

Access Cost Estimation for Unified Grid Storage Systems*

Kurt Stockinger¹, Heinz Stockinger¹, Lukasz Dutka^{2,3},
Renata Slota³, Darin Nikolow³, Jacek Kitowski^{2,3}

¹ CERN, European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland

² ACC Cyfronet-AGH, ul. Nawojki 11, Cracow, Poland

³ Institute of Computer Science, AGH-UST, al. Mickiewicza 30, Cracow, Poland

Abstract

In typical Data Grids large amounts of replicated data are stored all over the globe in different storage systems with access latencies ranging from seconds to hours. The task of a replica management system is not only to keep track of the replicas but also to select those replicas that can be accessed by an application program with a minimal response or transfer time. Most wide-area replication research focuses on network-based replica selection. However, our past experience with Data Grids has shown that often hierarchical storage systems are the main bottleneck rather than network links. This is due to the fact that access latencies of hierarchical storage systems can be of the order of seconds up to hours in case the data resides on a tape that is not mounted yet.

In this paper we give an overview of our replica management framework called Reptor and a storage system cost estimator that is used. Furthermore, we give details on access estimation of file replicas that reside on hierarchical storage systems. The results show that the access estimates provide a good basis for a replica management system to perform efficient replica selection.

1 Introduction

Certain scientific application domains such as High-Energy Physics or Earth Observation [9] are expected to produce several Petabytes of data that are analyzed and evaluated by scientists all over the globe. In order to achieve high levels of availability and fault tolerance, as well as

*This work was partially funded by the European Commission program IST-2000-25182 through the EU DataGrid Project, by the IST-2001-32243 project "CrossGrid" and by the Polish Committee for Scientific Research (KBN) 112/E-356/SPB/5.PR UE/DZ 224/2002-2004 project. AGH grant is also acknowledged.

minimal access times for these large data volumes, data replication is applied very frequently. Several Grid projects have implemented data replication systems that handle parts of the requirements.

Data is mostly stored on disks and mass storage systems with different access latencies. What is more, these storage systems are connected via networks with different effective bandwidth that show different peaks throughout the day. Without specific monitoring tools of various Grid resources, it is difficult to predict the access time of data intensive jobs in such a heterogeneous environment. For instance, access latencies of mass storage systems can be of the order of seconds up to hours in case the data resides on a tape that is not mounted yet.

Within the European DataGrid project [9] we designed and developed a replica management framework called Reptor [12] which provides both network and storage access estimation for distributed replicas. This performance information can be used by a Grid scheduler to submit jobs to the best available Grid resources. In addition, our system allows for optimal cost-based replica selection during the run time of a job.

In this paper we discuss one implementation of a storage access estimator that we developed within the European CrossGrid project [4]. We detail the replica selection model and the storage access cost parameters for estimating the access latencies of file replicas that reside on hierarchical storage systems distributed all over the globe.

We thus make several important contributions:

- Firstly, we show the effectiveness of replica selection based on storage cost estimation. The results show that the access estimates provide a good basis for the Replica Manager to perform efficient replica selection.
- Secondly, we demonstrate interoperability between two major European Grid projects using standardized

programming techniques based on web service technology.

The paper is organized as follows. In Section 2 we provide an overview of the cost model that is used for the replica selection process. Emphasis is put on the cost estimations for hierarchical storage systems. Section 3 outlines architectural details of the replica management framework (including the replica selection process) and the cost estimator based on a component expert system. Implementation details and interoperability issues are discussed in Section 4. Experimental results are shown in Section 5. After some related work in Section 6, we conclude the paper and provide some insight into future work.

2 Model for Cost Estimation

Access estimation for distributed and replicated files in a Data Grid requires a specific model that we describe here. This model is then used by the replica management system to select replicas as well as by the estimator components that provide the necessary information on file access costs.

2.1 A General Cost Model

For the access cost model we assume that data files are partially replicated to several sites (hosting data stores) connected via wide-area networks. In such an environment end-users usually want to access one of the replicas as quickly as possible, regardless of the replica location. As discussed in [12], it is up to the replica management software system to select the “best” file that has the minimal access costs. Based on previous work [18], we define replica selection as follows:

Replica selection is the process of selecting a single “best” replica from a set of identical replicas where best means that the response time for accessing a file locally is minimal. If replicas are only available at remote sites, the access time also includes the data transfer time from the remote to the local site.

