Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid

Mark Carman, Floriano Zini, Luciano Serafini
ITC-irst
Via Sommarive 18
38050 Povo (Trento), Italy
{carman, zini, serafini}@itc.it

Kurt Stockinger
CERN
Geneva, Switzerland
Kurt.Stockinger@cern.ch

Abstract

We are working on a system for the optimised access and replication of data on a Data Grid. Our approach is based on the use of an economic model that includes the actors and the resources in the Grid. Optimisation is obtained via interaction of the actors in the model, whose goals are maximising the profits and minimising the costs of data resource management. In the system, local optimisation results in global optimisation through emergent marketplace behaviour. In this paper we give an overview of our model and present part of the complex economic reasoning required to support this desired marketplace interaction model.

1. Introduction

In a typical Grid environment, where many users are sharing limited amounts of computing and storage resources, the optimisation of resource usage is very important in order to guarantee reasonable execution time and/or cost of users' tasks as well as fairness among them. Such environments are typically highly heterogenous and the resources themselves are dynamic in nature. In the case of the so-called Data Grids, there is the added problem of needing to manage vast quantities of data (up to several Petabytes) [5]. Here, the main challenge is the improvement of data access efficiency given the limited number and size of storage devices available, which in turn constrains the amount of data replication that can be carried out.

In a Data Grid a user typically submits a *job* to the Grid from her workstation, which is located at a particular site on the Grid, and requires that the job be executed as fast as possible[1]. To execute, a job basically requires three kinds of resources: computational resources, data resources, and network resources. Ideally, a Grid optimisation service should be able to manage the usage of these resources in order to bring the needs of a single user into agreement with the demands of the whole community of Grid users. Optimisation should be carried out based on the status of Grid resources (workload and features of computation sites, location of data, network load) and should result in the allocation of a convenient site for job execution, as well as the allocation of a convenient replica of the job's input data (possibly involving dynamic replication of data between Grid sites).

In this paper we focus on a particular aspect of optimisation and deal primarily with the problem of optimising the replication of data in a Grid environment, that is, with deciding when and where to create and delete replicas of data files. The aim here is to minimise the overall cost of file access on the Grid in the "long-term" [3], given a finite amount of storage resources. We do however, also deal with the complementary problem, which is that of selecting the optimal replicas of data for use by a job currently executing within the Grid environment, (which we referred to as the "during job execution" optimisation in [3]).

We do not tackle the problem of job scheduling on the Grid, i.e. the problem of deciding where and when to schedule jobs for execution. We assume that jobs are dispatched for execution to different sites on the Grid by some scheduling system, which uses knowledge of the available computational, data, and network resources to make "rational" scheduling decisions. Even though job scheduling and replica selection are related, for simplicity of our model we make the assumption that our optimisation starts when the scheduling decision has already been made and the job has started (or is about to start) execution on a particular site.

We propose a fully distributed optimisation of data access and replication, based on an economic model for the interaction of different optimisation units at each node/site on the network. The main focus is on optimising local re-

[1] Note that for simplicity we assume that a job is atomic and can thus not be decomposed into subjobs.

sources and thereby achieving global optimisation through emergent marketplace behaviour. In our model data files represent the goods in the market. They are purchased by computing elements for running jobs and by storage elements in order to make an investment that will improve their revenues in the future. They are sold by storage elements to computing elements and to other storage elements. Computing elements try to minimise the file purchase cost, while storage elements have the goal of maximising profits. We propose that computing and storage elements be wrapped by intelligent optimisation components which perform the reasoning required in our model.

The document is structured as follows. We first argue the case for using an economic model to optimise data access in a Grid environment. We then describe the Grid environment in more detail, and describe the components of our "economic model of the environment". The next sections detail our work towards designing these components, by defining the marketplace interactions between them and the economic reasoning contained in them. We finally discuss some related works and then conclude the paper with future developments.

