

Distributed Storage Management Service in UNICORE

Tomasz Rękawek^{1,2}, Piotr Bała^{1,2}, Krzysztof Benedyczak^{1,2}

¹ Interdisciplinary Center for Mathematical and Computational Modelling,
University of Warsaw, Warsaw, Poland

² Nicolaus Copernicus University
Toruń, Poland

E-mail: {newton, bala, golbi}@mat.umk.pl

In this paper we are presenting a UNICORE Distributed Storage Management Service (DSMS). The system provides a native UNICORE solution for exposing storage resources distributed between different UNICORE sites as a single logical space. Such a feature is highly desired by end-users, allowing them to forget about an actual location of their files. Additionally the system allows for a better Workflow System experience as a larger storage space does not limit the size of files used in users' workflow jobs. The paper also compares the DSMS system to the other existing solutions, both available in UNICORE and other grid middlewares.

1 Introduction

The experience with grid deployment shows that storage resources are similarly important as compute resources. UNICORE middleware provides possibility to uniformly expose distributed compute resources but in case of distributed storage either separate solutions must be deployed and then integrated or the system is limited to use separate storage resources independently.

There are two possible approaches which can mitigate this situation. The first one is to implement a new solution from scratch, integrating it as good as possible with the existing UNICORE stack. Another option is to provide a UNICORE interface to an another existing solution from a different grid middleware. The later approach brings all the problems of different security mechanisms and can be painful for administrators as they have to install and maintain an additional (usually very complex) software. Furthermore it is highly possible that even despite of a significant effort the result might have some other limitations resulting from different middleware paradigms. As we believe that cost of the second approach is higher than implementation of a dedicated UNICORE solution we have decided to implement the first approach.

The next two sections of this paper present existing storage capabilities in UNICORE and other grid middlewares. The section 4 precisely defines the project aims. Next the DSMS system architecture is described along with explanation of its idea and realization. The subsequent section 6 discuss the synchronization issues and respective solutions implemented to overcome them. Finally the comparison with other systems is given with a description of the current and future work.

2 Existing storage capabilities in UNICORE

UNICORE Storage Management Service (SMS) provides an abstract filesystem-like view on a storage resource. It has common POSIX-like operations (eg. `mkdir`, `delete`,

listDirectory) and a few more to initiate file transfers.¹ The standard UNICORE distribution includes the following SMS implementations:

- **FixedStorage** — stores files in a directory defined in a configuration,
- **HomeStorage** — stores files in a user's home directory,
- **PathedStorage** — a destination path is resolved at runtime so environment variables can be used to parametrize it (e.g. `/opt/uni-storages/$JOB`).

Unfortunately none of these implementations allows client to distribute files throughout the grid. There are, however, three solutions that extend standard storage capabilities: UniRODS, UniHadoop and Chemomomentum Data Management System.

UniRODS²³ is a Storage Management Service implementation providing access to iRODS⁴ distributed data storage system. iRODS provides a shared disk space consisting of multiple physical storages. Metadata and storage information are persisted in iCAT service based on PostgreSQL, MySQL or Oracle database engines. Advantage of the UniRODS is the usage of the standard UNICORE Storage Management Service interface, so the client software can work with it without any modifications. UniRODS is also very easy in installation, kept simple and integrates well into the internal UNICORE/X architecture.

On the other hand, because authorization mechanisms in UNICORE and iRODS are completely different, UniRODS has to use static files to map users. Another disadvantage is necessity to install and maintain yet another, complicated service on grid nodes.

UniHadoop⁵ is a similar solution to UniRODS, but uses a different backend — Apache Hadoop⁶. Hadoop is an open-source implementation of Google File System⁷ (GFS). GFS is optimized for using a large number of cheap disk servers, therefore replication and monitoring are priorities for the system. Files stored in GFS are split into fixed size chunks which are distributed into *chunk servers*. Each chunk has 64 MB and is replicated on three servers. UniHadoop, like UniRODS, exposes Storage Management Service interface and additionally is available in the standard UNICORE distribution. Additional advantage is that Hadoop uses system user name for authorization, so it's possible to apply standard Storage Management security model in which grid user (identified by his X.509 distinguished name) is mapped to the system user (identified by her login). This mapping is done by attribute source service (e.g. XUADB or UVOS) existing in a grid. Unfortunately, with UniHadoop there is again a requisite for an additional complex system to maintain — Apache Hadoop. Moreover Apache Hadoop internal security is not yet matured and misses features which are crucial for the grid. There is no support yet for transport level encryption and proposed authentication solutions are going to be based on Kerberos which is not the best option for X.509 certificates based UNICORE.