We also assume that a file should be accessed locally at a given site and thus we want to minimize the transfer costs from a remote site to a given site. In [18] we identified two main performance parameters: network and storage system costs. We can now define optimal replica select as the following minimization problem:

$$\min(\text{file_transfer}_i(\text{site}_{\text{local}}, \text{site}_{\text{remote}})) \quad (1)$$

where the file transfer is defined as follows:

$$\text{file_transfer}_i = \text{access_cost}_{\text{network}} + \text{access_cost}_{\text{storage}} \quad (2)$$

In this paper, we only concentrate on the storage cost estimation. Note, for simplicity we assume that Grid applications request only single files and leave the case of multiple file requests for future work.

2.2 Access costs for HSM systems

In the following section, we outline the cost estimation for data servers, in particular for Hierarchical Storage Managers (HSM). Thus, we extend Equation 2 and give details about the access cost of storage.

The access cost for HSM systems depends on many factors, like current load of the HSM system, number of available drives and their types, localization of the data (disk cache, MOD, tape) and the data compression rate. We assume that the access cost, $\text{access_cost}_{\text{storage}}$, is defined by the start-up latency time, $\text{time}_{\text{latency}}$, and the transfer time, $\text{time}_{\text{transfer}}$, of the HSM system itself. The latter is defined as the time for making data locally available and it is needed for serving data to the Grid environment. The network latency is not part of the access cost estimation for HSM.

Generally, $\text{time}_{\text{latency}}$ for HSM can be broken into the following elements:

- t_w – **waiting time**: defines the waiting time for necessary resources to become idle. These resources can be a tape drive, the tape itself or the robot arm. The waiting time depends on the previous requests in the queue.
- t_u – **unmounting time**: time to unmount a tape in order to free a drive for the current request.
- t_m – **mounting time**: the time to mount a tape into a drive and get it ready. It depends on the robot and the drive performance.
- t_p – **positioning time**: the time of positioning from the current block to the first block (referenced as a destination block) of the file on a tape. It depends on the type of media, the current and destination block numbers.
- t_t – **media transfer time** from removable medium (tape or plate if the medium is an optical disk) to disk cache. It depends on the drive transfer rate and the file size.
- t_d – **disk cache start-up latency**. This is the latency experienced by the client when the file is in the disk cache. The client is assumed to be local, so the network latency is 0 seconds. It mainly depends on the hardware parameters of the hard disk drive (seek times).

For particular cases some values can be equal to 0. Those concern files located already in the caches, for which all times except t_d are equal to zero. The access time is also zero when the tape is mounted and idle, $t_w = t_m = 0$.

$time_{transfer}$ depends on the HSM disk cache transfer rate ($transfer_{rate_{cache}}$, the file size ($size_{file}$) and the access method (NFS or FTP).

The final formula depends on HSM type and for the Legato DiskXtender HSM system, for which we implemented the access cost estimation function, is as follows:

$$access_cost_{storage} = time_{latency} + time_{transfer} \quad (3)$$

where:

$$time_{latency} = t_w + t_u + t_m + t_p + t_t + t_d \quad (4)$$

and

$$time_{transfer} = size_{file}/transfer_{rate_{cache}} \quad (5)$$

3 Architecture

In the following section we briefly describe the architecture of the entire replica management framework that contains a replica optimization service (from EDG) as well as the HSM estimator (from CrossGrid).

3.1 Replica Management System

In EDG, we have designed and developed a replica management system that takes care of replicating files between several storage locations, locating replicas and selection “best” replicas as discussed in Section 2.1. The entire system uses several external services and is briefly depicted in Figure 1.

We do not go into the design details and refer the reader to [12]. Here we only concentrate on the components and services that are necessary for the replica selection process. It is important to point out that Figure 1 represents the user’s point of view and thus all interaction with services like a Replica Optimization Service is done via the replica management service interface.

