ARCO: MOVING DIGITAL LIBRARY STORAGE TO GRID COMPUTING

Han-fei, Nuno Almeida, Miguel Loureno, Paulo Trezentos
ADETTI/UNIDE
Av. Foras Armadas, Edif. ISCTE, 1600-082 Lisboa, Portugal

Jos Borbinha, Joo Neves
Biblioteca Nacional
Direco de Servios de Inovao e Desenvolvimento
Campo Grande, 83 - 1749-081 Lisboa, Portugal

Keywords: Distributed Multimedia Storage, Grid Computing, Digital Library, HPC

Abstract: Storage has been extensively studied during the past few decades (Foster et al., 1997; Jos Guimares, 2001). However, the emerging trends on distributed computing bring new solutions for existent problems. Grid computing proposes a distributed approach for data storing. In this paper, we introduce a Grid-based system (ARCO) developed for multimedia storage of large ammounts of data. The system is being developed for Biblioteca Nacional, the National Library of Portugal. Using Grid informational system and resources management, we propose a transparent system where TeraBytes of data are stored in a beowulf cluster built of commodity components with backup solution and error recover mechanisms.

1 INTRODUCTION

The National Library of Portugal (BN - Biblioteca Nacional) is a patrimonial library. This presents special motivations to pursue some goals at both national and international levels, related with a wide range of new technical area especially concerning with digital publishing; digitization; preservation of cultural artifacts; metadata creation, processing and exchange; services for resource discovery, access and interoperability; etc. Each of these areas brings new expectations and requirements for new skills, which BN has been identifying and developing during recent years in trials and pilot initiatives. As a result, valuable expertise has been built, resulting in the actual initiative for the National Digital Library (BND - Biblioteca Nacional Digital).

The BND is defined to serve two main purposes. The first and most immediate is the development of an operational framework by which the BN will address specific problems posed by the management of large collections of digital and digitized items. As a side effect, we expect to reach a critical mass (in technology, knowledge and contents) after which we will be able to address the other areas in a complementary approach.

The actual moment of the BND is not anymore to be experimental, presenting short-term results, but to be stable and trustworthy. That requires thinking seriously in stable models and long-term sustainable strategies for the developments and convergence with the traditional library. Also, this has to be done in accordance not only with our general mission, but especially taking in account the limited resources that BN will be able to reserve for it.

To provide us the expected necessary flexibility, it was decided that the technical solutions will be based mainly on open and scalable technology, to be built and integrated incrementally. This will make it also possible to better consolidate the results with the traditional library's services.

The main services that the BND is expected to assure are the storage, search, retrieval and preservation of digital resources. In this paper we are addressing the problem of the storage.

2 PROPOSED ARCHITECTURE

The goal of ARCO system is to provide and managing a transparent layer for storing large amounts of data. The high level requirements suggest that all interactions with lower or higher abstractions layers would be throughout interfaces. This interfaces should be public and well defined in the System Definition document.

Figure 1 depicts the ARCO architecture. The left figure represents the interaction of ARCO system

-fei H., Almeida N., Loureno M., Trezentos P., Borbinha J. and Neves J. (2004).
ARCO: MOVING DIGITAL LIBRARY STORAGE TO GRID COMPUTING.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 64-69
DOI: 10.5220/0002651400640069

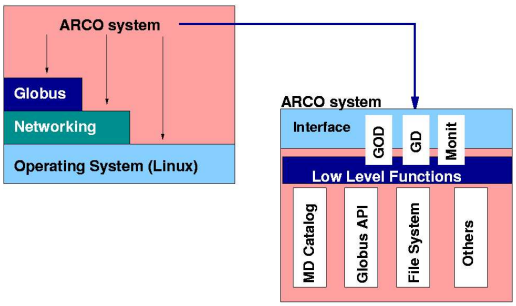

Figure 1: ARCO high level view

with the operating system, the network resources and with Globus middleware. ARCO uses several OS system calls to access the local file system, uses network resources for integration purposes and Globus for Grid Services(Foster, 2001; Open Grid Services Infrastructure (OGSI), 2003). ARCO is build over Globus toolkit 3.0 that supports Webservices (Foster et al., 2002).

The right part of 1 contains the internal components of ARCO system. In the upper level we have the interface layer that is responsible both for providing graphical interfaces for storing data and retrieving it. The graphical interfaces are implemented in HTML/PHP but they could be implemented with other programming language since there is a low level layer with commands for the basic features of the system.