2. Why an Economic Model?

The primary reason for employing an economic model is to be able to make optimised replication decisions in a distributed manner. Performing such complex multidimensional optimisations in a centralised manner is very difficult, as the planning domain (attributes of the resources being controlled) is very large. Moreover such centralised systems do not scale well. The Grid optimisation service must be scalable in terms of both the number of network nodes (tens, hundreds, or even thousands) and the volume of data on the network (hundreds of Terabytes or even Petabytes of data). By restricting ourselves to local optimisations we make the "reasoning" problem far more manageable, and by allowing economic interactions to lead the system toward global optimisations through emergent marketplace behaviour, we also make the system optimal.

The optimisation service needs also to be both reliable (it should reoptimise its configuration to account for network/device failures) and robust (no single point of failure should be capable of "bringing down" the system). Optimisation based on an economic model is fully distributed in nature and thus will be both reliable and robust.

The Grid is a highly dynamic environment in which the status of resources can change without warning. Thus it is extremely important to be able to perform dynamic (re)optimisation while jobs are executing. By using an economic model we can exploit the dynamism of the market to make more informed decisions at job execution time. Therefore, we do not need to make an a-priori irrevocable decisions (at job scheduling time) about which file replicas we will use to access data for a particular job. (In some cases one will not even know what data will be required by the job until it is actually executing.)

In general, a science Grid is set up by various *Virtual Organisations*, that contribute differently to its establishment. Moreover, each Virtual Organisation is made up by members from several *Real Organisations* that, in turn, invest different amounts of resources to build up the Grid. It is likely the case that virtual and real organisations need a way to regulate the use of the Grid according to the effort that each of them made to establish it. An economic model seems to be an accurate representation of this reality and a suitable way to regulate the interaction between organisations. Even if no real money is involved in assigning resources (as it is the case in many virtual organisations), every user could be given an amount of "virtual currency" that is proportional to his priority and importance within the virtual organisation. Such general "accounting models" are currently under investigation by a number of groups including [2].

3. The Grid Environment

We basically adopt the same view of a Data Grid as that proposed by the European Data Grid [1] project[2]. The Data Grid consists of following principal resources:

- A *Computing Element* is an abstraction for any *computer fabric*. It provides the Grid users with CPU cycles for job execution, as well as an interface for job submission and control on top of services offered by the computing fabric. Each Computing Element is located at a particular "site" on the Grid.
- A *Storage Element* is an abstraction for any storage system (e.g., a mass storage system or a disk pool) and provides Grid users with storage capacity. The amount of storage capacity available for Grid jobs varies over time depending on local storage management policies that are mostly not under the control of the Grid.
- The *Network* provides Grid users with bandwidth for data transfer form a Storage Element to another one and from a Storage Element to a Computing Element. There is a limited amount of such bandwidth and the "cost" associated with the use of it forms an important input to our model.

We assume that during execution on a Computing Element jobs can access files by staging the files to local disk and that

[2]The project aims at building a Data Grid for High Energy Physics (HEP), Earth Observation and Bioinformatics applications, and potentially for many other data-intensive science applications.

all data is physically organised into files that are constant in size and read only in nature, as is primarily the case for the European Data Grid project. We also assume that we are always able to replicate files from one site to the next and thus we ignore all issues concerning the importing and exporting of files into and out of database implementations. Finally, we assume that jobs access files in a random but not uniform fashion (i.e. some files are in more demand than others).

We may wish to model the fact that different users have different privileges and priorities for using the Grid. For example, within a virtual organisation such as the HEP community it will most likely be the case that students will be able to use the Grid to perform experiments only at a very low priority, while first order scientists will be able to exploit it to its full potential. This situation could be modelled (similarly to [4]) by assigning amount of "grid credits" to users that is proportional to their level of priority. The difference, however, of our approach is to optimise file access based on replicated data rather than job scheduling.

