

# Dynamic Data Replication in LCG 2008

C. Nicholson<sup>1</sup>, D. G. Cameron<sup>2</sup>, A. T. Doyle<sup>1</sup>, A. P. Millar<sup>1</sup>, K. Stockinger<sup>3</sup>

<sup>1</sup> University of Glasgow, Glasgow, G12 8QQ, Scotland

<sup>2</sup> CERN, European Organization for Nuclear Research, 1211 Geneva, Switzerland

<sup>3</sup> Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

## Abstract

To provide performant access to data from high energy physics experiments such as the Large Hadron Collider (LHC), controlled replication of files among grid sites is required. Dynamic, automated replication in response to jobs may also be useful, and has been investigated using the grid simulator OptorSim. In this paper, results from simulation of the LHC Computing Grid in 2008, in a physics analysis scenario, are presented. These show, first, that dynamic replication does give improved job throughput; second, that for this complex grid system, simple replication strategies such as LRU and LFU are as effective as more advanced economic models; third, that grid site policies which allow maximum resource sharing are more effective; and lastly, that dynamic replication is particularly effective when data access patterns include some files being accessed more often than others, such as with a Zipf-like distribution.

## 1 Introduction

The latest high energy particle accelerators, such as the Large Hadron Collider (LHC) at CERN, the European Organization for Nuclear Research, are eagerly awaited by particle physicists. The LHC alone is expected to produce tens of petabytes of raw data annually, making it necessary to distribute computing and storage resources in a worldwide grid. Specific to the requirements of the LHC is the LHC Computing Grid (LCG), which uses a three-tiered grid architecture. In this architecture, raw data is produced at the CERN central Tier-0; it is replicated in a controlled fashion to a number of Tier-1 sites, which are responsible for permanent storage. Each Tier-1 has a number of associated Tier-2 sites, providing computing power for user analysis along with modest storage capabilities.

In such a data-intensive grid, replication of files

among grid sites is important to improve grid performance. Apart from the kind of controlled replication planned for LCG, however, dynamic data replication - automatically replicating files between sites in response to jobs - may also be useful in achieving an optimal distribution of data around the grid, and there are numerous possible strategies for such dynamic replication. Current grids are not yet mature enough to allow testing of this kind of replication, however, so the grid simulator OptorSim [1] has been designed to explore the effects of dynamic data replication under a variety of conditions.

In previous work, OptorSim was used to study various grids including the European DataGrid [3] and the LCG topology of 2004 [6]. In this paper, the much more complex setting of LCG in 2008 (the first full year in which the LHC will produce data) is investigated, evaluating some simple replication strategies as well as an economic model of file replication, under a range of conditions. First, a brief survey of related work is given. The grid optimisation principles and replication strategies investigated are then presented, followed by a short description of OptorSim in Section 4. The experiments performed are described in Section 5 and the results presented in Section 6, before drawing some conclusions in Section 7.

## 2 Related Work

In recent years there have been several grid simulation projects, examining various aspects of grid systems. The *Models of Networked Analysis at Regional Centres for LHC Experiments*, or MONARC, project was initiated to explore computing models for the LHC projects. As part of this project, a simulator was developed to provide a realistic simulation of distributed computing systems with which different data processing architectures could be evaluated [10]. Its main difference from OptorSim, however, lies in the lack of in-

infrastructure for automated replication and replica optimisation. GridSim [5] is a grid simulation toolkit developed to investigate resource allocation techniques and in particular, a computational economy. The focus is on scheduling and resource brokering; there is not, however, capability for data management such as would be required for investigation of replica optimisation strategies. ChicSim [12] is a simulator designed to investigate scheduling strategies in conjunction with data location. Bricks Grid [13], while initially focusing on job scheduling, has been extended to include replica management. Its replica management components, however, are centralised rather than the distributed architecture used in OptorSim. All these projects therefore give a complementary approach to that used in OptorSim, allowing exploration of different areas of parameter space.