3 SYSTEM COMPONENTS

The ARCO system is divided in three units, GOD (Digital Object Management); GG (Grid Management) and Monitorization. These are the types of operations that users are allowed to do, according to their functions. In each of these units exist subunits according to the type of tasks allowed to the users.

3.1 GOD

This section can be accessed both by the system administrator and other users. It has several operations regarding Digital Objects, such as Store (Stores a new object in the grid), Remove (Removes an existing object from the grid), Copy (Copies an object from the grid to the working directory), Replace (Replaces an existing object with a new one), Update (Updates an object with a newer version) and Recover (Recovers a deleted object). When removing an object, the object is not physically removed. Instead, it is moved to a different location, allowing its recovery. The physical removal is only allowed to the system administrator.

3.2 GG

The system management operations are available in this unit. The system administrator is responsible for performing these tasks, concerning to “Grid Nodes”, “Volumes”, “Users” and “Groups”.

The **Node Management** subunit is where the grid nodes’ configurations are done. The operations available are Add Node (Add a new node to a volume), Remove Node(Remove a node from a volume),Edit Node(Edit the node configuration), List all Objects(List all objects presented in the node) or Locate an object(Locate one object in the system). This kind of operations are very useful because it’s very easy to add or remove one node from the system, and if a node with objects is removed it’s possible to synchronize the volume or use the volume mirror as master.

The **Volume Management** is where the operations related to volumes are done. The role of operations possible are Add Volume(Add a new volume to the system), Remove Volume(Remove a volume from the system), Edit Volume(Edit volume information), List Volumes(List all volumes) and Synchronize Volume(Synchronize volume information with the volume mirror). Here there are several considerations to have in mind. One volume can only have one mirror, but when a master volume is deleted or shutdown the mirror one will pass to master automatically. It’s not possible to remove a volume that has nodes associated to it. Every time that one object is inserted in one volume it is inserted automatically in his mirror, if there is one.

In the **User Management** subunit the user can do several operations related to the system’s users. The user management operations are Add User (Adds a new user to the system), Remove User (Removes a user from the system), Edit User (Edits the user’s data) and List Users (Lists all the users).

In the **Group Management** area the user can do several operations related to the system’s groups. The group management operations are Add Group (Adds a new group to the system, specifying the group’s access permissions on GOD, GG, and Monitorization), Remove Group (Removes a group from the system), Edit Group (Edits the group’s data and access permissions) and List Groups (Lists all the groups). It’s not possible to remove a group that has users associated to it.

3.3 Monitorization

The **Monitorization** is done over a grid node. It is very useful to analyze node performance and behavior along the time and setup the needed alarms.

The main goal is to prevent system failures by warning the administrator and provide information to

the users, that can be used to improve the storage strategy.

It presents information of the actual state of the node (available space, used space, cpu temperature, detected errors, system logs, etc.). It is also possible to setup alarm values and desirable actions like warning the system administrator or do a load balancing.

4 LOW LEVEL OPERATIONS

4.1 Object Description

Figure 2 is the general system structure of the ARCO system. The ARCO system can be abstracted into different level from several viewing angles.

Firstly, the whole concept of digital library can be broken down into volumes, grid nodes, file systems, books. Secondly, actions can be from groups, users, and can be further divided as basic operations. Finally, operations can be taken to objects in different abstraction level of ARCO system.

The ARCO system provides webpage interface by using API function calls, as well as command line interface.

ARCO metadata catalogue is used to provide details of object description in different level, operation description, and system configures information description.

There are two interfaces between the high level webpage toolkit and the low level function call realization. The first is the data interface that uses the common metadata information, the second is the function call control interface. The metadata set includes information about user and user group description, digital library general description, volume storage object description, grid nodes description, book object description, and management operation description.

The user object has fields of user id, user name, password, and group id. The description of group object includes group name and group id. The permission description provides detailed control information about users, groups, and different operations. The digital library virtual structure is described in several levels.

The full storage of the library can be divided into normal part and mirror part. The mirror storage is a real time backup of the normal storage in the meaning of every operation to book object. The mirror mechanism demands every operation to the normal storage in the book object level will also take place meanwhile to the mirror storage. The library storage object, normal or mirror, is composed of one or several volume objects. The volume object description depicts volume attributes, such as volume name, mirroring pointer, and other management information. Every volume is composed by one or several grid host nodes. Each node can only belong to one volume.