The Chemomomentum Data Management System⁸ was designed as a native UNICORE distributed file storage with support for advanced meta data, ontologies and support for retrieving data for external storages like scientific databases available on-line. The system is comprised of a single entry point service called *Data Management System Access Service* and several back-end services providing actual functionality. The physical storage of files is performed using the standard UNICORE SMS service.

The Chemomomentum DMS is a powerful system but bears a significant amount of problems for its potential users:

- The single service being an entry point to the whole system is a bottleneck.

- The whole system is complex and therefore hard to maintain. It seems that support for the system was dropped and it is not updated to the newest UNICORE releases. The effort to perform such an update can be assessed as quite high as there are many services, some of them require a domain-specific knowledge (like external database access tools).
- The system requires extensions on a client side. This is especially hard for UNICORE users as they have to install the extension and for developers who have to maintain plugins for each UNICORE client.

3 Data management solutions in other grid middlewares

Beside the UNICORE world, data management solutions are usually called Storage Resource Managers (SRM) what is at the same time a name of the popular SRM protocol⁹ used to access a storage. In this section the four most popular managers are shortly introduced.

StoRM² is a simple implementation SRM. It exposes a selected POSIX file system. If the distribution is needed, then distributed file system (like IBM GPFS) has to be used. Authorization is done by the POSIX Access Control List and external mapping (from grid users to the system login) is necessary. StoRM uses an internal database for storing metadata and file locations. StoRM has also plug-in mechanism which can be used to utilize additional capabilities of the file system¹⁰.

Disk Pool Manager (DPM) *is a lightweight solution for disk storage management*¹¹. Despite its advertised „lightweightness“, DPM is a more complex system than StoRM. Usual installation is distributed into two type of servers: *head node* and *disk node*. Head node manages metadata and is an entry point for clients. Disk nodes store physical data. The following services run on the head node:

- **DPM nameserver daemon** — manages metadata and directory structure,
- **SRM daemon** — a client entry point with the SRM interface implementation,
- **DPM** — a main daemon which executes requests.

These services do not have to be on the same server, but can not be replicated.

dCache¹² is a much more sophisticated solution than StoRM and DPM. Additional to disk storage, it supports tertiary memory (usually magnetic tapes). That kind of memory is very slow but cheap. dCache has mechanisms to copy (cache) recently used files from tapes to the disks. It also can recognize very often opened files (so called *hot spots*) and replicate them throughout the disk servers in order to balance the load. There are three types of servers in dCache:

- **head node** — unique node that runs the database, name server and other not replicable services,
- **door nodes** — entry points for clients (can be replicated),
- **pool nodes** which stores the data.

dCache's name server is called Chimera. It uses PostgreSQL database for storing files metadata and directory structure.

CERN Advanced STORage manager¹³ (CASTOR) is similar to dCache. It also supports tertiary storage and uses disks as a cache memory. CASTOR consists of many stateless daemons querying the central database and such architecture increases scalability. The most important components are:

- **request handler** is a lightweight service adding new requests to the database,
- **request scheduler** schedules requests for execution, it is possible to use an external scheduler such as LSF or Maui Cluster Scheduler,
- **Stager** is executing clients' requests,
- **resource monitoring daemon** gathers information for the scheduler,
- **name server** stores directory structure,
- **Distributed Logging Facility** is responsible for logging.

The SRM resources are able to provide distributed storage, however distribution is practically limited to the hosts at one site. The Logical File Catalogue (LFC) service¹⁴, available in gLite middleware, provides a capability to maintain the global file space. The LFC uses an Oracle or MySQL database and exposes a proprietary interface to control its contents. The LFC also implements an internal authorization system. It is integrated with other gLite tools and serves as one of the principle components of the Worldwide LHC Computing Grid.

