UNIVERSITY
of
GLASGOW

Department of Physics & Astronomy
Experimental Particle Physics Group
Kelvin Building, University of Glasgow,
Glasgow, G12 8QQ, Scotland
Telephone: +44 (0)141 339 8855 Fax: +44 (0)141 330 5881

GLAS-PPE/2003-09
August 2003

Evaluation of an Economy-Based File Replication Strategy for a Data Grid

William H. Bell[1], David G. Cameron[1], Ruben Carvajal-Schiaffino[2],
A. Paul Millar[1], Kurt Stockinger[3], Floriano Zini[2]
[1] University of Glasgow, Glasgow, G12 8QQ, Scotland
[2] ITC-irst, Via Sommarive 18, 38050 Povo (Trento), Italy
[3] CERN, European Organization for Nuclear Research, 1211 Geneva, Switzerland

Abstract

Optimising the use of Grid resources is critical for users to effectively exploit a Data Grid. Data replication is considered a major technique for reducing data access cost to Grid jobs. This paper evaluates a novel replication strategy, based on an economic model, that optimises both the selection of replicas for running jobs and the dynamic creation of replicas in Grid sites. In our model, optimisation agents are located on Grid sites and use an auction protocol for selecting the optimal replica of a data file and a prediction function to make informed decisions about local data replication. We evaluate our replication strategy with OptorSim, a Data Grid simulator developed by the authors. The results show that our proposed strategy can result in a marked improvement over traditional replication strategies in a Grid environment.

International Workshop on Agent based Cluster and Grid Computing at International Symposium on Cluster Computing and the Grid (CCGrid 2003)
Tokyo, Japan

1 Introduction

Grid computing is a novel and emerging paradigm, whose goal is providing *virtual organisations* of geographically distributed users with software/hardware infrastructures that allow the effective sharing of computational and storage resources. Several international projects are currently working towards the realisation of Data Grids [7, 10] for the analysis of huge amounts of distributed data by computationally intensive applications. A Data Grid should provide virtual organisations with a set of services that automate the access to resources, bringing the needs of a single user (typically minimisation of execution cost for their jobs) into agreement with the demands of the virtual organisation of which the users are members (maximisation of Grid resource exploitation).

Data is the most important resource in a Data Grid, where users' jobs require access a large quantity of data. This data is stored in files which can be spread among geographically diverse Grid sites. The execution cost for a job may vary considerably, depending on the computing resource chosen for job execution and the location of data files to which the job requires access. Data replication is considered to be an important technique in the reduction of job execution cost [10]. Replication involves the creation of identical copies of data files that are distributed over various Grid sites. This can reduce data access latency and increase the robustness of Grid applications. However, in order that Grid performance can actually be improved, it is essential that data access and replication services are able to deal with the dynamic changes in the Grid environment. In particular, given the dynamic distribution of data files and file requests from jobs, such strategies should be able to:

1. determine the "best" replica, when given a request by a job for a particular file;
2. trigger both replication and deletion of data files in Grid sites by analysing the patterns of previous file requests; thereby affecting the migration of files toward sites that show a corresponding increased frequency of file-access requests.

In [6] the authors proposed an approach, based on an economic model, that aims to fulfil these requirements. In the model, data files represent the goods in the market and are traded by different optimisation agents on each Grid site according to file requests from running jobs. The idea underlying our model is to achieve global optimisation of replica distribution through emergent marketplace behaviour. Our economic approach is currently being evaluated using a Data Grid simulator called OptorSim [2]. OptorSim was developed to study the complex nature of a typical Data Grid and evaluate the effectiveness of replica optimisation algorithms within such an environment. Some early results of this evaluation are presented in [2], where a replication strategy based on a simplified version of the economic model was considered.

In this paper we extend our economy-based replica optimisation strategy. We introduce a novel auction protocol for optimal replica selection and discuss a prediction function for future file values that in turn triggers dynamic replication for highly requested data files. In order to evaluate the efficiency of our algorithm, we run a set of benchmarks on our Grid simulator OptorSim based on a real-world Data Grid testbed from High Energy Physics.