4. The Components of the Model

The components in the model are shown in Figure 1. By the term *Replica Optimiser* we intend a Grid service that performs the optimisation of data access and replication, as described in [3]. In the following we detail the components of the model.

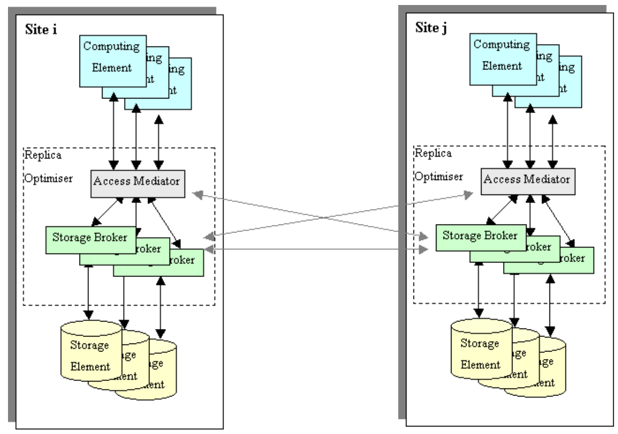

Figure 1. The components in the model.

Computing Element. Computing Elements at each site access files via a standard file services interface provided by an Access Mediator Component. For the sake of this paper, we are not interested in modelling the internal behaviour of Computing Elements and thus we simply model them as a stream of requests for file access. Requests may include a maximum amount to be payed for the required file access (which would depend on the type of user making the request). If available, this value would be used by the Access Mediator to decide when a job's data request cannot be fulfilled and in such cases inform the Grid Scheduler such that it could cancel or possibly reschedule the job to another location.

Access Mediator. When a Computing Element attempts to get a file on the local disk, the Access Mediator must locate and make available a copy of the required file. It does so by interacting with Storage Brokers on the same site and other sites with the intention of accessing a copy of the file at the lowest possible price. The Access Mediator can be viewed as a rational agent, who's task it is to procure file access services at minimum expense to the Computing Elements demanding them.

Storage Broker. Storage Brokers represent Storage Elements to Access Mediators. The primary task of each Storage Broker is to control the set of replica files which are stored on the Storage Element it looks after. A Storage Broker makes decisions as to which files to replicate locally in order to maximise the utility of the storage space available to it. Thus each Broker can also be viewed as a rational (utility maximising) agent, which controls a finite set of dynamic resources (storage capacity), under a dynamic set of constraints (policy set by the particular storage resource owner).

Storage Element. A Storage Element is modelled as a finite amount of storage space. The amount of space available to the Grid varies dynamically according to the number of internal (Grid) and external (non-Grid) users storing data on the device. Local policy (set by the device owner) may determine the level of priority given to each type of user. Storage Elements execute cleanup (recovery of space) autonomously by removing files of "lowest importance" from storage. (A file's importance level may be set by the Storage Broker.)

5. Marketplace Protocols

In an open economy, optimal resource allocation results from an efficient marketplace, in which suppliers compete in perfect competition with one another. The question for us becomes one of maximising competition in the simulated marketplace. Since we are dealing with a commodity marketplace, we can guarantee maximum competition by maximising price transparency. In the simulation we do this by introducing Auctioning Protocols.

We will use a form of reverse auctioning (buyer driven auctioning) to decide upon prices in this environment. Importantly, sellers in a reverse auction become "price takers", which means that each Storage Broker does not need to do any complicated reasoning as to the price to offer a particular file for. Instead it simply offers the file at the price requested if it is able to do so, given the file transport cost to the location requesting it. Notice that Storage Brokers do not have any control over network cost, that is for simplicity assumed to be costant per unit of data and unit of time.

There are a number of drawbacks associated with the use of an auctioning mechanism, namely the messaging overhead, the time delay associated with running an auction, and required decision making as to how "widely" to publish requests. We are presently experimenting with P2P inspired auctioning techniques to deal with some of these problems.