4 The aim of the project

Purpose of the system presented here is to combine multiple UNICORE storage resources into a single logical resource. This logical space should be usable by both grid client software directly and grid services (for instance by the UNICORE Workflow Service). Users should not have to know where their files are stored physically. Optionally advanced users should have an opportunity to be aware of the distributed nature of the storage and physical files locations. As it can be learnt from the Chemomomentum DMS lesson, the system should not be too complex — otherwise its future maintenance would be problematic.

The requirement which can be directly derived from the above paragraph is that the standard and existing UNICORE client software should be able to access and operate on the newly created solution. Therefore the interface to the distributed file space must be the UNICORE Storage Management Service. The service should not limit the standard capabilities of the Storage Management Service as for example supported file transfer protocols.

To achieve high performance the system should be ready for providing an arbitrary number of entry points. Additionally file transfers must be performed directly between source and destination storages, without employing any proxy.

Last but not least the system must use the standard UNICORE authorization and authentication mechanisms to enable smooth and natural integration with the whole middleware.

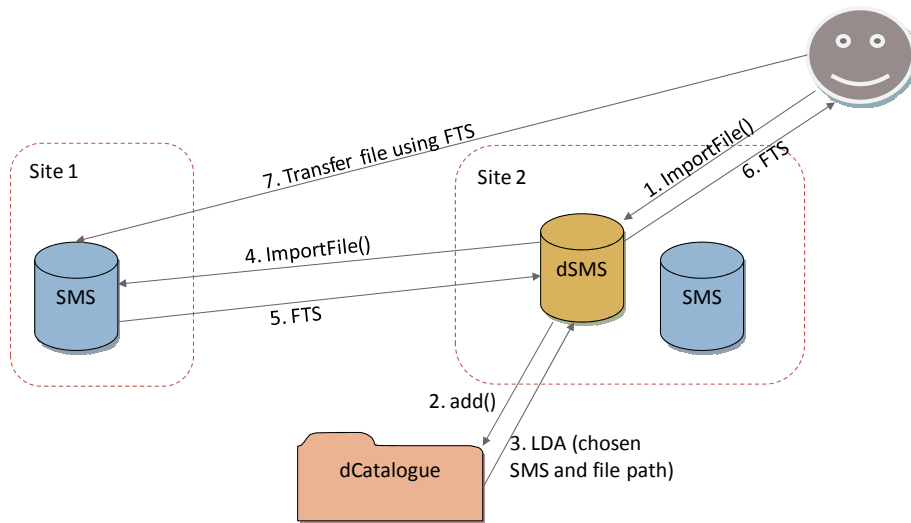


Figure 1. `ImportFile` operation invoked on DSMS

5 DSMS approach

DSMS consists of following components:

- data is stored on existing, standard Storage Management Services,
- file locations and directory structure is stored on a service called **dCatalogue**,
- client entry point is a special Storage Management Service implementation called **DSMS**.

The presented system can use an unlimited number of storages, which are available from a UNICORE Registry. Also an entry points number is unlimited. `dCatalogue` is a central service. It is storing only a minimal amount of information about the files to minimize synchronization and performance issues. `DSMS` despite of exposing Storage Management interface also connects to the `dCatalogue` and physical storages.

The overall idea is to mask distributed nature of the system with the standard Storage Management implementation. We will show the way we have achieved this using two representative examples of operations. The figure 1 presents the `ImportFile` operation invoked on `DSMS`.

`ImportFile` operation allows user to add a new file to the storage. At the beginning a user connects to the `DSMS` using a standard UNICORE client. The client invokes function `ImportFile` (step 1) and waits for an endpoint reference of File Transfer Service (FTS). `DSMS` connects to `dCatalogue` (2) asking for a physical SMS where the file should be saved. `dCatalogue` chooses one SMS out of those existing in the grid (using a special algorithm) and returns its EPR^a and physical file name (3). This pair - SMS EPR and

^aEPR stands for an *Endpoint Reference Address*, which precisely defines a network address of a UNICORE service and a concrete resource accessible through this service.