2 Related Work

Economic approaches have been mainly used in Grid computing for minimising the cost of job scheduling. Here, we briefly describe a few examples and refer to [8] for a more complete overview. The *SPAWN* system [17] provides a market mechanism for trading CPU-times in a network of workstations. The *POPCORN Project* [13] provides an infrastructure for globally distributed computation with market mechanisms for buying and selling CPU-time based on the Vickrey auction protocol [16]. POPCORN is currently based on a central market that could not scale with a large number of concurrent buyers and sellers. *Nimrod-G* [4] is an economy-driven resource broker for scheduling parametric computations in a typical Grid environment. Resource allocation is based on a deadline and budget constraint scheduling algorithm [3] with the goal of either minimising the runtime of jobs or minimising the cost of the usage of Grid resources. However, the location of data (and thus considerations about optimal replica selection) are not part of the scheduling algorithm.

Various examples of replication and caching strategies are discussed and tested within a simulated Grid environment in [14], while their combination with scheduling algorithms is studied in [15]. The replication algorithms proposed are based on the assumption that popular files in one site are also popular in other sites. Replication from one site to another is triggered when the popularity of a file overcomes a threshold and the destination site is chosen either randomly or by selecting the least loaded site.

In this paper, we take a complementary approach. Our replication algorithms are used by Grid sites when they need data locally and are based on the assumption that in computational Grids there are areas where particular sets of data are highly requested.

3 Design of OptorSim

OptorSim [2] simulates the Grid architecture shown in Figure 1 for studying economy-based replication strategies as presented later in Section 4. The simulation is constructed assuming that the Grid consists of several sites, each of which may provide computation and data-storage resources (called Computing and Storage Elements) for data intensive jobs.

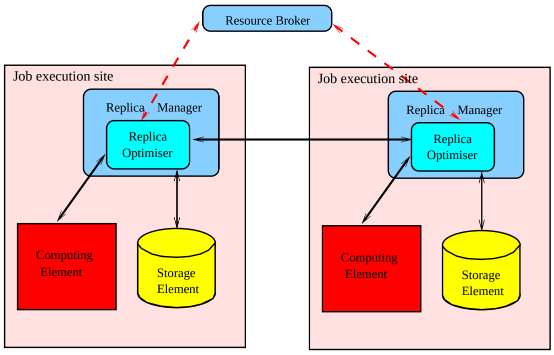

Figure 1: Simulated DataGrid Architecture.

Jobs are submitted to the Grid over a period of time via the *Resource Broker (RB)*. The RB schedules each job to the *Computing Element (CE)* with the goal to improve the overall throughput of the Grid. A *Replica Manager (RM)* at each site manages the data flow between sites and interfaces between the computing and storage resources and the Grid. The *Replica Optimiser (RO)* inside the RM is responsible for the dynamic creation and deletion of replicas of files at each site.

Since the Data Grid is a highly dynamic environment, it is important to be able to cope with the changes in the status of resources when performing resource scheduling. In particular, we should avoid making a priori irrevocable decisions at job scheduling time about which file replicas will be used to access data for a particular job.

Therefore, optimisation is performed at three points in time during the the life-time of a job:

1. The first optimisation phase occurs when the RB determines on which CE the job should run. We refer to this phase as *Scheduling Optimisation*.

2. In the second phase optimal *Dynamic Replica Selection* is achieved during the run time of a job via the auction mechanism we describe in Section 4.2.

3. In the final phase, replication can be triggered at third party sites. This is *Dynamic Replica Optimisation*.

We detail the algorithms of these phases in the following sections.

3.1 Scheduling Optimisation

Our scheduling algorithm takes into account both locality of files requested by jobs and CE loads. The calculation of the file accessing cost is implemented in the RO function `getAccessCost()` [1]. For each CE, the RB calls `getAccessCost()` with a list of files required for the job. The function consults the Replica Catalogue to find all the replicas of each file, then calculates the time to access each replica from the given CE by examining the current network status. By summing the times to access the "best" replica of each file, `getAccessCost()` returns the estimated file access time the job would have if scheduled to the CE.

Since we are concerned with the time to complete a job, i.e. the time the job is submitted to the Grid and the time it has finished executing on a CE, we also take into account the workload of each CE. A combined cost is found by simply adding the file transfer cost obtained from `getAccessCost()` to the number of jobs waiting in the queue at the CE, as shown in (1).

$$\text{cost} = \text{getAccessCost}() + \text{estimatedQueuingTime} \tag{1}$$

The scheduling algorithm in OptorSim schedules a job to the CE with the minimum combined cost.