For the discussion in this paper, one only needs to consider the following modules and components:

- The **Core** module co-ordinates the main functionality of the replica management system, which is replica creation, deletion, and cataloging by interacting with third party modules. These external modules include transport services, replica location services, meta-data services for storing replication meta-data such as file meta-data (size, checksum, etc), management meta-data, and security meta-data (such as access control

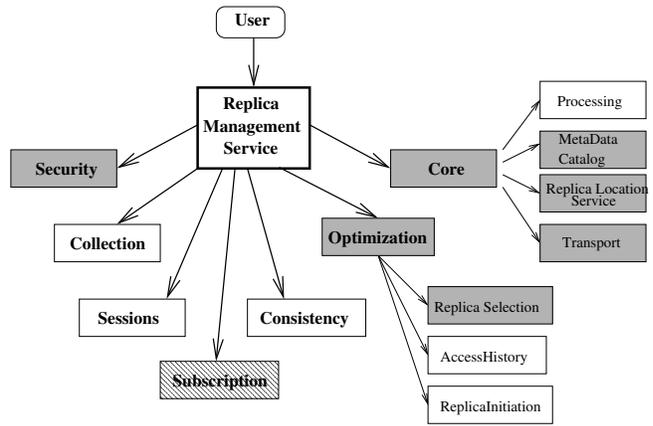


Figure 1. Main Components of the Replica Management System. Shaded components represented components that are implemented.

lists), and processing services that allow pre- and post-processing of files being replicated.

- The goal of the **Optimization** component (also referred to as the Replica Optimization Service (cf. Section 3.2) is to minimize file access times by pointing access requests to appropriate replicas and pro-actively replicating frequently used files based on access statistics gathered.
- The **Security** module manages the required user authentication and authorization, in particular, issues pertaining to whether a user is allowed to create, delete, read, and write a file.

3.2 Replica Optimization Service

The goal of the optimization service is to select the best replica with respect to network and storage access latencies. In other words, if for a given file several replicas exist, the optimization service determines the replica that should be accessed from a given location. Similarly, the optimization service might also be used to determine the best location for new replicas.

The *Replica Optimization Service* is implemented as a light-weight web service (called *Optor*). It gathers information from the European DataGrid (EDG) [9] network monitoring service and the CrossGrid [4] Cost Estimation (also referred to as “Storage Element monitoring”) service about their network and storage access latencies. Based on this information *Optor* takes the decision about which network link should be used in order to minimize the transfer time between two end points as described in [2].

Apart from selecting replicas and storage locations Optor also provides methods to retrieve estimated file access costs. These can be exploited by other Grid services, such as meta schedulers like the EDG Resource Broker [10]. Based on the information obtained by Optor the broker can schedule jobs to sites that allow efficient file access while maximizing the overall throughput. Thus, the Replica Manager, in particular its optimization component, assists the Resource Broker in the job scheduling process.

The interaction of the main components of the Replica Manager Reptor is depicted in Figure 2. For this paper, we are mainly interested in the interaction between the Replica Optimization Service and the SE Monitor/Cost Estimation Service. Note that Optor provides an interface called “get-SECost” that either receives information from SE Monitor or from SE Cost Estimator. In our current model, we use a Cost Estimator as described in the next section.

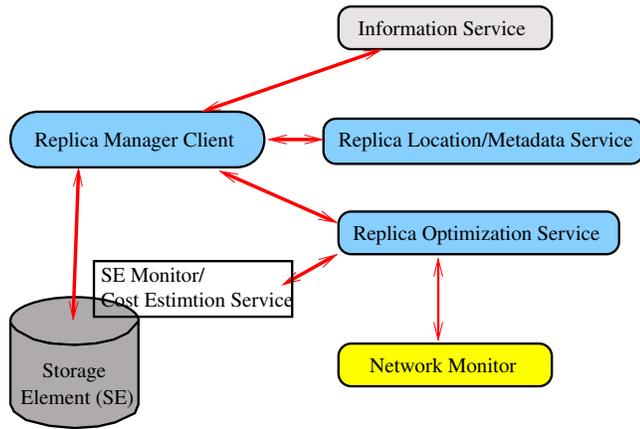


Figure 2. Interaction of Replica Manager with other Grid services.

