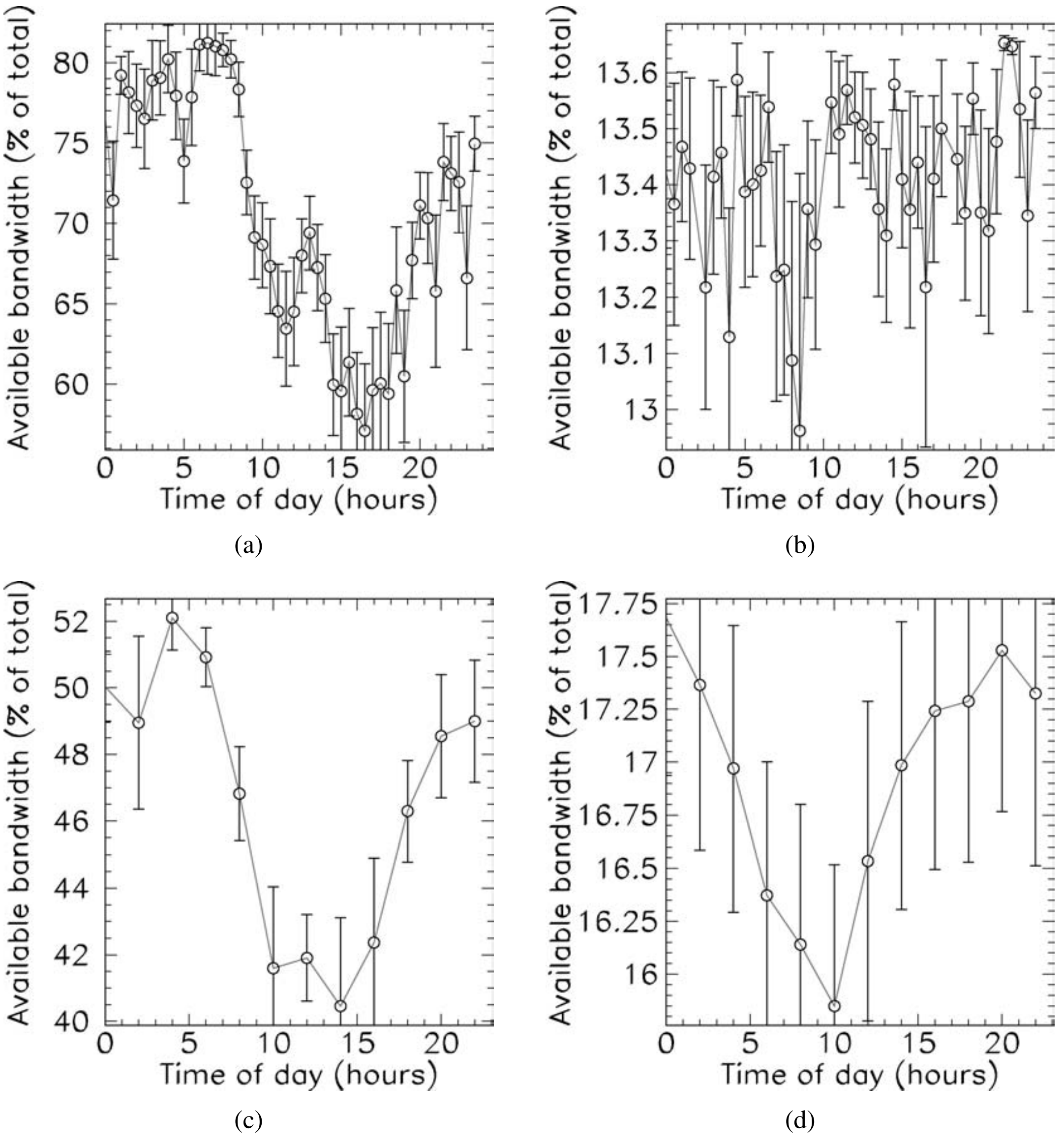


Figure 8. Mean available bandwidth on some of the monitored links: (a) Lyon – CERN; (b) Lyon – Bologna; (c) SLAC – Florida; (d) FNAL – IC.