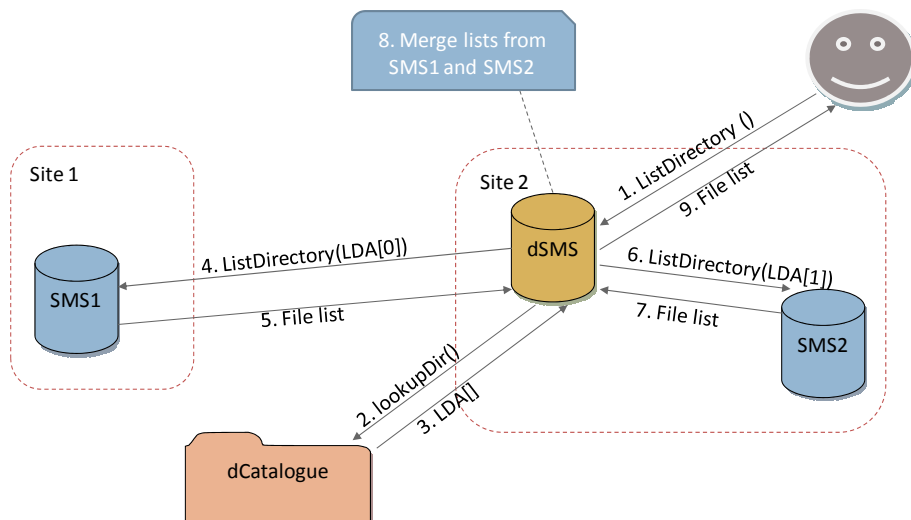


Figure 2. ListDirectory operation invoked on DSMS

file path is called Location Dependent Address. DSMS connects to the received SMS (4) calling an `ImportFile` operation. SMS returns FTS EPR (5) which is returned to the client (6). Further communication is performed as usual: the client sends the file using the received EPR (7). Operations 2 and 4 are done on behalf of the client using trust delegation mechanism and hence the client is an owner of the file on the SMS. The user connecting directly to the SMS has access only to his or her own files. dCatalogue also authorize all operations using the client's X.509 distinguished name.

`ListDirectory` operation presented on the figure 2 works slightly different. Although directory structure is stored in dCatalogue, file metadata (such as size, privileges, etc.) has to be retrieved from appropriate storages. So DSMS (in step 2) invokes the `lookupDir` operation and gets the list of storages which contains files from the given directory (3). After that DSMS connects to each storage and invokes the `LookupDirectory` operation (4–7). When all lists are collected they are merged (8) and sent to the user (9).

Storage Factory is a service introduced with UNICORE 6.3. It can create separate SMS instances for each workflow job (other use cases are also possible). Advantage of the Storage Factory is that it can remove all files created by a workflow job by removing the whole SMS instance. Factory can be configured to create new DSMS within a separate namespace. UNICORE Storage Factories can be also used by the system by dynamically creating instances of DSMS service. A DSMS which was created using the Storage Factory can subsequently create new physical SMSes and store files in them.

Of course usage of the Storage Factory is not limited to the UNICORE Workflow System: it can be used for any other application which can take advantage of a dynamic creation of a SMS instance.

6 Handling errors and synchronization issues

Synchronization between the dCatalogue database and a real file state in the SMS resource is a very important issue. There are two main possible causes of losing this synchronization:

- A) file, which does not exist in dCatalogue, appears in SMS resource,
- B) file which belongs to DSMS is removed from SMS, but not from dCatalogue.

DSMS operations which could suffer from the above problems can additionally be divided into three main groups:

1. directory listing operations (methods `ListDirectory` and `Find`),
2. operations which add new file (methods `ImportFile`, `ReceiveFile`, `Copy`, `Rename`),
3. operations that get file or file information (methods `ExportFile`, `SendFile`, `ListProperties`, `ChangePermissions`, `Copy`, `Rename`).

During `ListDirectory` and `Find` methods DSMS gets file lists from each used SMS, merges them and sends back to the client, so there is no synchronization problem possible here: even if data in dCatalogue is invalid the client will not notice it.

If we consider operations which add a new file, synchronization problem A) may appear if a file with a given name exists on an SMS (but not in dCatalogue) and user tries to send that file again in order to replace it. To avoid A) problem DSMS checks each used SMS looking for the file with the given name. If the file exists, it will be added to the dCatalogue and replaced by the new file. Synchronization problem B) with this kind of operation can not appear.

The last group of operations are these which read file information. If user want to fetch file which does not exist in the dCatalogue, then before throwing the `FileNotFoundException` exception, that file will be searched in each SMS. If it is found then it will be added to the dCatalogue and returned to the user (and the A) problem is resolved). If the file exists in the dCatalogue then DSMS also checks if the file exists in the proper SMS. If not, file will be removed from the dCatalogue and user gets the mentioned exception.