3.2 Replica Selection

Once a job is running on a CE, it requires access to files in order to process data. Since there can be multiple copies of each file available on the Grid, the RO function `getBestFile()` [1] is called whenever a file is requested by the job. The function uses the auction mechanism described in Section 4.2 to return the best available replica. Depending on the popularity of the file, replication may also occur if it is deemed to be economically profitable (see Section 4.3 for details).

3.3 Dynamic Replica Optimisation

Grid sites also monitor and log the patterns of file requests over time. If a particular file appears to be popular and a site decides it is likely to make a profit by purchasing it, replication can be triggered to third party sites: those not involved in the initial request for the file. This way the data will migrate towards the areas the data is most likely to be requested.

4 An Economy-Based Optimisation Strategy

We propose an economic model for data access and replication, to be used in optimisation phases 2 and 3. In the model, data files are "purchased" by CEs for running jobs and by *Storage Elements (SE)* to make an investment that will improve their expected future revenue. These files are sold by SEs to both CEs and other SEs. CEs try to minimise the file purchase cost, while SEs attempt to maximise their profits. CEs and SEs interact with intelligent optimisation agents which perform the reasoning required in our model.

The adoption of an economic approach has two main motivations. The first reason is to be able to make replication optimisation decisions in a distributed manner. Performing such complex multidimensional optimisations in a centralised manner would be difficult, as the domain (attributes of the resources being controlled) is large. By restricting ourselves to local optimisation through economic interactions, we make the problem manageable whilst allowing the system to achieve global optimisation through emergent marketplace behaviour. The second reason is that the Data Grid is a highly dynamic environment in which the availability of resources can change without warning. By using an economic model, we can exploit the dynamism of the market to make informed decisions at job execution time.

4.1 Internals of the Economic Model

The Grid service that performs optimisation of data access and replication is the *Replica Optimiser (RO)*. It is a distributed service and is provided by a set of *RO agents*, one for each Grid site (see Figure 2). A RO agent is invoked by a Grid job, running on a local Computing Element (CE), that needs to access a data file. The RO is able to select the best replica of that file, possibly triggering file replication between Grid sites if this is predicted to improve Grid performance. The abstract architecture for RO agents includes the following components:

Access Mediator (AM). This component processes file requests from jobs running on a CE. For each requested file, it starts an auction to identify the cheapest replica of the file (see Section 4.2 for details). The AM gathers bids for the file from local and remote *Storage Brokers (SB)*, selects the winner of the auction and performs file payments.

P2P Mediator (P2PM). These are responsible for establishing and maintaining a peer to peer communications infrastructure between Grid sites. They propagate auction messages between AMs and SBs.

Storage Broker (SB). This component is responsible for listening for file request messages from the local P2P Mediator. If it can meet the request with a file stored in the corresponding SE, it responds immediately to the P2P Mediator. If the file is not stored in the corresponding SE, it may start a *nested auction* in order to obtain a local replica of the file and be able to reply to the parent auction.

Since a Grid site might exclude CEs or SEs, some of the above components might be absent from the available RO agents for that site. This is depicted in Figure 2. The figure shows an example Data Grid containing four sites. Site 4 contains both Computing and Storage Elements and thus the local RO agents comprise of all components. Other sites include only the components needed and thus the local RO agents are simpler.