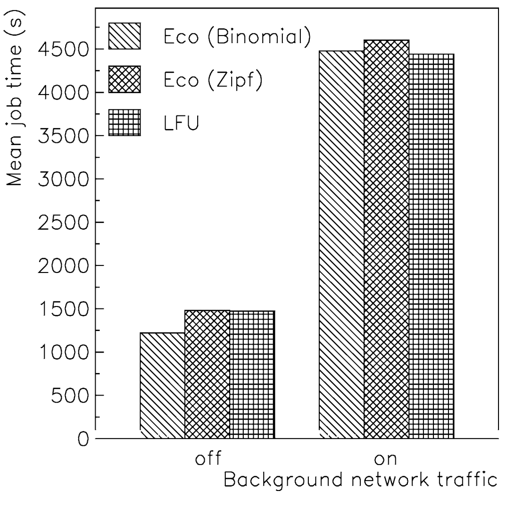
all the optimisation strategies, this increase is around a factor of 3.

There is also a big increase in the effective network usage for the binomial-based economic strategy and LFU strategy, while for the Zipf-based economic strategy it remains roughly constant. This is perhaps due to the changing network bandwidths leading to less reliable replication decisions by the optimisation strategies, which in the long term means that more replication takes place – except in the Zipf-based economic strategy, which seems to be the most stable to fluctuations.

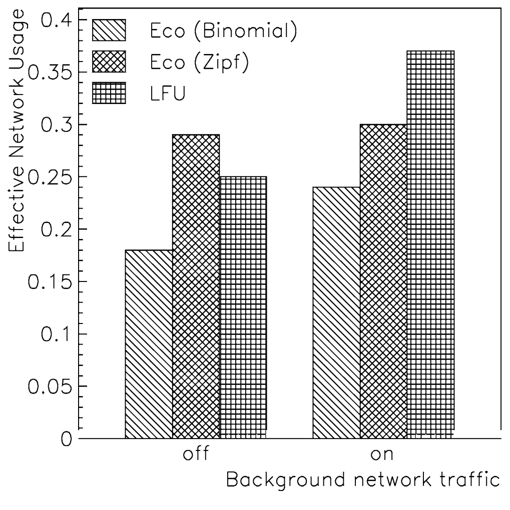
8. Conclusions and Future Work

In this paper we have described an environment suitable for the simulation of realistic Grid scenarios and the evaluation of Grid optimisation algorithms. We have discussed various strategies in scheduling optimisation and replica optimisation and presented results showing their performance in tests carried out with the Grid simulator OptorSim.

We have shown that the choice of strategies used can affect the throughput of Grid jobs and the extent to which Grid resources are exploited. In particular, our



(a)



(b)

Figure 9. Effects of background network traffic on (a) mean job time and (b) effective network usage.

experiments show that *Queue Access Cost*, a scheduling algorithm which takes into account both the file access cost of jobs and the workload of computing resources, is the most effective at optimising computing and storage resources and reducing the average time to execute jobs.

We have also shown that a suitable choice of data replication strategy can improve Grid performance; for most situations, particularly with large numbers of jobs, the economy-based strategies we have developed have the greatest effect, regardless of the presence of background (non-Grid) network traffic.

In the future, OptorSim will continue to be developed to simulate the Grid environment even more realistically. This will include more accurate simulation of clusters of worker nodes within Computing Elements and the effects of different types of storage devices such as tape storage. In addition, OptorSim currently assumes a static Grid in which resources are always available. In reality, however, resources such as network links and Computing Elements will not always be available. Although work has been carried out in simulating the unstable nature of network traffic, more investigation into the dynamic Grid environment is required.

An important study will be the comparison of simulation results with real Grid performance measurements. These data should become more widespread as the use of Grid technology increases and so adjustments can be made to OptorSim to bring it closer to the reality of a working Grid and validate the results which have been achieved.

Some areas of Grid research are moving towards a Grid conceived as a network of inter-operable services, with user access regulated and optimised by means of suitable meta-level optimisation agents. These issues are the focus of *Service-Oriented Computing* [15]. Since OptorSim has been demonstrated to be a valuable instrument for the simulation of Data Grids (which can be seen as networks where the only service to be optimised is data access), another interesting research direction is the extension of our simulator for service-oriented environments.

In summary, there are many future directions which can be taken and the extensible nature of the OptorSim code means that any of them can be explored.

Acknowledgments

This work was partially funded by the European Commission program IST-2000-25182 through the European DataGrid Project, the ScotGrid Project and PPARC. We would also like to thank James Casey, Heinz Stockinger and Tony Doyle for their feedback on the paper.