The grid node is not only the component of the storage volume from the view point of digital library abstraction, but also it is the basic unit of computing grid environment for staging operations, as well as the basic unit of grid computational information and resource discovering entity.

The MDS(Fitzgerald et al., 1997) in grid node provides resource information, such as every mounted file system. The ARCO system utilizes the MDS computational resource information by querying the LDAP server default port 2135 in every grid node. In ARCO metadata catalogue, the grid node description include host full domain name and ip address.

There are different operations which can be taken to different objects. The smallest basic object is the *digital object*[1]. The digital object description gives out the details of a book, for instance. The book object description includes object id, status, book size, last operation date, book name, file system mounting point where book locates, full path, node ip address, user name who command the operation.

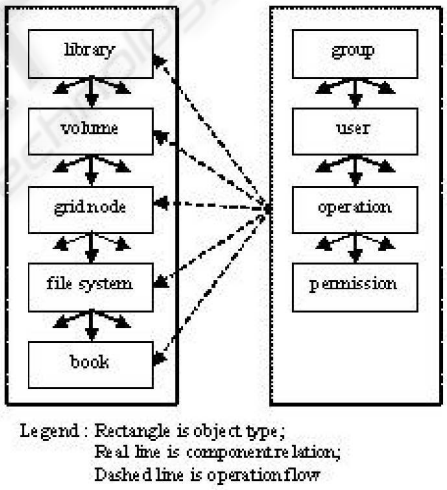

Figure 2: Object component and operations

4.2 Operation Basis

In order to provide stable and different operations for the ARCO system, several data descriptors have been used to define objects. The data structure of bk-list is a structure type. There are mainly three parts of fields in the bk-list structure, the first is information that can be queried and got from book object description table, the second is information that can be queried

[1] A digital object can be a book or a film.

and got from grid node description table, and the third is information that can be queried and got from volume object description table. The function bk-list-init does a joint query to several tables in the database, get as many attributes as the metadata can provide. The function bk-list-print is for printing the bk-list structure.

In the god-stdlib.h header file, we have defined server host names, globus gass server port for url-copy, globus-personal-gatekeeper port and login name, ldap server query port, ldap search base, file system mounting points in grid nodes, etc.

To describe the grid node, we use sv-list structure type. The library storage can be composed of several volumes. There can be several grid nodes in each volume. When a book is copied into the library storage system, firstly according to the volume attribute of the book, there are several nodes can the possible place where book will be copied into. And in each of these nodes, there can be several file systems mounted. In each node, some file systems are only used for operation system, such as swap area, system installation partition. The file system that can be used on the library storage will named in a special way, including a special mode that will be recognized and be taken as a possible candidate for storing the new book. Starting from the volume attribute of the new book, the function sv-list-init firstly query the metadata database to get the nodes information belong to this volume, so the sv-list structure has fields to store the total number of available nodes, and a list of recorder for each of the grid node. Then function sv-list-init will try to query the node object description table to get some detail information about each node, such as node ip address. Next step, the function sv-list-init will send ldap query to mds server of each grid node, parse the query result and get detailed node information, such as total number of mounted file system, file system name, free space, then we will further recognize the file systems that can be used for book storage, fill all the detailed file system information into each node field of sv-list structure.

4.3 Functions and Operations

In ARCO system, there are many operations, which can be taken to an object that can be in a different abstraction level. In general, the library as an object, different users and user groups have different permission to take actions on it. The library is composed by volumes, some of which are mirroring backup of others. In each volume there are several nodes. Every node has several file systems can be used.

The function god-update is a low level operation, which copy a new book to a proper file system or update an old book that is already in the system. In this function, volume attribute can be got from bk-list, then by using volume attribute, sv-list can be initialized, and we can get the proper file system for storing the book. The book will be tarred and transferred to remote node, and then untarred. The procedure of taring, transferring and untarring are staged to the grid system as a job, so the command can be send from a server, and actual operation can take place between other two remote nodes. If the volume of the new book has a mirror volume, then the same procedure will also take place to the mirror volume for real time backup. After the work has done, book information will insert into the table of book object description. For the time being, the scheduling policy of god-update is ”largest capacity first”, which means we always choose the server that has largest storage capacity. We can also change the scheduling policy by using different calling parameters.