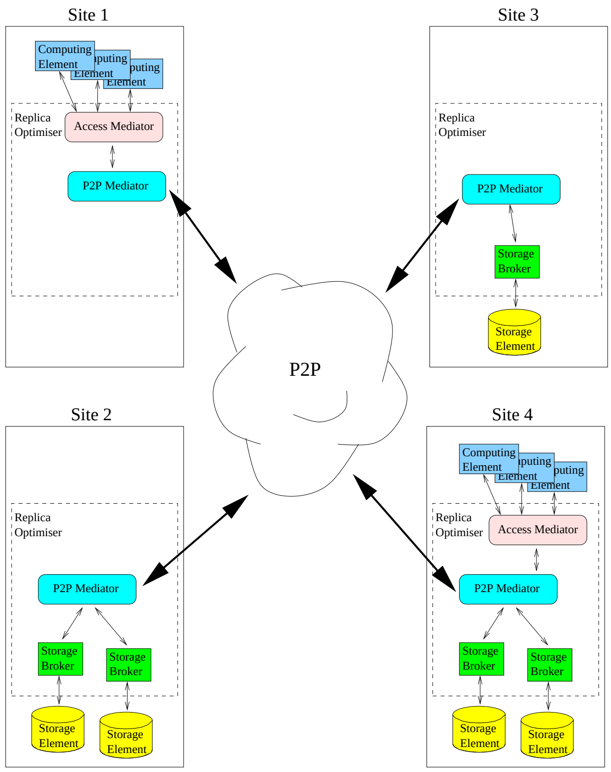

Figure 2: Components of the economy-based Grid model.

4.2 Auction Protocol

In our economic model the goal of the auction protocol is to select the cheapest replica of a file needed by a job running on a CE. To this end we use a type of *Vickrey auction* [16]. Vickrey auctions are *second-price sealed-bid* auctions. They involve a single negotiation round, in which each bidder submits a bid to the auctioneer. Other bidders cannot see the bid. The winner of the auction is the agent that made the highest bid, but only pays the price of the second-highest bid. An advantage of this type of auction over others is that the dominant strategy for each bidder is to bid his true valuation. In fact, bidding more than the true valuation is clearly a risky strategy: if he won the auction he could be asked to pay a higher price than he would otherwise. On the other hand, bidding less than the true valuation would reduce the chances of winning. Bidding the true valuation is then the best strategy for an agent. This maximises his chances of winning the auction and, if he won, he would pay a price less than his valuation (the second best price).

In our case the role of the auctioneer is played by the AM, while the SBs play the role of bidders. However, the AM does not start an auction for selling an item (a file in our case) but for buying it. The SBs bid the price they are willing to sell the file and the winning SB is paid the second-lowest bidding price by the AM. In other words, our economic model uses a *reverse Vickrey auction*.

The auction protocol we use is detailed in Figure 3 by a UML sequence diagram. When replica selection is performed by calling `getBestFile()`, the replica that minimises the access time is returned. It may be stored locally or remotely. In the latter case the file is accessed by the CE via remote file access. In our economic model we perform `getBestFile()` by starting a reverse Vickrey auction. The auction is performed by an *AccessMediator*, which starts an *auction* thread, then waits until the end of the auction before returning the winning replica to the CE.

An auction thread first issues a *BidRequest* message for the required file, which is propagated by the local P2P Mediator to the local SBs and other SBs via the P2P network. Depending on the topology of the network and the maximum distance for auction propagation (a parameter of the auction), the message will reach a set of Grid sites. Once it has issued a request for bids, the auction thread waits so potential file sellers can bid for the file (see *addBid* messages in Figure 3). After a fixed time, the auction thread selects the auction winner (i.e. the SB that submitted the lowest bid). An *Announce* message is propagated over the P2P network informing the SBs of the auction result[1)]

[1)] If the auction thread receives no bids whilst waiting, the auction will have no winner.