3.3 Access Estimation for Hierarchical Storage Systems

In this section, we give details on a cost estimation service. As mentioned in previous sections, the cost estimation of the storage access is one of the most important parts in the entire replica selection process. However, estimation strategies strongly depend on many external factors, especially on device types, e.g. for Hierarchical Storage Managers (HSM) they are different from those for disk drives, or for databases in comparison with estimation for raw files. That fact is very important due to the heterogeneity of the storage systems used in Grid environments.

Since the internal organization of the storage system is hidden from the estimation service clients, there should be

an additional layer selecting the best estimator for a particular context, i.e. for each storage type there is a different estimator. To cope with that challenge a Component-Expert Architecture (CEA) [6, 8, 7] is proposed with the general scheme depicted in Figure 3.

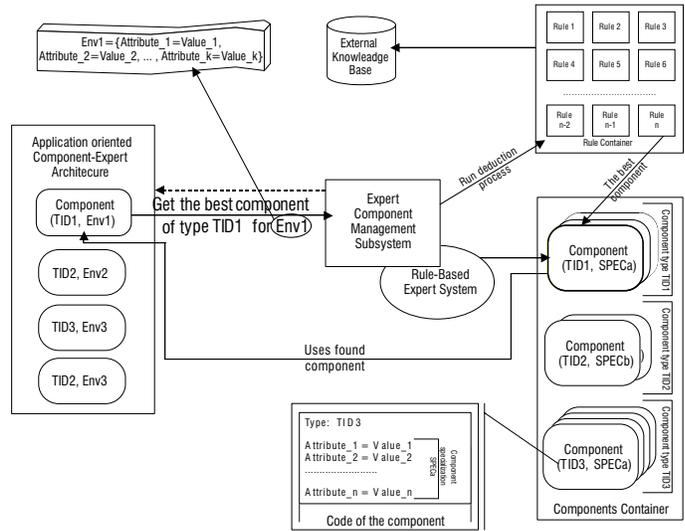


Figure 3. General connection diagram in Component-Expert Architecture.

The proposed CEA allows building very scalable systems based on independent components, which, similarly to popular plug-ins, can easily be registered in the system at any time. The most powerful feature of that architecture is the possibility to transfer the responsibility of the component selection. In the classical component architecture [19] the components are selected by developers during the development process and their choice is compiled into the code. In CEA the task of component selection is entrusted to an additional module called Rule-Based Expert System (see Figure 3). That module decides on-the-fly which component is the most suitable for a current context. Thus, the decision is taken by a previously prepared set of rules and is based on the following factors (see Figure 3): the current context (Call-Environment), a classification of all registered components prepared previously by the developers (Component Specialization) and additional information available from external knowledge bases. In practical implementations, e.g. [7], these rules first try to eliminate inappropriate components (components, that are specialized for other purposes) and finally choose the best one from all components fulfilling particular requirements. The development of the rules should be based on practical experience of a human expert; during the run-time the knowledge of the system can be extended by modification of the external knowledge base.

For the estimation purpose of the data access cost for Grid-enabled storage, CEA is used as a framework managing a set of estimators and taking over the responsibility for the selection of the most adequate estimator. The estimators are implemented as CEA components. They differ from each other by the component specialization (e.g., estimators for HSMs, disk drives, optical devices, etc.) but the decision concerning the best estimator is taken during the run time of the system by means of a previously prepared rule set.

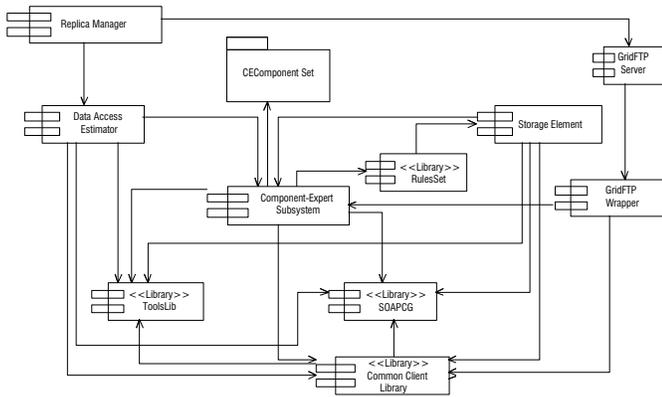


Figure 4. Component dependencies diagram in the CrossGrid environment applying a Component-Expert Architecture. The “Replica Manager” component refers to the Replica Optimization Service that acts as a client of the system.