### 3 Grid Replica Optimisation

In a grid environment, there are many variables which determine its overall performance, and which are impossible to harmonise into one optimal configuration for the whole grid. It is possible, however, to optimise those variables which are a part of the grid middleware itself. For a data grid, the most important areas to optimise will then be job scheduling and data management. This paper will concentrate on data management, and dynamic replica optimisation in particular.

#### 3.1 A Replica Optimisation Service

There are several important design decisions behind the OptorSim replication model. First, it is distributed rather than centralised or hierarchical. Each site is able to manage its own replica content and there is no single point of failure; it also removes the need for sites to know the state of the whole grid.

Second, it operates on a pull rather than a push model. In a push model, the site containing a particular data file would decide when to replicate it and where to. In a pull model, a site which did not initially have the file would decide when to replicate it to itself, and where from. In a real particle physics grid, however, sites would be unlikely to accept spontaneous replication of data from some other site and so a pull model is favoured, with each site responsible for its own replica optimisation.

Finally, a replication trigger must be chosen. When a site is requested for a file which it does not have, for example, this could trigger the first stage of the replication strategy. Another possible trigger could be a file on some other site reaching a certain level

of popularity. This would require monitoring of all file popularities, however. The simplest approach is to trigger on a file request, and this is what has been implemented in OptorSim.

The overall aim of such a distributed replication model would be to achieve global optimisation as a result of local optimisation by each site's Replica Optimiser (RO). Each RO therefore has two goals: the minimisation of individual job execution costs, and the maximisation of usefulness of locally stored files. A good replication strategy will be one which achieves a good trade-off between individual running times and overall resource utilisation.

#### 3.2 Stages of a Replication Strategy

Replication can be logically separated into three stages through which any replication strategy must proceed. These are delineated as follows, with each stage depending on the success of the preceding stage. First comes the *Replication Decision* where, given the trigger condition, the RO at a site must decide whether or not to replicate the file to its local site. If it decides not to replicate, the file must be read remotely. The second stage is *Replica Selection*, where if the RO has decided to replicate the file, it must choose which existing replica to copy. Finally, in the *File Replacement* stage, if the local site does not have sufficient space to store the new replica, one or more files must be deleted until there is enough space. These three stages will each be discussed for the replication strategies which are described in Section 4.

## 4 OptorSim

OptorSim is an event-driven simulator, written in Java. As dynamic data replication involves automated decisions about replica placement and deletion, the emphasis is on simulation of the replica management infrastructure. The architecture and implementation are described in [8] and so only a brief description is given here.

### 4.1 Architecture

The conceptual model of the OptorSim architecture is shown in Figure 1. In this model, the grid consists of a number of sites, connected by network links. A grid site may have a Computing Element (CE), a Storage Element (SE) or both. Each site also has a Replica Optimiser (RO) which makes decisions on replications to that site. A Resource Broker (RB) handles the scheduling of jobs to sites, where they run on the CEs. Jobs process files, which are stored in the SEs and can

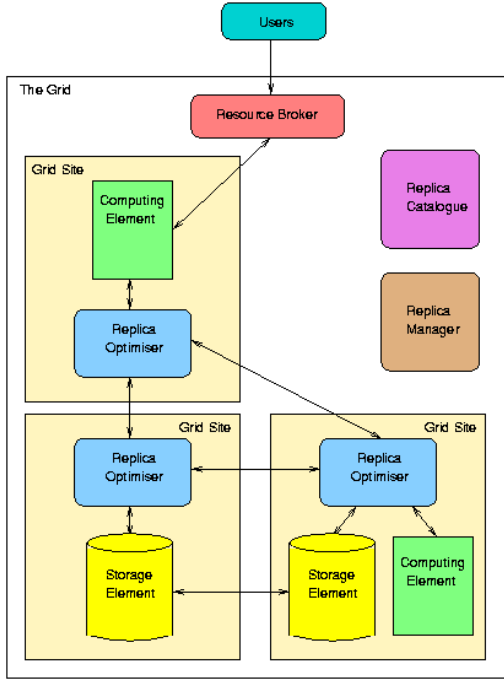


Figure 1: Grid architecture used in OptorSim.

be replicated between sites according to the decisions made by the RO. A Replica Catalogue holds mappings of logical filenames to physical filenames and a Replica Manager handles replications and registers them in the Catalogue.