The function god-cp is also a low level operation, which just simply copies a book between two remote nodes. The realization of this function is a much simple version of function god-update. The function god-cp is used for copying a book between different working areas.

The function god-rm is for remove a book in a volume. In this function, we first find which node the book locates in, then move the book to a temp delete directory, and update the book status as deleted in book description table. All of the procedures are submitted as remote jobs to grid system and take place in remote nodes.

The function god-rc recovers a removed book in a remote node. This function does reverse operation as the function god-rm.

To every action, when the operation has finished, the operation execution information (book name, size, time costing in different procession stage, etc.) will be recorded into the job information statistic database. This is done by the statistic module, which collect time benchmarks in different places of procession and update the statistic database.

The module func-job-submit and url-copy provide the programming interface to the globus[5-3][5-4] fundamental service. The url-copy provides two different interfaces for remote file transfer and copy, respectively for globus version 2.0 and globus version 2.2. The func-job-submit receives globus-personal-gatekeeper port and login name, and job description rsl language sentence as the parameters. In our prototype, all the operations that take place on remote server are customized and submitted from local server. The rsl description sentence defines job details, such as the name of executable, parameters, execution environment demand, stdout and stderr, etc.

One of the project objectives is to provide the system with scalability, so at any time, there are the possibility of adding in some new servers into the system or shutting down and removing some servers from the

system. The function god-shutdown is a function for physically take a node out off the system. When a node is to be shutdown and taken out off the system, all the books located in this node need to be reloaded to other nodes, to guarantee the data integration. There are two cases, one is the volume has a mirror, another is the volume don't have any mirror. In the first situation, books in this node can be copied from the mirror volume, so the node can be shutdown at once. In the second case we need firstly to redistribute all books in this node to other nodes in the same volume. The realization of the god-shutdown utilizes the god-update and god-cp basic operations.

The function god-synch is for check the data integration between a volume and its mirror. It tries to find any difference between the two volumes and take proper operation to recover the storage.

5 EXPERIMENT AND RESULTS

In this phase of the prototype the work has not evolved into network and processing optimization. Nevertheless, it is interesting to present some benchmarks with process phases. This benchmarks were also used to verify Grid applicability using Trezentos-Oliveira model (Trezentos and Oliveira, 2003).

Table 5-1 depicts the performance statistic result with different book size.

In our experiment, the largest book is about 660 MB. The table presents in the first column is the book size in Kbytes; the second column is the time used for query free space, this time cost is a constant value. The free space query has done by send ldap query to ldap server port in each server. For the time being, our prototype has any one node act as book input point, it means books will be copied from this local server toward distributed multi-servers. The reason about why we provide dynamic free space capacity query is for the ability of our prototype expanding into multi-servers input, and copy toward multi-servers. From the trend in figure 3, we can clearly know that the time cost of tar process is perfect linear, correspondent to different book size. The time cost item of url copy and all, are nearly linear relation relative to the book size. In ideal environment, the url copy time should be linear relation with the book size. In our result, the url copy time costing coarse is introduced by the network delay, operation system, globus system work load and scheduling policy, etc. The feature of time cost of untar is something like a step function, within a sphere of book size, it keeps stable, out of the sphere, it steps down or up a class.

Table 5-1 Book Size and Process Benchmark

Bk Size(kb)	FS Query	Tar	Url CP	All
132	4	0	0	4
132	2	0	0	2
6156	3	1	6	10
8652	3	2	15	20
79360	3	16	96	115
101272	2	20	102	124
126676	2	25	112	139
131564	4	28	138	170
192000	2	41	296	339
295952	2	62	244	308
447652	2	92	736	830
660372	2	137	830	969

