An effective XML based name mapping mechanism within StoRM

Ezio Corso, Alberto Forti, Antonia Ghiselli, Luca Magnoni, Riccardo Zappi

† INFN-CNAF, Viale Carlo Berti Pichat, 6/2, I40127, Bologna, IT
‡ ICTP Trieste, Strada Costiera 11 34014, Trieste, IT
E-mail: {alberto.forti, antonia.ghiselli, luca.magnoni, riccardo.zappi}@cnaf.infn.it
E-mail: {ezio.corso}@ictp.it

Abstract.
In a Grid environment the naming capability allows users to refer to specific data resources in a physical storage system using a high level logical identifier. This logical identifier is typically organized in a file system like structure, a hierarchical tree of names. Storage Resource Manager (SRM) services map the logical identifier to the physical location of data evaluating a set of parameters as the desired quality of services and the VOMS attributes specified in the requests.
StoRM is a SRM service developed by INFN and ICTP-EGRID to manage file and space on standard POSIX and high performing parallel and cluster file systems.
An upcoming requirement in the Grid data scenario is the orthogonality of the logical name and the physical location of data, in order to refer, with the same identifier, to different copies of data archived in various storage areas with different quality of service.
The mapping mechanism proposed in StoRM is based on a XML document that represents the different storage components managed by the service, the storage areas defined by the site administrator, the quality of service they provide and the Virtual Organization that want to use the storage area. An appropriate directory tree is realized in each storage component reflecting the XML schema. In this scenario StoRM is able to identify the physical location of a requested data evaluating the logical identifier and the specified attributes following the XML schema, without querying any database service.
This paper presents the namespace schema defined, the different entities represented and the technical details of the StoRM implementation.

1. Introduction
High energy physics data-intensive applications demand large storage systems capable of serving hundreds of terabytes of storage space in a fast and reliable way, with high throughput and easy management. Nowadays, Storage Area Network (SAN) solutions are commonly deployed at LHC centres, and parallel file systems such as GPFS [1] and Lustre [2] allow for reliable, high-speed native POSIX I/O operations.
In the near future, approximately one hundred of petabytes will be produced and stored in multiple storage system geographically dispersed. The naming capability allows users to refer to specific data resources in a physical storage system using a high level logical identifier. The identifier is defined by Virtual Organizations and is typically organized in a file system like structure, a hierarchical tree of names. A new requirement coming from Grid applications is
the orthogonality of the logical name and the physical location of data, in order to refer, with
the same identifier, to different copies of data archived in various storage areas with different
quality of service.

The various Storage Resource Manager services have different approaches to address this
requirement; usually it’s based on a persistence service to store the mapping between the high
level logical identifier and physical location of data. The different approach proposed by StoRM
is based on the evaluation of a XML document that represents the different storage components
managed by the service, the storage areas defined by the site administrator, the quality of service
they provide and the Virtual Organization that want to use the storage area. An appropriate
directory tree is realized in each storage component reflecting the XML schema. In this scenario,
StoRM is able to identify the physical location of a requested data evaluating the logical identifier
and the specified attributes following the XML schema, without querying any database service.

In this paper we introduce the Storage Resource Manager paradigm, we describe the StoRM
project, we introduce the storage area and storage space management defined by the advanced
SRM v2.2 functionalities and the requirement coming from the WLCG context. The namespace
component implemented in StoRM is then described as an efficient solution based on a well
defined XML schema and appropriate file systems structures.

2. Storage Resource Manager services and StoRM

In a Grid environment the capability to use standard interface to manage heterogeneous storage
resources is fundamental to gain real interoperability. A storage service interface must be flexible
efficiently to be used to access storage systems based on different storage technologies. At the
same time, its usage should be simple enough so that clients do not need to have any knowledge
of the underlying storage back-end. This is the main goal of the Storage Resource Manager
(SRM) interface, a middleware service aiming to provide the dynamic management of a storage
resource engaged in a distributed computing system. A detailed description of SRM service and
its specifications is given in references [3, 4].

In the WLCG context exists many SRM implementation, this article describes the Grid
Storage Resource Manager StoRM [5]. StoRM has been developed with the specific aim of
providing access to parallel file systems like GPFS and Lustre, but also standard POSIX file
systems, in Grid through a SRM interface.