6 Overview of Economic Decision Making

The goal of the Storage Broker is to maximise the utility of storage space available to it, where utility is defined in terms of profit associated with the use of the storage device. In this section we present an overview of a mathematical model that we intend to use for Storage Broker decision making. A Storage Broker will use the reasoning presented here to decide whether or not to buy a file when presented with the opportunity to do so.

6.1 Storing Information

We assume the Storage Broker records all filled file requests (to which it is a witness) in a log. Each filled file request is considered to be an n-tuple of the form:

$$FR_i = \langle t_i, o_i, g_i, n_i, r_i, s_i, p_i \rangle \tag{1}$$

where t_i is the timestamp at which the request was filled (i.e. the requested file was purchased); o_i, g_i, n_i together represent the (logical) file being requested, where o_i is the virtual organisation to which the file belongs, g_i is the group from which the file comes and n_i is the identification number of the file within the group. This particular model is inspired by the organisation of the High Energy Physics experiments [3].

In each tuple, r_i and s_i represent the element requesting and element supplying the file respectively; while p_i represents the price actually paid for the file. In the following we will use the abbreviation F to represent the triple (o, g, n) and F_i to represent (o_i, g_i, n_i).

6.2 Maximising Profit

In order to design profit (utility) maximising agents, we need first state clearly what it is we mean by the value of a file. We define the (future) value of a file in storage at time t_k as the sum of the future payments that will be received (by the broker) for the use of the file over the period T_{av} (which is the average lifetime of files in storage). As an equation this can be written:

$$V(F, t_k) = \sum_{i=k+1}^{k+n} p_i \delta(F, F_i) \delta(s, s_i) \tag{2}$$

where $t_{k+n} \leq t_k + T_{av} < t_{k+n+1}$ and s is the local Storage Element. (The δ function returns 1 if the arguments are equal and 0 if they differ.)

We define the "investment" cost U (for file F at time t_k) as the difference between the price paid for the new file by the Storage Broker, pp_k, and the price received by the broker when the file is immediately on-sold, p_k. (The file is on-sold to the originator r_k of the request k which triggered the current investment.)

We assume there is only a finite amount of storage space available. Thus if a new file is added to the storage device, an old file must be deleted. So the investment decision becomes a choice between investing in (purchasing) a new file F' and keeping an old file F (i.e. a file which is currently in storage). To make such a decision we need to have a look at the change in profit ΔP which will result from a decision to invest in the new file F'.

$$\Delta P(F', F, t_k) = V(F', t_k) - U(F', t_k) - V(F, t_k) \tag{3}$$

Note the fact that the investment cost, U, for file F is not included in the calculation, because this cost has already been paid in the past (at $t < t_k$) and therefore does not affect the current investment decision (at $t = t_k$). We could now assume that a "rational broker" would decide to invest in a new file F' if the ΔP value calculated above was positive.

6.3 Risk Averse Investment

We now consider a "rational broker" that makes investment decisions by taking into account also the risks involved in a particular investment. We do this by first extending equation 3 to calculate the change in profit generated by the investment up until the time t. I.e. we are now calculating the change in profit which would result from the investment at time t_k, given that the new file F' will be removed from storage at time t, due to a subsequent investment decision. (Note that ΔV in this equation is merely a shorthand way of writing the difference between the value of files F' and F at time t_k.)

$$\Delta P(t) = \frac{t - t_k}{T_{av}} \Delta V - U \text{ for } t_k < t \leq t_k + T_{av} \tag{4}$$

We see from the equation that as the difference between t and t_k increases the return on the initial investment U increases, and with it the overall profit. What we want to do now is calculate the probability of this ΔP value being greater than zero, (i.e. of having a positive return on investment). In order to do so we need first make assumptions regarding the statistics of the file request arrival process.