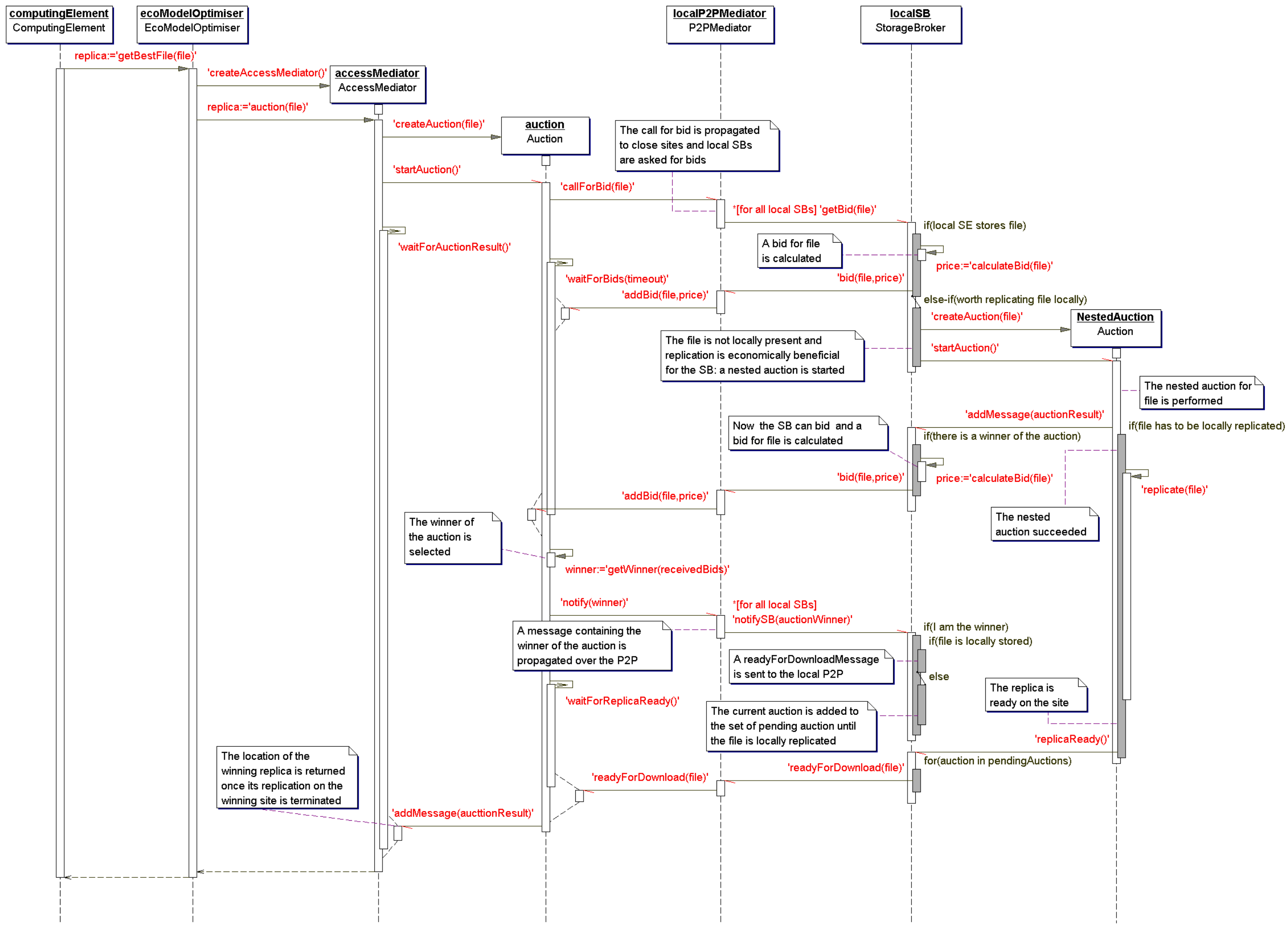

Figure 3: A Reverse Vickrey auction is used to locate a file.

When the winner has been notified, the auction thread waits until the winning replica is ready on the site[2]. After any replication process on the winning site has completed, the auction thread is notified via a *ReadyForDownload* message and the physical location of the winning replica is returned to the CE via the AM.

Once a SB receives a *BidRequest* message, it first checks if its SE stores the required file. If the file is present, it calculates a bid[3] and replies with a *BidReply* message. The local P2PM gathers *BidReply* messages from all local SBs and forwards them to the P2PM on the site that started the auction. This P2PM will forward the messages to the corresponding RO agent.

If the file is not stored locally, the SB might start a nested auction. The purpose of this nested auction is to create, on the corresponding SE, a replica of the requested file. If the SE has space, the SB will always start a nested auction. Otherwise, a nested auction is only started if the SB decides that having a local replica of the file is economically beneficial. The details of this decision process is described in Section 4.3.

Once a nested auction has successfully terminated, the SB calculates a bid for the file and take part in the original (or parent) auction. It is worth emphasising that we separate bidding in the parent auction from any replication activity. As a result, if a SB wins an auction it must check that any corresponding replication has finished. If the replication has not completed, the current auction is added to the set of pending auctions. Only once the replication has finished is the parent auctioneer notified that the winning replica is ready for access.