2.1. The SRM paradigm

The Storage Resource Manager (SRM) paradigm is nowadays a building-block of the storage
requirements in the computing models of the HEP experiment. The SRM aims to provide the
dynamic management of a storage resource engaged in a distributed computing system. The
Open Grid Forum Grid Storage Management (OGF-GSM) working group has defined a standard
interface (SRM v2.2) to hide storage dependent characteristics and to allow interoperability
between different storage systems.

The SRM functionalities can be grouped in the following categories:

- **Data management**, this group of functions provides the capability to manage the data
  stored in a SRM; data can be prepared for access operation (staging, etc.) or the
  environment can be set up for receive a new file.

- **Space management**, this functionalities manage the space into storage resources; space
  can be reserved by user to have the guarantee of availability when the transfer operation
  takes place.

- **Directory management**, this is a set of UNIX-like directory operation used to manage
  the SRM namespace creating, moving and deleting directories and files.
• **Query functionalities**, this set of function is used to discover information on a specific SRM endpoint.

SRM systems can store the data in several physical locations depending on the storage system characteristics. A specific data can be brought from tape to disk to provide a direct access to client.

### 2.2. StoRM

The StoRM project has been developed by a collaboration between the "Istituto Nazionale di Fisica Nucleare (INFN)” and the “the Abdus Salam International Centre for Theoretical Physics (ICTP)” in Trieste, the latter operating in the framework of the EGRID project [6]. The main idea behind StoRM [5], [7] is to create an SRM implementation to leverage the advantages of high performance parallel file systems in a Grid environment. It provides users and applications with the standard SRM v2.2 functionalities combined with the capability to perform secure and local accesses (the file:// transfer protocol is supported) to the shared storage resources. StoRM decouples the file system service from the SRM service, in the sense that the data-access functionality can be provided by any POSIX file system, while StoRM specifically provides just the SRM functionality.

High performing cluster an parallel file systems usually provide specific API to use advanced management functionalities, in this cases a specific driver is plugged to StoRM to take advantages from the functionalities available. With this driver approach StoRM can be easily configured to run on the GPFS or Lustre parallel file systems, which provide POSIX access to the data and are widely employed commercial products.

StoRM provides the advanced SRM functionalities to dynamically manage space and files according to the user requirements, to specify the desired lifetime and to allow for advance space reservation and a different quality of service provided by the underlying storage system.
Since it generalizes the file system access to the Grid, StoRM also takes advantage from the security mechanisms of underlying file system to ensure an authenticated and authorized access to the data.

StoRM has a multi-tier architecture, the figure 1 gives a schematic view of the main component. The Front-End (FE) exposes the service interface and manages clients authentication. The Back-End (BE) is the core of StoRM, it processes the SRM requests managing metadata and authorization aspects and interacting with other Grid services. It supports different file systems through a driver mechanism easy to expand and improve. Each component can be instanced and replicated in a dedicated host, to satisfy the scalability and availability requirements.

2.3. SURL and TURL
A site URL (SURL) is a logical identifier that provides an abstraction of the file namespace to address a data in a specific SRM service. A transfer URL (TURL) is a physical identifier for data access. It depends on the access protocol specified by client and on the physical location of data in the storage system. A specific data can be brought from tape to disk to provide a direct access to client.

Figure 2 show the structure of SURL in query mode, it is composed by the prefix, that is the srm string for all SRM services, by the authority, made by the service hostname and port, by the service path and, in the specific case of query form the value for the Storage File Name (SFN) expressed with the http get format. The Storage File Name is a logical identifier for the data, it has a file system like structure that can be different on the real location of data into storage. The second part of 2 show the structure of TURL in case of the gsiftp protocol, the general format depends on the transfer protocol the client specified. The Physical File Name (PFN) represent the real position of the data into the storage resources.

To access a file through a SRM service, a SURL is provided as a parameter embedded
in the specific SRM request, such as the \texttt{srmPrepareToPut} method for preparing the SRM to accept new data and the \texttt{srmPrepareToGet} method to get read access to the wished data resource. The SRM interface provides two classes of methods: asynchronous and synchronous ones. Asynchronous methods return a token corresponding to the request, and the corresponding action is performed by the SRM asynchronously. The client can retrieve at any time the status of the request by addressing it through such a token. This is the case for data management functionalities. Synchronous requests return at completion giving back the control to the client (\textit{blocking calls}). This is the case of directory and file management (e.g. \textit{srmLs}, \textit{srmMkdir}, \textit{srmRm} and \textit{srmRmdir}) and space management functions (e.g. \textit{srmReserveSpace}).