## 4.2 Simulation Inputs

### 4.2.1 Grid Topology.

To input a grid topology, a user specifies the storage capacity and computing power at each site, and the capacity and layout of the network links between each. SEs are defined to have a certain capacity, in MB, and CEs to have a certain number of “worker nodes” with a given processing power. Sites which have neither a CE nor an SE act as routers on the network. Background traffic on the network can also be simulated.

### 4.2.2 Jobs and Files.

A physics analysis job usually processes a number of files. This is simulated by defining a list of jobs and the files that they need; a job will process some or all of the files in its dataset, according to the *access pattern* which has been chosen. The time a file takes to process depends on its size and on the number and processing power of worker nodes at the CE.

### 4.2.3 Access Patterns.

Several file access patterns have been implemented in OptorSim, aiming to simulate various possible grid scenarios. These include *sequential* access, where a job accesses each file it requires once, in sequence; and *Zipf*<sup>1</sup>, where a few files are accessed many times while others are accessed infrequently. It has been shown in [9] that both cases can occur in a particle physics situation and it is therefore interesting to examine the effects of replication under both these conditions.

### 4.2.4 Site Policies.

Different grid sites are likely to prioritise different kinds of job. A university with strong involvement in a particular experiment, for example, may prefer to accept jobs from that experiment, whereas a regional Tier 2 centre may be contracted to serve all experiments. In OptorSim, each site is given a list of job types which it will accept.

## 4.3 Optimisation Algorithms

There are two kinds of optimisation algorithm which may be investigated using OptorSim: the job scheduling algorithms used by the RB to decide which sites jobs should be sent to, and the data replication algorithms used by the RO at each site to decide when and how to replicate files. The focus of this paper is on the data replication algorithms, and so the job scheduling algorithms are not described here.

There are three broad options for replication strategies in OptorSim. Firstly, one can choose to perform no replication. Secondly, one can use a “traditional” algorithm which, when presented with a file request, always tries to replicate and, if necessary, deletes existing files to do so. Algorithms in this category are the LRU (Least Recently Used), which deletes those files which have been used least recently, and the LFU (Least Frequently Used), which deletes those which have been used least frequently in the recent past. Thirdly, one can use an economic model in which sites “buy” and “sell” files using an auction mechanism, and will only delete files if they are less valuable than the new file. Details of the auction mechanism and file value prediction algorithms can be found in [4]. There are currently two versions of the economic model: the binomial economic model, where file values are predicted by ranking the files in a binomial distribution according to their popularity in the recent past, and

<sup>1</sup>A Zipf-like distribution is defined as  $P_i \propto i^{-\alpha}$ , where  $P_i$  is the frequency of occurrence of the  $i^{\text{th}}$  ranked item and  $\alpha \leq 1$  (a pure Zipf distribution would have  $\alpha = 1$ ).

the Zipf economic model, where a Zipf-like distribution is used instead (a Zipf distribution is a power law in which a few events occur very frequently, while most events occur infrequently).

#### 4.4 Evaluation Metrics

For evaluating grid performance, different users may have different criteria. An ordinary user will most likely be interested in the time a job takes to complete. Resource owners, on the other hand, will want to see their resources being used efficiently. The evaluation metric used in this paper are: *mean job time*, which is the average time a job takes to run, from the time of scheduling to completion; and *effective network usage (ENU)*, which is defined as

$$ENU = \frac{N_{remote\ file\ accesses} + N_{replications}}{N_{remote\ file\ accesses} + N_{local\ file\ accesses}}$$

The ENU is thus the ratio of file requests which use network resources to the total number of file requests, and so the lower the ENU the less loaded the network is and hence the more efficiently it is being used.