CEA has been used in the CrossGrid [5] project to manage the estimators as well as to manage data access handlers. As a result, an extremely flexible and unified data access platform has been developed, which can easily be expanded by additional components (plug-ins). Furthermore, the newly registered components can immediately be exploited in the appropriate context. In Figure 4 the UML model of the dependencies between the components realizing the CEA in the CrossGrid environment is shown.

The ‘Component-Expert Subsystem’ component as well as the ‘RulesSet’ and ‘CEComponent Set’ packages are responsible for the selection of the most proper estimator or data handler depending on a context (Fig. 4). The ‘Storage Element’ component functions as the external knowledge provider. It provides all precise information on the current configuration and state of the storage device keeping the particular data object (the type of the device, the device vendor and the entire information important for the decision process). The final decision is taken by ‘RulesSet’ and further processed by ‘Component-Expert Subsystem’.

3.4 Simulating HSM Access Costs

We estimate the access cost by simulating the HSM system. We treat the HSM system as a Gray Box, which means that we have some knowledge about how it works internally. This knowledge is gathered by observing the HSM system behaviour as well as from the available documentation. Based on this knowledge a simulation model of the HSM system has been developed. The model is based on the fact that the process of serving a request by the HSM system goes through different stages (waiting, unmounting, mounting, positioning, transferring). The transition between the stages is specified by the state transition diagram which is shown in Figure 5 and has been described in detail in [13].

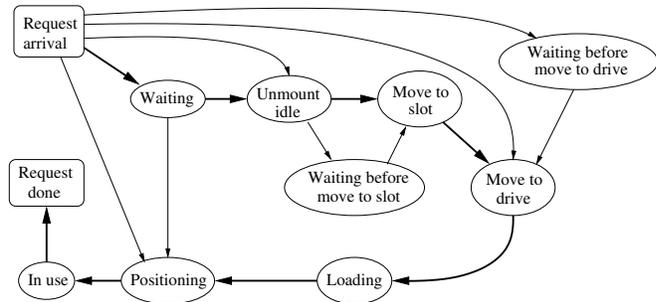


Figure 5. State transition diagram for HSM systems.

The estimation system consists of Monitor and Simulator modules. The Monitor module collects essential information from the HSM (about its current state, the queue state, and about the file which the access cost is estimated). The Simulator module simulates future state changes using this information and the mentioned above model of the HSM system in order to estimate access cost for the file.

The simulation is event driven. At the beginning of simulation, depending on the estimation request and the current state of the HSM system, one or more future events, associated with the fact that a certain stage has just been completed, are scheduled and placed into an event queue. The number of events is equal to the number of real requests being processed by the system plus one for the estimated request. The simulation algorithm then picks the next event from the event queue and schedules zero or one future events according to the mentioned state transition diagram. No event is scheduled for a request that has passed the last stage (transferring). When the last stage for the estimated request is over, the simulation is stopped and the simulated cost is returned.

For each stage, except the waiting stage, the time to complete is estimated by using the appropriate model for that

stage. The waiting stage completion time is obtained automatically during the simulation.

The accuracy of the estimation depends on the accuracy of estimation of the time-to-complete for each stage except the waiting stage. The main source of inaccuracy is the time-to-complete estimation for the mounting, positioning and transferring stage.

The mounting time depends on the contamination of tapes and drives. It has been observed that sometimes the process of becoming on-line for a tape drive takes longer than specified. A few years ago, the same drives had quite predictable mounting time. The estimation model assumes that the mounting time is constant for the given library-drive pair.