References

1. M.F. Arlitt and C.L. Williamson, “Web Server Workload Characterization: The Search for Invariants”, in *ACM Sigmetrics International Conference on Measurements and Modeling of Computer Systems*, Philadelphia, PA, USA, 1996.

2. P. Barford and M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation”, in *ACM Sigmetrics International Conference on Measurements and Modeling of Computer Systems*, Madison, WI, USA, 1998.
3. W.H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger and F. Zini, “Design of a Replica Optimisation Framework”, Technical Report DataGrid-02-TED-021215, CERN, Geneva, Switzerland, 2002.
4. W.H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger and F. Zini, “OptorSim – a Grid Simulator for Studying Dynamic Data Replication Strategies”, *International Journal of High Performance Computing Applications*, Vol. 17, No. 4, 2003.
5. W.H. Bell, D.G. Cameron, R. Carvajal-Schiaffino, P. Millar, K. Stockinger and F. Zini, “Evaluation of an Economy-Based File Replication Strategy for a Data Grid”, in *International Workshop on Agent Based Cluster and Grid Computing at CCGrid2003*, Tokyo, Japan, 2003.
6. L. Breslau et al., “Web Caching and Zipf-like Distributions: Evidence and Implications”, in *IEEE INFOCOM’99*, New York, NY, USA, 1999.
7. R. Buyya and M. Murshed, “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”, *The Journal of Concurrency and Computation: Practice and Experience*, pp. 1–32, 2002.
8. L. Capozza, K. Stockinger and F. Zini, “Preliminary Evaluation of Revenue Prediction Functions for Economically-Effective File Replication”, DataGrid-02-TED-020724, CERN, Geneva, Switzerland, July 2002.
9. P. Crosby, “EDGSim”. http://www.hep.ucl.ac.uk/∼pac/EDGSim/
10. B.T. Huffman et al., “The CDF/DO UK GridPP Project”, CDF Internal Note 5858.
11. H. Lamehamedi, Z. Shentu, B. Szymanski and E. Deelman, “Simulation of Dynamic Data Replication Strategies in Data Grids”, in *Proceedings of the 12th Heterogeneous Computing Workshop (HCW2003)*, Nice, France, 2003.
12. V. Lefebure and T.W. (eds), *The Spring 2002 DAQ TDR Production*, CMS Internal Note, 2005/000, Geneva, Switzerland.
13. Y.-T. Li, “GridNM”. http://www.hep.ucl.ac.uk/∼ytl/monitoring/gridnm/gridnm-client.html
14. C. Logg et al., “SLAC WAN Bandwidth Measuring Tests”. http://www.slac.stanford.edu/comp/net/bandwidth-tests/antonia/html/slac%_wan_bw_tests.html
15. M. Orlowska, S. Weerawarana, M. Papazoglou and J. Yang (eds.), “Service-Oriented Computing – ICSOC 2003 First International Conference”, in Lecture Notes in Computer Science, No. 2910. Springer-Verlag: Trento, Italy, 2003.
16. K. Ranganathan and I. Foster, “Identifying Dynamic Replication Strate gies for a High Performance Data Grid”, in *Proceedings of the International Grid Computing Workshop*, Denver, CO, USA, 2001.
17. K. Ranganathan and I. Foster, “Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications”, in *International Symposium of High Performance Distributed Computing*, Edinburgh, Scotland, 2002.
18. “SLAC WAN Bandwidth Measuring Tests at FNAL”. http://dmzmonO.deemz.net/∼wanbanmon/html/slac_wan_bw_tests.html#summary
19. “The European DataGrid Project”. http://www.edg.org
20. A. Tirumala et al., “Iperf.” http://dast.nlanr.net/Projects/Iperf/
21. “UK e-Science Grid Network Monitoring”. http://gridmon.ucs.ed.ac.uk/gridmon/
22. W. Vickrey, “Counterspeculation, Auctions, and Competitive Sealed Tenders”, *The Journal of Finance*, Vol. 16, No. 1, pp. 8–37, 1961.
23. WP2, “OptorSim, a Replica Optimiser Simulator”. http://cern.ch/edg-wp2/optimization/optorsim.html
24. WP7, “Network Services”. http://ccwp7.in2p3.fr/
25. G.K. Zipf, *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press: Cambridge, MA, 1932.