3. Storage Area and Storage Component

A storage element is a Grid service that provides users with capabilities to manage, transfer and access data and space in a Grid environment. Several new concepts have been introduced to model the advanced data management features provided by SRM interface v2.2, to satisfy the requirements coming from LHC experiments in terms of storage system and storage solutions available. WLCG users ask for the capability to identify the quality of service a storage resource is able to provide, in terms of quality of retention and access latency, or to define the characteristics of a specific space assigned to a certain Virtual Organization. Entities introduced in this process have been defined and used into the GLUE schema to model the storage element services.

3.1. Storage Component

A Storage Component (SC) defines a storage system in terms of specific properties that model the storage characteristics. A storage component is strictly related with the physical storage structure, in example a SC can represent a tape system or a cluster of disks. The properties of a storage component are:

(i) \textbf{Retention Policy}. This property represents the quality of retention a storage system is able to provide, in term of probability of data loss once stored. The value can be: \textit{custodial}, \textit{replica} or \textit{output}.

(ii) \textbf{Access Latency}. This property represents the latency on data access operations, it depends for example on staging operation or disk performance. The value can be: \textit{nearline}, \textit{online} or \textit{offline}.

(iii) \textbf{Access Protocol} This property represents the access protocol a client can use to access data into the storage system. Allowed value can be: \textit{rfio}, \textit{dcap}, \textit{file}, etc.

3.2. Storage Area

A Storage Area (SA) defines a logic partition of the total available space on different storage components within a Storage Element. A Storage Area (SA) can be created on various storage components, in WLCG the SA is created by Virtual Organization Managers and is identified per VO by a Space Token Description (SA_Token), which is optionally published in the Information System. In WLCG use case a SA implements a Storage Class instance.

3.3. Storage Class

A Storage Class defines a classification of storage system in terms of Retention Policy and Access Latency. Asking for a certain storage class users can specify the desired storage system characteristics to store data. In WLCG has been decided to overload the meaning of term introduced by SRM mapping the quality of storage together with the combinations of storage devices. The storage devices are described as Tapes (or reliable storage system always referred
Figure 3. Example of a Storage Area on three Storage component based on three GPFS file systems.

3.4. The orthogonality from path and token

In the WLCG approach for the SRM v2.2 functionalities the SURL is the main identifier for a data, it has a file system like structure, with a well defined Storage Paths, and its name space have to be orthogonal to the Storage Class in which the desired file is finally stored.

Storage Classes classify the different quality of service an SRM system is able to provide. The user can ask for desired Storage Class specifying the Storage Area Token when the file is stored. In the WLCG use case the Virtual Organization administrator creates the Storage Area for the desired VO and assign to it a proper Space Token. Users can migrate copies of the files to different storage areas through an SRM ChangeSpaceForFiles operation. The SURL does not change when the file is migrated in example from tape to disk, this means SURL with the same structure can refer to files on different storage area and storage components.

To address these requirements many SRM implementations rely on a persistence layer to store the mapping between the Site URL for the file, the Storage Area Token, and the physical location of data into storage systems. For every SRM request a query to the database has to be done in order to establish the physical position of file and build the correct transfer URL. StoRM has a different approach, this mapping functionality is provided by a dedicated
Figure 4. Example of information represented in the StoRM namespace schema in a real WLCG use case.

The StoRM namespace component provides the capability to identify a specific data resource in the storage systems using a high level logical identifier specified by the user (the SURL). The standard approach is to query, for each SRM request, a persistence service to retrieve the mapping between the high level identifier and the physical location of the data in the storage component. The StoRM namespace acts in a different way, it relies on the underlying file system structure and the mapping are done without interacting with any database service but evaluating the configuration file namespace.xml and the input parameters (SURL and user credential and attributes).

This means that the same logical identifier could refers to different copy of data archived in different storage area, with possibly different quality of services. The advanced StoRM namespace component improves the mapping mechanism used to identify the physical location of the data on a specific storage area and storage component evaluating the SURL, the Storage Area token and the user credential. This is done by representing the storage area and storage component structure in a XML schema and defining an appropriate mapping algorithm to identify the candidate storage components.
Figure 5. Example of the directory tree replicated in each Storage Components belonging to a specific Storage Area.