### 5 Experimental Setup

To evaluate the performance of these replication strategies in a realistic grid scenario, a simulation of LCG using the planned resources for physics analysis in 2008 was set up as follows. The topology, based on the the layout of the national research networks, is shown in Figure 2. The network capacity shown in Figure 2 was modified through the simulation by the addition of background network traffic, which varied according to the time of day in the simulation as shown in Figure 3. This profile was based on measurements of available bandwidth between CERN and Lyon, and assumes that patterns of network usage will not change significantly over the next few years.

#### 5.1 Jobs and Files

Four datasets were defined, corresponding to the four major LHC experiment collaborations. Each of these datasets was placed at the Tier-0 (CERN) and each Tier-1 at the beginning of the simulation, so each file initially had 12 replicas around the grid. Six job types were defined, with the job and file parameters as shown in Table 1. The simulated jobs processed their given subset of files from the dataset. When a job ran on a site, it retrieved the files it required (according to the chosen access pattern) and processed them according to the computing resources available at that site. The probability of a particular job being

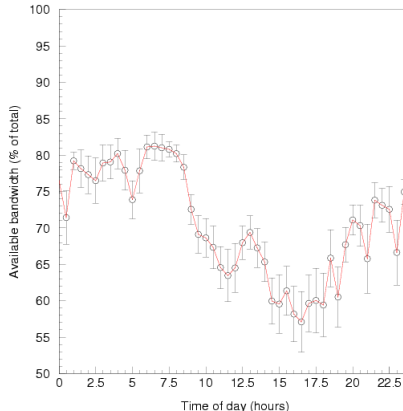


Figure 3: Profile of bandwidth variation used in LCG 2008 simulation.

run on the grid was modelled by the relative number of expected users for the different experiments.

Job	Dataset size (TB)	Total no. of files	Files/ job
alice-pp	50	25000	25
alice-hi	25	12500	125
atlas	200	100000	50
cms	75	37500	25
lhcb-small	75	37500	38
lhcb-big	75	37500	375

Table 1: Job configuration parameters used in the LCG 2008 configuration.

#### 5.2 Resources

The compute and storage requirements for LCG in the first few years of LHC data-taking were drawn from the LCG Technical Design Report [2]. While the data is initially stored at Tier-0 and Tier-1 sites, user analysis jobs are expected to run at Tier-2 sites. Each Tier-2 was therefore given an averaged CE of 645 kSI2000. The Tier-0 and Tier-1 sites were given SEs according to their planned capacities, each being sufficient to hold a complete copy of all the data files.

Each Tier-2 site was given a canonical value, averaging the total Tier-2 requirements over the number of Tier-2 sites. This gave an average SE size of 197 TB. Due to the limitations of available memory when running the simulation, however, the simulations were restricted to the order of 1000 jobs. To reach a state in the simulation where the SEs are full and file dele-