The positioning time for DLT tapes is estimated by using the low cost seek model proposed by Sandst  and Midstraum [15]. Again, the contamination can influence the accuracy, by causing more “recover from error” condition to occur increasing the time. The fluctuation of data compression rate along the track also decreases the accuracy. Another source is the unpredictable occurrence of bad blocks along the tape.

The transfer time from the removal media to the disk cache depends mainly on the file size and the transfer rate of the tape drive (we assume the disk cache has a higher rate). The transfer rate itself depends on the compression rate of the stored data. Data with higher compression rate have higher transfer rate. The ratio of the transfer rate for data which is well compressible to the transfer rate for data which is not compressible is usually about 2. The load of CPU and I/O (because of many requests issued to the system at the same time), in turn, can cause that the effective transfer rate to decrease. However, this is rarely the case for well configured HSM systems. The current estimation model assumes a constant transfer rate for a given type of drive.

The access cost estimation function is implemented for the Legato DiskXtender HSM system. An effort to adopt it for CASTOR [3] is undergoing. All these estimators are implemented as independent components compatible with the Component-Expert Architecture, thus they are registered in the CComponentSet (see Figure 3), which fulfils the Components Container (see Figure 4) role. The decision of their usage is taken by CEXS together with RuleSet, which are shown in Figure 4.

4 Implementation Details and Discussion

One of the main achievements of this work is the interoperability of services and components provided by the EU DataGrid as well as the EU CrossGrid projects. In the cost estimation architecture, the main interaction is between the Replica Optimization Service and the Cost Estimator,

where the former acts as a client of the later.

In order to allow for an easy interoperability of the services, both services (Replica Optimization Service and Cost Estimation Service) use web service technologies and communication via SOAP. The main interface is through the method “getSECost” where the Replica Optimizer requests the access cost of certain files located on a Storage Element. In more detail, the Replica Manager (the top level entry point for an end-user as pointed out in Figure 1) provides a client interface “listBestFile” and interacts with the Replica Optimization service for obtaining the transfer costs of a given file.

The Replica Optimization Service is implemented as a Java web service based on a Tomcat servlet container and AXIS SOAP for client-server communication. Thus, the Replica Manager client communicates through AXIS SOAP with the Replica Optimization Service. All these elements are a part of the replication framework. Next, the Replica Optimization Service calls the Cost Estimation Service that is implemented in C++ using gSOAP for client-server communication.

For simplicity, we created a command line interface for the Replica Optimization Service where we can directly invoke the getSECost method. An example is given below where we ask for access costs of a 1GB file on a given Storage Element.

```
./edg-replica-optimization getSECost \  
-s http://zeus07.cyf-kr.edu.pl:18001 \  
-f pfn://zeus07.cyf-kr.edu.pl/f1GB.dat \  
-h lxshare0343.cern.ch
```

```
Result:  
SE cost: 1367.0 [sec]
```

The results are given in seconds and thus one can see the estimated access time returned from the Cost Estimator. Internally, the Replica Optimization Service then adds the network costs for the given files and can thus determine the entire access latency for a given file (according to Equation 2).

5 Experimental Results

In the following section we provide experimental results of the replica selection process that is based on the interaction of the Replica Optimization Service and the SE Cost Estimation service. In more detail, a Replica Optimization Service is deployed at CERN (Switzerland) and interacts with three different storage systems located in CYFRONET-AGH (Poland).

The cost estimation experiments have been performed on three Storage Elements (SEs) - SE1, SE2, SE3 - keeping a set of files, f10KB.dat, f100KB.dat, f1MB.dat, f10MB.dat,

f100MB.dat, f1GB.dat, with different sizes. These file-names also correspond to logical file names (LFN). Furthermore, for each of the LFNs, identical replicas exist at different storage systems. The main goal is to identify the optimal storage location (“best file”) for a given LFN.

Each SE is configured differently (cf. Table 1). Storage SE1 has only disk drives, storage SE2 has disk drives and connections via FTP link to the HSM system controlled by an external machine. The last storage SE3 is connected to another HSM system via a NFS link. The types of the links are important in the estimation process due to differences in transfer performance (e.g. NFS is significantly slower than FTP). The location of the replicated files (fxxxxx.dat) in these three machines is presented in Table 1: files are partially replicated to the three existing SEs.