We model the inter-arrival time for "worthwhile file investments" as a Poisson process. I.e. the time before the arrival of the next investment opportunity (and thus the removal of the file F' from storage) conforms to an exponential distribution. Thus the probability of a file still being in storage at time $t = t_k + \Delta t$ is given by: $Prob(\tau \geq \Delta t) = e^{-\Delta t / T_{av}}$.

Now if we substitute this probability equation into equation 4 and apply the constraint that the resultant ΔP should be greater than zero we find:

$$\frac{\Delta V}{U} > \frac{-1}{\ln Prob} \tag{5}$$

A broker can then use this equation to decide whether or not to invest in a new file. The probability value $Prob$ gives the probability that the investment will result in a positive change in profit. A broker could, for example, have a conservative investment policy whereby it only invests in new files if it is 80% convinced of a positive return on that investment. Thus according to the equation it should only invest in a new file if the ratio of the return on the new file ΔV to the investment in it U is greater than 4.48.

A more complicated investment rationale can also be supported. In the real world, "risk averse" humans demand higher guarantees of return $Prob$ for higher investment risks U. We can model such behaviour in the brokers by using equation 5 to calculate the probability value $Prob$ for a particular investment, and then feeding this value and the investment cost U into a second equation $f(Prob, U)$ which would compare the two values and return positive if an investment is "recommended". The defining of such an equation is, however, left to future work.

6.4 Predicting Returns

The important problem to deal with now is that of predicting future returns (values) of files given only past information as input. To make such predictions we exploit three important heuristics of the file request process, namely:

1. Temporal correlation (requests for the same file are clustered in time).

2. Geographic correlation (requests for the same file are clustered in locality and time).

3. Sequential correlation (requests for files containing similar/sequential data are clustered in locality and time).

In order to exploit these heuristics we define a function for the expected value of a file. The expected value of a file is given as the weighted sum of the prices paid in the past for that file (or files similar to it). The values c_i in the following equation represent "constant" coefficient values (see the next section for a discussion of how they are calculated).

$$E[V(F, t_k)] = c_3 f_1(r_k) + c_0 \sum_{i=1}^{k-1} \{p_j + c_1\} S(F, F_j) D(r_j, s_j) e^{-c_2(t_k - t_j)}|_{j=k-i} \tag{6}$$

The summation in equation 6 cycles over the requests stored in the history log. For each request the price paid (plus a constant) is scaled according to a "similarity function" S, a "distance function" D and an exponential time decay function. The similarity function, as the name suggests, compares the file of interest F with the file from the log F_j (trying to exploit sequential correlation between them). The distance function takes into account the locations that requested and supplied the file (exploiting geographic correlation). Finally the exponential term in the equation causes older information to be weighted with lower importance than more recent information.

The first term of equation 6 can be seen as a "bonus" component which takes into account the distance to the source of the incoming file request.

6.5 Improving Predictions by Learning

The coefficients c_i used in the equations to predict future values of given files, need to be calculated themselves based on data concerning the heuristics of the file request distribution. Our intention is to use offline standard data to find reasonable (bootstrap) values for these coefficients, and then to enable online refinement and continual improvement of the values based on historical data. This process is very important as we assume that not only will the distribution of requested files change over time (i.e. that the popularity of individual files will vary), but that the heuristics of the distribution of file requests will also change (e.g. the total number of popular files will vary over time). The process for refining the coefficients is as follows. Cycle over historical (log) data from $(t_n - T_{av})$ backwards checking all of the decisions made to purchase a particular file. For each decision, we compare the estimated value (calculated using equation 6) with the actual value (calculated using equation 2). If the values differ "significantly" then other coefficient values are tested in equation 6 in an attempt to improve the estimate. The improved coefficient values c_i are recorded, and statistical processes (at present relegated to future work) will be used to calculate the "best" coefficient values over the set of past decisions.