There is an important difference between first level (parent) and nested auctions. The goal of the former is to locate the cheapest replica required by some job executed in a CE. The best replica might be located either on the same site as the CE or on any remote site. The latter always aims to replicate a file to the local SE, as this will increase the SB's expected future income. In other words, the mechanism underlying

[2] This because a SB can bid for a file even if the related SE doesn't store a replica of the requested file. See below for details.
[3] In the current implementation, the bid corresponds to the transfer time between the SB's site and the site that started the auction, based on the current available network bandwidth.

the auction protocol performs long term optimisation allowing automatic replication towards "data hot-spots". Also, nested auctions allow replication to third party sites: sites where the file was not initially needed. Third party replication appears to provide a good mechanism for distributing required files amongst a neighbourhood of nearby SEs. It reduces the bottleneck caused by only considering close SEs (those located on the same site as the CE) for replication.

4.3 Prediction Function for Future File Value

In our model, nested auctions may trigger *Dynamic Replica Optimisation* (see Section 3.3), and thus over time vary the set of files stored on a SE. The decision about whether or not a nested auction is started depends on whether the local replication of the file is economically beneficial for the SB. Ideally, a SB should keep those files on the local SE that maximise the SB's expected future revenue: if the local replication of a new file increases SB's expected income, then a nested auction should be conducted.

In the presented model, we use a prediction function for estimating the future revenue of data files. We assume that the file access history can be represented by a random walk in the space of files. In the random walk, the identifier of the next requested file is obtained from the current identifier by the addition of a step, the value of which is given by a *binomial distribution*. The function uses the history of file requests (*access patterns*) experienced by the SB in a time window T' in the past and returns the predicted number of times a file will be requested in a time window T in the future. We refer to [5] for a more detailed discussion on the prediction function used in our economic model.

5 Experimentation

5.1 Configuration and Goal of Simulation

The authors have developed OptorSim [2] to study various replica optimisation algorithms of generic Grid jobs. In this paper, we simulate a set of high energy physics analysis jobs based on the CDF experiment use case (as described in [11]). In this case each file has size of 1 GB and the total size of the file set is 97 GB. The Grid topology (see Figure 4) comprises 20 sites in Europe and the USA that were used during a world wide data production from the CMS experiment [12]. To simplify the simulation it is assumed there is no contending network traffic. The goal of our simulation is two-fold. First, we want to verify our replication algorithm against a real world use case. Second, these simulation results serve as the bases for understanding large scale scientific analysis in general and in particular assisting the CMS experiment in resource planning for its next data challenge in 2003/2004. In this data challenge up to 150 TB of data will be analysed in a distributed manner over some 50 to 100 sites [12].

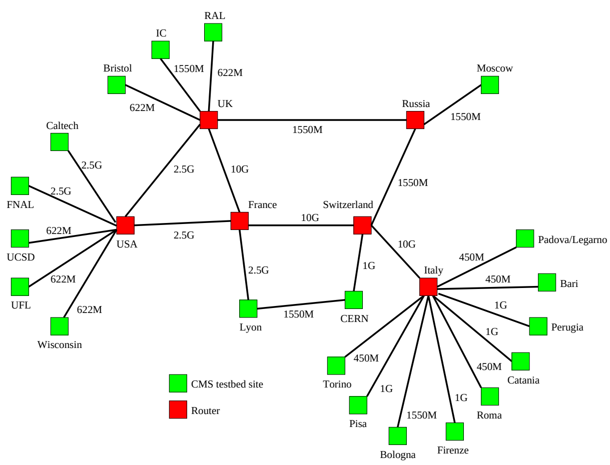

Figure 4: Grid topology for CMS world wide data production challenge in spring 2002.

In this paper, we consider the current CMS testbed in which the total storage capacity of all SEs is 1.1 TB. The SEs at CERN and FNAL have a capacity of 100 GB each and the remaining 18 testbed sites depicted in Figure 4 have a storage capacity of 50 GB each.

We assume that initially each file has only one physical instance referred to as *master copy* [9]. For the simulation, we considered two initial file distributions:

- All master copies are stored either at CERN or FNAL
- All master copies are stored either at CERN or FNAL and replicas are randomly distributed among all testbed sites until all the SEs are full.