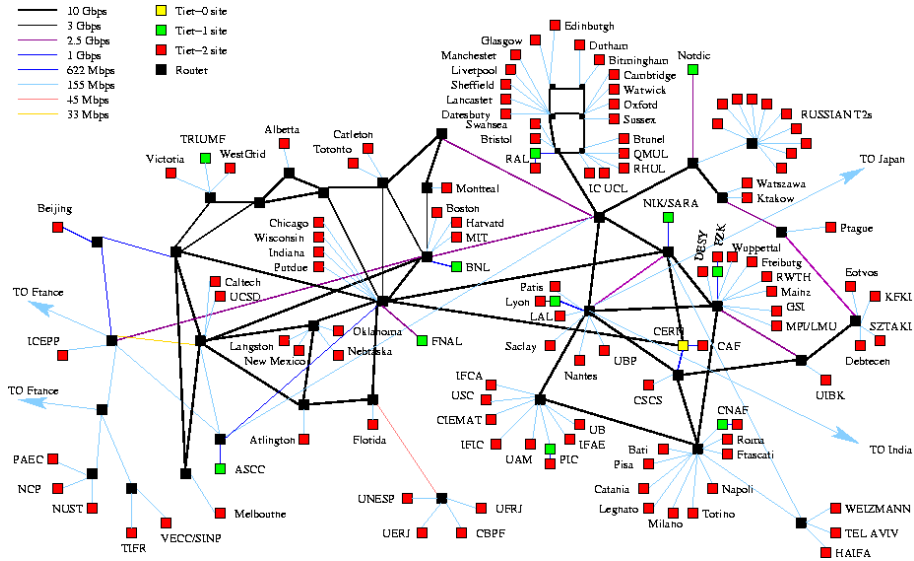


Figure 2: Simplified topology of LCG in 2008.

tion is occurring would require about 200,000 jobs if the SEs were kept at 197 TB, and so the Tier-2 SE sizes were scaled down to 500 GB. These then hold 250 files, allowing file replacement to start when at most 10 jobs have been submitted to a site. This has the disadvantage that the file prediction algorithms will not perform to their best advantage. The effect of changing the size of the dataset compared to the SE sizes, however, is among the tests presented.

## 6 Results

For the results presented here, 1000 jobs were submitted to the grid at random intervals and, unless stated, a sequential access pattern was used. A job scheduling algorithm was used which balanced the queue length at sites with the data requirements of the jobs, which has previously been shown [7] to perform well. Tests were repeated at least 3 times and a mean value taken.

### 6.1 Varying Dataset Size

The ratio of average SE size to total dataset size can be a useful characterisation metric for a data grid, and is here designated  $D$ . The value of  $D$  indicates the likelihood of replication occurring. If  $D > 1$ , an average SE is able to hold all the files that jobs could require, so there will never be any deletion. For  $D < 1$ , the replication strategy becomes more important, as the SE is not capable of holding all the files and deletion must take place. For  $D \ll 1$  due

to a large dataset, however, replication will begin to lose its advantage, as each new job is likely to request files which are not in the access history. The value of  $D$  at which the switch between these two behaviours occurs will depend on the grid itself.

The default value for the experiments presented here, due to the scaling of Tier-2 SEs, is  $1.2 \times 10^{-3}$ , which is too small to effectively compare the replication algorithms. The first experiment presented therefore examines the dependency of the replication strategies on  $D$ . The overall dataset size was successively halved, varying  $D$  from  $1.2 \times 10^{-3}$  to  $7.5 \times 10^{-2}$ , bringing it closer to a more realistic level of  $\mathcal{O}(10^{-1})$ . The results of this test are shown in Figure 4. It did not prove possible to test higher values of  $D$  due to the memory limitations of the available hardware, but testing with  $D \sim 1$  would be desirable in future work.

Firstly, as should be expected, the mean job time without replication is independent of  $D$ . For  $D \lesssim 10^{-2}$ , replication gives no advantage; for  $D > 10^{-2}$ , the mean job time drops rapidly for all the replication strategies. For the highest value of  $D$  tested, the LRU and LFU are slightly faster than the economic models. Examining the ENU, it is seen to fall as  $D$  increases, for all the replication algorithms. Without replication it naturally remains constant. This shows the increasing effectiveness of the replication strategies as their access histories contain a more representative sample of the whole dataset.