	SE1	SE2	SE3
f10KB.dat	disk	n/a	HSM cache
f100KB.dat	disk	n/a	HSM cache
f1MB.dat	n/a	HSM tape	n/a
f10MB.dat	n/a	disk	n/a
f100MB.dat	disk	HSM cache	HSM tape
f1GB.dat	n/a	HSM tape	n/a

Table 1. File location in respective Storage Elements

The experiment was aimed to verify the correctness of the integrated DataGrid Replica Manager (Reptor) with the CrossGrid unified access cost estimation facilities. In more detail, the cost function in Equation 1 is applied where only the storage cost is minimized, i.e. the network cost is neglected in the experiments here. The Replica Manager can then answer the question “listBestFile(LFN, relative location)” where it returns the location of the replica with the minimal access cost relative to a given location.

Given the replicas in Table 1, the obtained replica selection decisions of Reptor are shown in Table 2. Thus, for each of the 6 LFNs, the “best” files are listed. For example, for the LFN f100MB.dat the best location is SE2.

One of the most important steps in the Replica Manager decision process is to evaluate the real data access cost on the SE. The obtained access cost estimations of our experiments and comparison with the reality are shown in Table 3 (performed previously tests for the Legato DiskXtender HSM system, which are discussed in detail in [13], show that the exact estimation of the HSM system access cost is difficult and sometimes errors can exceed 20%; the reason of this is shortly discussed in 3.4 section). Since all Storage Elements are located in the same local area network, the network cost for all of these storage elements is the same and can be ignored here. Therefore, the decisions taken by

	SE1	SE2	SE3
f10KB.dat	the best	n/a	
f100KB.dat	the best	n/a	
f1MB.dat	n/a	the best	n/a
f10MB.dat	n/a	the best	n/a
f100MB.dat		the best	
f1GB.dat	n/a	the best	n/a

Table 2. Decision taken by the Replica Manager based on the estimated access

the Replica Manager are in accordance with the cost estimations given by the Cost Estimation service. Moreover, comparing the real values shown in Table 3 with the replica selection decisions taken by Reptor (see Table 2), demonstrates that these decisions have been correct and the cost estimations work properly.

	SE1	SE2	SE3
f10KB.dat	1 / 1 (0)	n/a	1 / 1 (0)
f100KB.dat	1 / 1 (0)	n/a	1 / 1 (0)
f1MB.dat	n/a	201 / 277 (27)	n/a
f10MB.dat	n/a	1 / 1 (0)	n/a
f100MB.dat	8 / 8 (0)	10 / 9 (11)	627 / 802 (22)
f1GB.dat	n/a	5478 / 5935 (7)	n/a

Table 3. Estimated/Real access cost in seconds rounded up (Estimated Error in %)

6 Related Work

Replica selection is a rather new field in Grid research but has longer traditions in the Internet community. Thus, most wide area replica research focuses on network-based replica selection. For example, the Earth Science Grid (ESG) [1] uses a network bandwidth for replica selection. However, our past experience with Data Grids has shown that mass storage systems are often the bottleneck rather than network links due to the possible large range of access latencies of tape systems.

The Storage Resource Broker (SRB) [14] takes a similar approach as we do and currently uses storage latency rather than network bandwidth as the main criteria for replica selection. Thus, a disk-stored replica will be accessed first before trying a replica in a database and finally trying a replica in an archival storage system. If more than one replica is in the same latency category, SRB tries the one that is local before trying a remote copy.

In Storage Resource Managers (SRM) [17] (a uniform interface for secondary and tertiary storage systems) the notion of replica selection is currently not introduced. SRM provides an important interface for Grid storage systems and is currently implemented by several projects within the Grid community.

Further related work on optimization and performance modeling of tertiary storage systems can be found in [16] and [11].

7 Conclusions

In this paper we introduced the replica selection process of the EU DataGrid Replica Manager based on storage access estimations from a system developed within the EU CrossGrid. We discussed the architecture of the Replica Manager and the estimation function for a unified Grid storage system which can be a disk subsystem or a Hierarchical Storage Manager with tape robots. We carried out a set of benchmarks for estimating the access times of files located at different levels in the storage hierarchy. The results demonstrate that the estimates provide a good basis for the Replica Manager to make efficient replica selection decisions.