7. Related Work

The development of Grid optimisation services is crucial for the success and spread of Grid technology but, due to its novelty, research in this field is still at the beginning and is oriented toward particular aspects of the problem. An example is [6], which focuses on the evaluation of replication and caching strategies within a simulated Grid environment. As we do, they face the problem of optimising data shipping once a site for execution of a Grid job has been chosen. The paper compares different strategies (e.g., no replication, replication of files that are most requested, fast spread replication along the path between the requesting node and the source of data) against file access pattern ranging from completely random access pattern to access patterns with a high degree of geographic and/or temporal locality. The lesson learned from the comparison is that there is not a strategy that is definitely superior to the others, but their performances notably vary depending on the file access pattern.

Another research in the field, which deals more with scheduling than replication strategies, is presented in [7]. *I/O communities* are the basic concept introduced by the paper. An I/O community is defined as a cluster of execution and storage sites that participate in the wide-area Grid system. The idea is that Grid computations should be performed mainly within the boundaries of a community, that should include (most of) the resources the job needs. The paper presents a matchmaking system that takes as input a job specification and tries to find the I/O community which contains the resources it needs, according to its requirements. An I/O community is set up by adding to the description of execution elements one or more properties that define the name of the storage elements that belong to the community. Since this is done by hand by the administrator of the resource, there is no way to dynamically adjust communities and thus the performance of the system depends on a reasonable setting of the communities at the beginning.

Both approaches stated above tackle the problem of resource optimisation in a relatively "static" way. In a real data Grid, however, file access patterns or, more generally, resource request patterns vary quite unpredictably over time and thus some mechanism that automatically adjusts optimisation strategy seems to be essential. Our opinion is that an economic model may be the basis of a self-regulating optimisation strategy, that dynamically adapts to the changes in the resource request pattern.

8. Conclusions

In this paper we have presented the basis of a service for optimization of file access and replication in a Data Grid. The core of our system is an economic model that regulates the sale of files between rational agents that wrap storage and computing elements in the Grid. Our economic model allows for short term optimisation (the optimised access to a particular file - the goal of the Access Mediator) and long term optimisation (optimal distribution of file replicas - the goal of the Storage Broker) to be achieved concurrently and as a consequence of the economic interaction between resources. We have described the model that is used by a storage broker agent to make rational decisions about whether or not to purchase a file.

Our goal is now to practically demonstrate the feasibility of our approach by performing experiments in an agent-based simulated Grid environment. We have already implemented a preliminary version of the simulator and we are going to extend it by incorporating the economic model presented here.

Acknowledgements. We wish to thank Paolo Busetta for useful discussions regarding the structure of the economic model presented in this paper.

References

[1] EU Data Grid project. http://www.eu-datagrid.org.

[2] C. Anglano, S. Barale, L. Gaido, A. Guarise, S. Lusso, and A. Werbrouck. An accounting system for the Data Grid project - Preliminary proposal v2.0, September 2001. http://www.to.infn.it/grid/accounting/.

[3] P. Busetta, M. Carman, L. Serafini, K. Stockinger, and F. Zini. Grid query optimisation in the Data Grid. Technical Report IRST 0109-01, Istituto Trentino di Cultura, September 2001.

[4] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic models for management of resources in peer-to-peer and grid computing. In *Commercial Applications for High-Performance Computing, SPIE's International Symposium on The Convergence of Information Technologies and Communications (ITCom 2001)*, Denver, CO, August 20-24 2001.

[5] W. Hoschek, J. Jean-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data management in an international data grid project. In *1st IEEE/ACM International Workshop on Grid Computing (Grid'2000)*, Bangalore, India, December 17-20 2000.

[6] K. Ranganathan and I. Foster. Identifying dynamic replication strategies for a high performance data grid. In *Proc. of the International Grid Computing Workshop*, Denver, CO, November 2001.

[7] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Gathering at the well: Creating communities for grid I/O. In *Proc. of SC2001*, Denver, CO, November 2001.