It must be noted that in standard circumstances none of the synchronization issues should appear. To cause A type of problem, a user must use the physical SMS directly (what can be made difficult by administrators, by hiding the SMS from the Registry) or copy the files with other tools like FTP or SCP. Additionally the files must be placed in a special, hidden folder created by the DSMS. The situation with the B type of issue is analogous. Because of this fact and the performance penalty introduced by the above described auto-synchronization methods, they can be disabled in the DSMS configuration.

7 Summary

The DSMS can be seen as a direct successor of the Chemomentum DMS: it is a native UNICORE solution. In comparison with its predecastor the DSMS is more lightweight as

it does not provide some of the more advanced features (metadata, support for ontologies and external databases). Additionally DSMS removes the original bottleneck of the system and necessity for usage of a client's extension. Therefore we hope that DSMS will be loads easier to maintain.

DSMS compared to UniHadoop and UniRODS does not require installation of an additional storage system and also does not introduce security mismatches between UNICORE and other system. The DSMS is also quite different than typical SRM solutions: it is designed to support storage which is distributed between different sites, while solutions like DPM are designed to support disks exposed by different machines located in a single administration domain. DSMS exploits arbitrary UNICORE SMS implementation. It may be even used to merge Hadoop SMS and iRODS SMS as a single logical space). Therefore DSMS can be seen more as a LFC counterpart. However LFC is designed differently as it stores more data about files centrally, what brings more complicated synchronization issues. Also the LFC requires usage of special operations to manage files in its context.

At the current project stage we can state that all basic requirements for the distributed storage were fulfilled. The system was recently integrated into the standard UNICORE distribution as an experimental type of storage. It can be seen as a perfect solution for a grid system comprising of several sites, where each of them is providing their own storage resource.

Our current work is concentrated on measuring the performance of the system. We are paying a special attention to an influence of the automatic dCatalogue synchronization feature on an overall performance. Of course system scalability is another crucial aspect that must be carefully determined. In future more advanced strategies for choosing the optimal physical storage resource are planned. Currently a round-robin strategy is a default one and another one which is available chooses a storage resource with the most free space.

References

1. Schuller, B.: General Introduction to UNICORE, URL: http://www.unicore.eu/documentation/presentations/unicore6/files/General_Introduction_to_UNICORE.pdf (10.05.2011)
2. Derc, K.: *Integracja systemu iRODS ze środowiskiem gridowym UNICORE 6*, Master's thesis, NCU Toruń 2008.
3. Janiszewski, K.: *Rozproszone bazy danych w systemach gridowych*, Master's thesis, NCU Toruń 2010.
4. The iRODS project, <https://www.irods.org> (16.05.2011)
5. Bari, W., Memon, A. S., Schuller, B.: Enhancing UNICORE Storage Management using Hadoop Distributed File System. *Euro-Par 2009 Parallel Processing Workshops*, LNCS, vol. 6043, Springer, Heidelberg 2010.
6. The Apache Hadoop project, <http://hadoop.apache.org> (16.05.2011)
7. Ghemawat S., Gobioff H., Leung S.-T.: The Google file system. *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003*. ACM 2003.
8. Ostropytskyy, V.: Data Management in Chemomomentum, European Grid Technology Days, Brussels, 2007,

ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/ssai/20070926-27-cm-d2-par4-data-management-chemomentum_en.pdf
(22.03.2011)

9. Sim, A., Shoshani, A. et al. (eds.): The Storage Resource Manager Interface Specification Version 2.2. OGF 2008.
10. Corso E., Cozzini S., Forti A., Ghiselli A., Magnoni L., Messina A., Nobile A., Terpin A., Vagnoni V., Zappi R.: StoRM: A SRM solution on disk based storage system. *Proceedings of the Cracow Grid Workshop 2006*, Kraków 2006.
11. Grid Data Management — DPM, URL: <https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm> (19.02.2011)
12. The dCache Book 1.9.5, URL: <http://www.dcache.org/manuals/Book-1.9.5/Book-a4.pdf> (25.02.2011)
13. The CASTOR project, <http://castor.web.cern.ch/castor> (16.05.2011)
14. The LFC webpage, <https://svnweb.cern.ch/trac/lcgdm/wiki/Lfc> (16.05.2011)