Although these results were gained with 1000 grid jobs, tests with simpler grids [11] show, within errors, a linear increase in job time with number of jobs up to

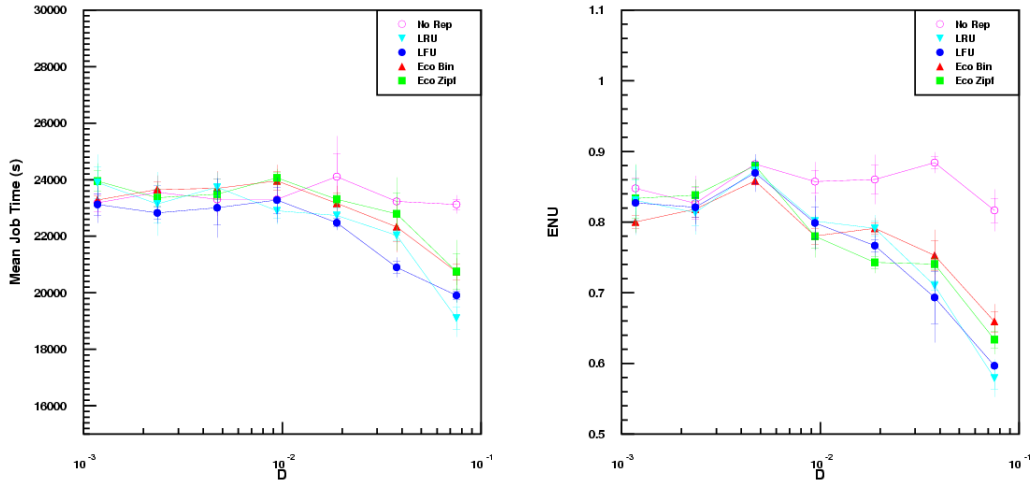


Figure 4: Mean job time (left) and ENU (right) for different replication algorithms, varying the value of  $D$ .

$\mathcal{O}(10^5)$  jobs, which is more realistic. This means that with realistic values for  $D$  and higher numbers of jobs, this relative improvement would hold. Replication is therefore an important way of reducing job times and network usage, and the relatively simple LRU and LFU strategies are the most effective for this topology.

In the sections which follow, the results are taken with the low value of  $D$  and it should therefore be remembered that in reality, replication would have a much stronger effect.

## 6.2 Effects of Site Policies

In the previous experiments, site policies - the types of jobs which would be accepted by each site - were set according to their planned usage. Here, the effect of site policies on the overall running of the grid are investigated. This was done by defining two extremes of policy. In the first, designated *All Job Types*, all sites accepted all job types. In the second, designated *One Job Type*, each site would accept only one job type, with an even distribution of sites for each job type. The default set of site policies is therefore in between these two extremes, and is designated in the results below as *Mixed*. The results are shown in Figure 5. These results show that the overall pattern of site policies on the grid have a powerful effect on performance. The mean job time with the *All Job Types* policy is about 60% lower than with the *One Job Type* policy. This is true across all the replication strategies, although the effect is strongest with no replication and with the LRU; it is again seen that the simple replication strategies are slightly faster than the eco-

nomic models. *All Job Types* also gives a lower ENU (about 25% lower than the others). It seems clear that an egalitarian approach, in which resources are shared as much as possible, yields benefits to all grid users.

## 6.3 Effects of Zipf-like Access Pattern

In all the experiments presented so far, jobs have accessed their files using a sequential access pattern. As Section 4.2.3 mentioned, however, Zipf-like file access may also be significant in a grid such as LCG. The performance of the replication strategies with a Zipf access pattern were therefore compared to that with sequential access, with the results shown in Figure 6. This shows quite a different pattern to the previous results. Although the four replication algorithms still have very similar performances, they are now about 75% faster than without replication. The ENU is correspondingly lower. This is due to the way in which a few files from each job's fileset are accessed many times during the jobs, while others are accessed infrequently. This allows the access histories to predict file values more accurately than with the sequential pattern, where they may see a file only once. As the number of jobs and the proportion of the whole dataset seen by an individual SE increases, however, the results with sequential access should tend towards a similar pattern as for the Zipf access. This is borne out by the results from varying  $D$  with sequential access. The presence of any Zipf-like element, even if combined with a sequential pattern, would make dynamic replication highly desirable.