Note: unit of book size is in Kbytes, unit of benchmark is in second

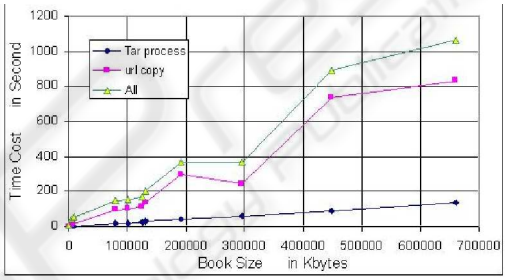

Figure 3: Process benchmarks

6 RELATED WORK

Indiana University's Distributed Storage Services Group (DSSG)(University, 2000) has provided a scalable, network accessible, standards-based storage infrastructure. This is accomplished via the Distributed Computing Environment Distributed File System (DCE DFS) to deliver a ubiquitous common file system to IU researchers and with the Massive Data Storage Service (MDSS, based on the High Performance Storage System or HPSS) to store vast amounts of archival or near line data on a hierarchy of storage media. These are augmented by the Andrew File System (AFS) for remote collaboration and data sharing. In September 2000, working with IBM, Indiana University instituted into production the world's first geographically distributed High Performance Storage System (HPSS). With hardware and software components distributed across two Indiana University campuses (located at Bloomington and Indianapolis, some 50 miles apart), users are able to read or write data locally. In DCE DFS system (Schroeder et al., 1999) , IBM 3590E Magstar tape drive and IBM H70 HPSS Tape Mover are used

to provide very large amount of storage capacity in types (total capacity of the tape library is nearly 1500 tapes).

However, with the rapidly progress in hardware, now the 100 Giga-bytes hard-disk become relatively cheaper, and the retrieving time of data resided on hard disk is quicker than tapes. The distributed server clusters composed of cheap brand-less ordinary performance computer with large capacity of hard-disk storage ability become possible for our demand. The remaining core technology point is how to automatically distribute data to geographically dispersed servers. The globus fundamental toolkit gives us viable solutions. Combined with globus job administrating and staging interface, we have developed our distributed storage prototype for digital library data storage system.

7 CONCLUSION

The first phase of ARCO system implementation proved that the architecture is efficient for large amounts of data. The management framework also is becoming very powerful for maintenance operations. Globus/Grid and commodity components as key components assured a very good price / performance compromise.

Nevertheless, the ARCO subsystem need to be further developed and enhanced. Future work will be developed in the optimization of ARCO transfer functions and storage process.

The development of the BND presents a unique class of challenges. The ambitious vision is bounded by the limited resources that BN can reserve for this purpose, which requires technical solutions with great flexibility, to be developed in an incremental process. In this context it was decided to give special attention to open and scalable technology, for the services and also for the storage.

The price of the storage technology has been dropping in an amazing rate. However, the prices of complete commercial storage solutions for libraries have been not following this trend in the same speed. Top market companies still price their products too high, especially when we compare them with the prices of the original hardware components and the expected software development.

Also, commercial companies have been too slow in adopting innovative emerging technology for digital libraries. After realizing this, it was decided to build a new specific framework for BND, based on open technology, new models for partnerships and alliances, and also internal developments.

At the services and contents level, the solution described in this paper will be complemented with other open source technology and community based solutions, such as the FEDORA(FEDORA, 2003) framework for middleware services and the METS format (METS, 2003) for the structural metadata.

REFERENCES

FEDORA (2003). Flexible extensible digital object and repository architecture. http://www.fedora.info/.

Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., and Tuecke, S. (1997). A Directory Service for Configuring High-Performance Distributed Computations. In *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, pages 365–375.

Foster, I. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–??

Foster, I., Geisler, J., Nickless, W., Smith, W., and Tuecke, S. (1997). Software infrastructure for the i-way high performance distributed computing experiment. *Proc. 5th IEEE Symposium on High Performance Distributed ComputingSoftware Infrastructure for the I-WAY High Performance Distributed Computing Experiment*.

Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration.

Jos Guimares, P. T. (2001). Spino: A distributed architecture for massive text storage. *ICEIS*, 1:244–248.

METS (2003). Mets. metadata encoding and transmission standard. http://www.loc.gov/standards/mets/.

Open Grid Services Infrastructure (OGSI), v. . (2003). Gwd-r (draft-ggf-ogsi- gridservice-23).

Schroeder, W., Marciano, R., Lopez, J., Gleicher, M., Kremenek, G., Baru, C. K., and Moore, R. (1999). Analysis of HPSS performance based on per-file transfer logs. In *IEEE Symposium on Mass Storage Systems*, pages 103–115.

Trezentos, P. and Oliveira, A. (2003). Metrics for grid applicability: a distributed elliptic curve platform assessment. *Proceedings of the Fifth International Conference On Parallel Processing And Applied Mathematics, to be included in Lecture Notes in Computer Science, Springer-Verlag*.

University, I. (2000). Distributed storage services group (dssg), http://www.indiana.edu/ dssg/.