5.2 Access Patterns

The order in which a job requests files is determined by the job's access pattern. We consider the following access patterns: **sequential** (all files are requested in a predetermined order) and **Gaussian random walk** (successive files are selected from a Gaussian distribution centred on the previous file).

5.3 Simulation Results

An indication of how effective the algorithm performs is the estimated time taken to complete all the jobs in a simulation run. It is important to note that there are other metrics for replica optimisation such as fault tolerance [9] or security issues. However, in this paper we wish to evaluate our optimisation strategy with respect to overall job throughput.

In each simulation run 1000 jobs (of various types) were executed. Jobs were submitted at five second intervals and the job-type was determined from a fix probability distribution, so some job-types were more likely than others. The estimated time taken to complete a job was calculated as the time waiting in the queue at the CE plus the execution time on the CE.

The duration of an auction (as used for replica location) is critical to the performance of the economic model. If an auction takes too long, the time taken to find the best replica becomes significant with respect to the file transfer time. If too little time is allowed, we might not receive all the possible bids. We achieved the best results with a 200 ms timeout, reducing this by 0.4 for nested auctions. These values allowed the SB enough time to conduct a nested auction and then return a bid to the parent auction.

We compare our economic model to a simple replication optimisation algorithm, *LRU (Least Recently Used)*. In this algorithm, files are always replicated to the site on which the job is executing. If the SE is full, the file that has been accessed the least number of times in the previous time interval δt is deleted. We call this the replica replacement decision. In the economic model, if the SE on the site on which the job is executing has space, files are always replicated to it. If there is no space left on the SE, the replica replacement takes place only if this is considered as economically beneficial for the SB (see Section 4.3) and the least valuable file is deleted to make space.

The results comparing the two algorithms for each access pattern and initial replica population policy are shown in Figures 5 and 6. Each bar shows the total job time, averaged over eight simulation runs. Figure 5 shows results for when there are no replicas at the start of the simulation and Figure 6 is when all the SEs are filled with replicas.

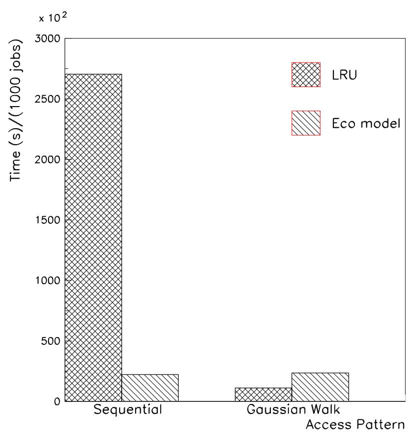

Figure 5: Total job times when there are no replicas at the start of the simulation.

The results show that the economic model provides substantially better throughput for jobs that have a sequential access pattern (typical of most High Energy Physics applications). This is mainly due to the inability of the LRU algorithm to optimise replicas when a job requests more files than can be stored on the local SE. Under these circumstances, when the algorithm replicates it deletes files that will be required at some

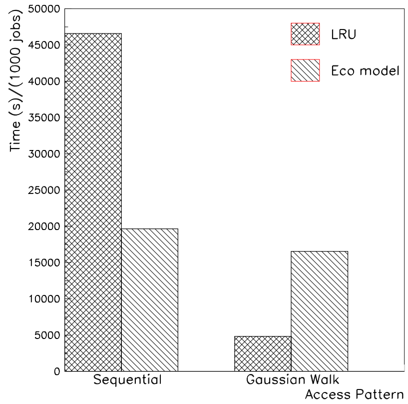

Figure 6: Total job times when all SEs are initially filled with replicas.

future point. It then will have to replicate the deleted file again when it is required by some job. However, the economic model, through use of the prediction function, learns that these files will be needed in the future and will not delete them. This will result in some files be read remotely, but it will reduce the overall network usage (by removing repeated replication) and hence jobs run faster.

The economic model is designed to work with sequential access patterns, so although a Gaussian walk access pattern results in files being requested more than once (hence the job execution time should be smaller), the economic model shows no improvement over sequential access patterns.

The LRU algorithm shows a significant improvement for Gaussian walk access pattern jobs.