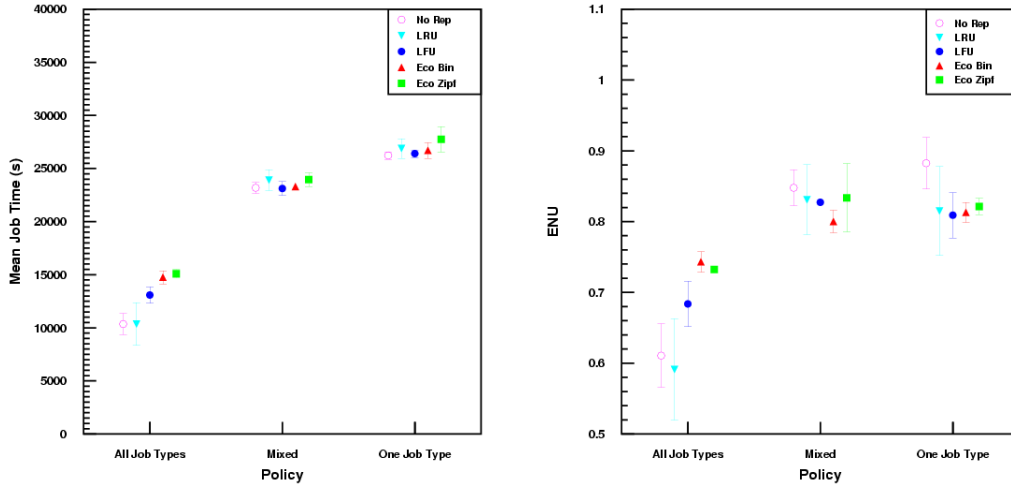


Figure 5: Mean job time (left) and ENU (right) for different replication algorithms, with different site policies.

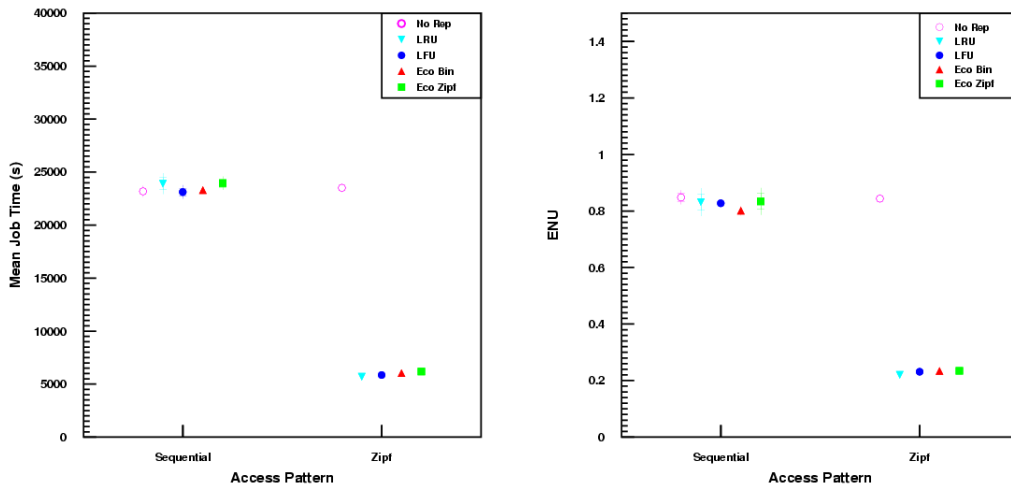


Figure 6: Mean job time (left) and ENU (right) for different replication algorithms, with Zipf-like access pattern.