In the future we will extend our storage access estimator to work also with other mass storage systems such as Castor [3].

References

- [1] B. Allcock, I. Foster, V. Nefedov, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies. In *Supercomputing 2001*, Denver, Texas, November 2001.
- [2] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Design of a Replica Optimisation Framework. Technical Report DataGrid-02-TED-021215, CERN, Geneva, Switzerland, December 2002. EU DataGrid Project.
- [3] The CASTOR Project. <http://www.cern.ch/castor/>.
- [4] The CrossGrid Project. <http://www.crossgrid.org>.
- [5] CrossGrid - Development of Grid Environment for Interactive Applications, 2001. EU Project, IST-2001-32243, Technical Annex.
- [6] L. Dutka and J. Kitowski. Application of Component-Expert Technology for Selection of Data-Handlers in CrossGrid. In D. Kranzlmüller, P. Kacsuk, J. Dongarra, and J. Volkert, editors, *Proc. 9th European PVM/MPI Users' Group Meeting*, volume 2474 of *Lecture Notes on Computer Science*, pages 25–32. Springer, Sept. 29 - Oct. 2 2002.
- [7] L. Dutka, R. Slota, and J. Kitowski. Component-Expert Architecture as Flexible Environment for Selection of Data-handlers and Data-Access-Estimators in CrossGrid. In M. T. M. Bubak, M. Noga, editor, *Proceedings Cracow Grid Workshop '02*, pages 201–209, Cracow, Poland, December 11-14 2002 2002.
- [8] L. Dutka, R. Slota, D. Nikolow, and J. Kitowski. Optimization of Data Access for Grid Environment. In *1st European Across Grids Conference*, Lecture Notes in Computer Science, Universidad de Santiago de Compostela, Spain, February, 13-14 2003. Springer. (in print).
- [9] The European DataGrid Project. <http://www.edg.org>.
- [10] EDG-WP1. Definition of Architecture, Technical Plan and Evaluation Criteria for Scheduling, Resource Management, Security and Job Description. EU DataGrid Project. Deliverable D1.2, September 2001.
- [11] B. K. Hillyer and A. Silberschatz. On the Modeling and Performance Characteristics of a Serpentine Tape Drive. In *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 170–179, Philadelphia, Pennsylvania, May 1996.
- [12] P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger. Replica Management with Reptor. Technical report, DataGrid-02-TED-030408, Geneva, Switzerland, April 2003.
- [13] D. Nikolow, R. Slota, J. Kitowski, and M. Dziewierz. Access Time Estimation for Tertiary Storage Systems. In B. Monien and R. E. Feldman, editors, *Euro-Par 2002 Parallel Processing, 8th International Euro-Par Conference Paderborn*, volume 2400 of *Lecture Notes in Computer Science*, pages 873–880, Philadelphia, Pennsylvania, August 2002. Springer.
- [14] A. Rajasekar, M. Wan, R. Moore, G. Kremenek, and T. Gupta. Data Grids, Collections and Grid Bricks. In *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST2003)*, San Diego, California, April 7-10 2003.
- [15] O. Sandstå and R. Midstraum. Low-Cost Access Time Model for Serpentine Tape Drives. In *Proc. 16th IEEE Symposium on Mass Storage Systems and the 7th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 116–127, San Diego, California, USA, March 1999.
- [16] X. Shen, W. Liao, and A. Choudhary. Remote I/O Optimization and Evaluation for Tertiary Storage Systems through Storage Resource Broker. In *Proceedings of IASTED Applied Informatics Conference*, Innsbruck, Austria., February 2001.
- [17] Storage Resource Management (SRM) working group. <http://sdm.lbl.gov/srm-wg/>.
- [18] H. Stockinger. *Database Replication in World-Wide Distributed Data Grids*. PhD thesis, Institute of Computer Science and Business Informatics, University of Vienna, Austria, November 2001.
- [19] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. ACM Press and Addison-Wesley, New York, NY, 1998.