A Gaussian walk access pattern allows for files in the set to be accessed more than once, whilst others are never accessed. Even if the set of files a job could potentially access would not fit on the local SE, the set of files selected by the Gaussian random walk probably would. Hence, the problem encountered by the LRU algorithm with sequential access jobs does not arise.

To conclude, the economic model shows significant performance improvements for sequential access patterns. Even though for Gaussian walk access patterns LRU performs better, the economic model is more robust against jobs accessing files with various patterns.

6 Conclusions

In this paper we presented a novel optimisation technique, based on an economic model, and discussed its use in a Data Grid to optimise both replica selection and dynamic access-pattern-driven replication. We detailed the reverse Vickrey auction protocol that the optimisation agents use for dynamically selecting the best replica of a requested file. This considers both data locality and network latencies. We also discussed the prediction function the agents use for making replication decisions, which uses historical data about file access patters.

To evaluate the efficiency of our algorithm we ran the Grid simulator OptorSim configured to represent a real-world Data Grid testbed from High Energy Physics, using job throughput as a benchmark. We studied the impact of different file access patterns on the performance of the economic model and compared our results with a traditional replication algorithm. The results clearly show that, for some access patterns, the economic model based replication algorithm shows a markedly improved performance compared to the traditional algorithm. This is due to its ability to prioritise files based on file access history and to replicate accordingly.

As part of our future work we plan to extend our simulation framework and evaluate our results against typical access patterns from other large-scale analysis efforts.

Acknowledgements

The authors thank D. Bosio, J. Casey, E. Laure and H. Stockinger for valuable discussions during the preparation of the paper.

References

[1] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Design of a Replica Optimisation Framework. Technical report, CERN, Geneva, Switzerland, 2002. EU DataGrid Project.

[2] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Simulation of Dynamic Grid Replication Strategies in OptorSim. In *Grid 2002, LNCS 2536*, Baltimore, MD, USA, November 2002. Springer.

[3] R. Buyya, R. Murshed, and D. Abramson. A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids. In *Int. Conf. on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, USA, June 2002.

[4] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. In *SPIE's Int. Symp. on the Convergence of Information Technologies and Communications (ITCom 2001)*, Denver, CO, USA, August 2001.

[5] L. Capozza, K. Stockinger, and F. Zini. Preliminary Evaluation of Revenue Prediction Functions for Economically-Effective File Replication. Technical report, DataGrid-02-TED-020724, CERN, Geneva, Switzerland, July 2002. EU DataGrid Project.

[6] M. Carman, F. Zini, L. Serafini, and K. Stockinger. Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid. In *Int. Workshop on Agent based Cluster and Grid Computing at Int. Symp. on Cluster Computing and the Grid (CCGrid 2002)*, Berlin, Germany, May 2002. IEEE-CS Press.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23, 2001.

[8] C. Ernemann, V. Hamscher, and R. Yahyapour. Economic Scheduling in Grid Computing. In *Job Scheduling Strategies for Parallel Processing, LNCS 2537*, Edinburgh, Scotland, July 2002. Springer.

[9] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, June 1996.

[10] W. Hoschek, J. Jean-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data Management in an International Data Grid Project. In *IEEE/ACM Int. Workshop on Grid Computing (Grid'2000)*, Bangalore, India, December 2000.

[11] B. T. Huffman et al. The CDF/D0 UK GridPP Project. CDF Internal Note. 5858.

[12] V. Lefebure and Tony Wildish (editors). The Spring 2002 DAQ TDR Production. CMS Internal Note. 2005/000, Geneva, Switzerland.

[13] N. Nisan, S. London, O. Regev, and N. Camiel. Globally Distributed Computation over the Internet - the POPCORN Project. In *Int. Conf. on Distributed Computing Systems (ICDCS'98)*, Amsterdam, The Netherlands, May 1998. IEEE CS-Press.

[14] K. Ranganathana and I. Foster. Identifying Dynamic Replication Strategies for a High Performance Data Grid. In *Proc. of the Int. Grid Computing Workshop*, Denver, Colorado, USA, November 2001.

[15] K. Ranganathana and I. Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In *International Symposium of High Performance Distributed Computing*, Edinburgh, Scotland, July 2002.

[16] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, March 1961.

[17] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2), 1992.