## 7 Conclusions

OptorSim has been used to explore file replication strategies, under various conditions, in simulations of the LHC Computing Grid (LCG). LCG is an example of a large and complex high-energy physics data grid, with well-defined resource plans and use cases, and so is a good test case for dynamic data replication. While the results presented here are for particle physics, OptorSim could also be used to simulate other data-intensive grids, such as those developed for biomedical or astronomical applications.

First, it was shown that dynamically replicating data between sites using a sequential file access pattern decreased the running time of grid jobs by about 20% and reduced usage of the network by about 25%, especially as sites' Replica Optimisers gained more knowledge of the overall dataset. While the performances of different replication strategies were similar, the simpler LRU and LFU strategies were found to perform up to 20% and 30% better, respectively, than the economic models. In previous work ([8], [7], [6]), the economic models were found to perform better; this may be due to the fact that in those scenarios, the grids were much simpler, leading to lower overheads for the economic model. Another reason may be that there were fewer initial replicas in these simulations, whereas in this case the higher number reduced the need for dynamic replication and thus reduced the gain from the economic models. In future, it would be good to extend the simulation to higher numbers of jobs and ascertain whether this behaviour does hold.

Examining site policies, it was found that a policy which allowed all experiments to share site resources was most effective in reducing data access time and network usage. Finally, it was shown that if data access patterns are Zipf-like, dynamic replication has a much stronger effect than with sequential access, with performance gains of about 75%.

In general, dynamic replication is likely to be a valuable tool in improving grid performance under a range of conditions, not only for high energy physics grids but for any grid where large datasets are involved. While economic models of file replication may be useful for future grids, simple replication strategies such as LRU and LFU perform well for current-generation grids like LCG. The next step is to implement prototype tools for dynamic replication for testing in a real grid environment.

## References

- [1] OptorSim - A Replica Optimiser Simulation. <http://cern.ch/edg-wp2/optimization/optorsim.html>.

- [2] LHC Computing Grid Technical Design Report. Technical Report CERN-LHCC-2005-024, CERN, June 2005.
- [3] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Simulation of Dynamic Grid Replication Strategies in OptorSim. *Int. Journal of High Performance Computing Applications*, 17(4), 2003.
- [4] 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 *Proceedings of 3rd IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid 2003)*, Tokyo, Japan, May 2003. IEEE CS-Press.
- [5] Rajkumar Buyya and Manzur Murshed. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience*, 14:1175–1220, 2002.
- [6] D. Cameron, P. Millar, C. Nicholson, R. Carvajal-Schiaffino, F. Zini, and K. Stockinger. OptorSim: a Simulation Tool for Scheduling and Replica Optimisation in Data Grids. In *Computing in High Energy and Nuclear Physics (CHEP)*, Interlaken, Switzerland, September 2004.
- [7] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini. UK Grid Simulation with OptorSim. In *UK e-Science All Hands Meeting*, Nottingham, UK, September 2003.
- [8] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini. Analysing Scheduling and Replica Optimisation Strategies for Data Grids with OptorSim. *Journal of Grid Computing*, 2(1):57–69, 2004.
- [9] A. Iamnitchi and M. Ripeanu. Myth and reality: Usage behavior in a large data-intensive physics project. Technical Report TR2003-4, GriPhyN, 2003.
- [10] Iosif Legrand. Multi-threaded, discrete event simulation of distributed computing system. In *Computing in High Energy and Nuclear Physics (CHEP)*, Padova, Italy, February 2000.
- [11] C. M. M. Nicholson. *File Management for HEP Data Grids*. PhD thesis, University of Glasgow, 2006.
- [12] K. Ranganathan and I. Foster. Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications. In *Proc. of the 11th International Symposium for High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, 2002.
- [13] A. Takefusa, O. Tatebe, S. Matsuoka, , and Y. Morita. Performance Analysis of Scheduling and Replication Algorithms on Grid Datafarm Architecture for High-Energy Physics Applications. In *Proc. of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press, June 2002.