Figure 4 is an high level view on the XML schema represented in the configuration file in case of a real Grid site. There are various storage component made by different kind of disk system, there are the different Storage Area and there are the different VOInfo that represents how the different VOs can access to the desired storage area, in term of space token and access control based rule.

4.1. The mapping algorithm
The formula at the end of Fig.4 represents the algorithm to identify the possible Physical File Names, then the physical locations on storage systems candidate to contains the desired data. The formula defines a directory tree (see example in Figure 5) that depends on the Storage Component path, the Storage Area path and the Access Control Based Rule (e.g. the user credential, in terms of Virtual Organization and VOMS attributes). The set of possible physical location is calculated applying the established path on each storage component composing the selected Storage Area.

At storage resource level, the directory tree is reproduced on each storage components belonging to the same storage area. Once the set of possible storage component candidate to contain the data is established, a lookup mechanism is used on each component to find the exact location of the data. Finally the TURL is built in according with user specification. In this way the same SURL can refer to different physical file with different quality of services. The lookup mechanism has only to check for the data on the well defined path on each component selected.

The schema on Figure 4 is formally represented in a XML structured file named namespace.xml. This file contains all information needed by the mapping algorithm, the storage systems structure, and the different storage area characteristics. The namespace.xml is parsed at StoRM startup time and each mapping request is satisfied at real time evaluating this information.

This approach allows StoRM to satisfy the requirement of orthogonality between Storage Area and path coming from the SRM interface v2.2 and from the WLCG use case. This is done by the namespace component without relying on a persistence service but using a well defined directory schema. This directory schema model the different parameter involved in the WLCG data access pattern and allow user to refer with same SURL to different file stored in different Storage Area, specifying the appropriate space token, as shown in Fig. 5.
4.2. Lookup mechanism

The formula determines the set of possible physical paths where the real data can reside, but there is the need to have an efficient and performing lookup mechanism to verify the real location of data. In StoRM this capability is integrated into the namespace component. The lookup mechanism can be configured in two ways, as sequential or parallel process. The sequential configuration means that for each SRM request the lookup operation take place in a sequential way, visiting all storage component candidates to contains data until the file location is found. The parallel configuration has a different approach. A set of lookup process, on for each storage component, is activated at StoRM startup time. Each process is dedicated to the lookup operation on a single storage component. When the lookup is needed, the namespace component query all the process with a parallel request. The lookup process manage the query, lookup for the file and send the response to the namespace component, in this way the correct location is discovered in an efficient way and the transfer URL is built and returned to client.

5. Conclusion

Storage management capability is one of the most important feature in modern Grid system. A standard interface for storage management is needed by user and application. The SRM interface version 2.2 has been adopted in WLCG, and the new concepts of Storage Area, Storage Class and Storage Component has been introduced to model the evolution of storage resources. The requirement of orthogonality between SURL path and the Storage Area is coming from this evolution of storage resources. In this paper we have shown how StoRM satisfy this requirement. StoRM has been designed to leverage the advantages of high performing parallel and cluster file system in the Grid environment. The namespace component provides the naming functionality, the capability to locate the physical position of data into the different storage resources. The namespace component satisfies the orthogonality requirement relying on the storage system structure and evaluating an XML file, named namespace.xml. The namespace.xml contains all the information on the Storage Component managed by StoRM, the Storage Area defined and the VO that want to access data. With this approach StoRM discovers the physical location of the data into storage resources without querying any persistence service but evaluating the SRM parameters together with the namespace XML schema, and using an efficient lookup mechanism on each storage component.

References

[1] 2007 GPFS: General Parallel File-System http://publib.boulder.ibm.com/infocenter/
[2] 2007 Lustre  http://www.lustre.org
[3] 2007 SRM Working Group  http://sdm.lbl.gov/srm-wg
[4] 2007 The Storage Resource Manager Interface Specification, Version 2.2
   http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html
[5] 2007 StoRM: Storage Resource Manager  http://storm.forge.cnaf.infn.it
[6] 2006 EGRID project. http://www.egrid.it/
[7] Corso E, Cozzini S, Forti A, Ghiselli A, Magnoni L, Messina A, Nobile A, Terpin A, Vagnoni V and Zappi R
   2006 StoRM: A SRM Solution on Disk Based Storage System URL http://www.cyfronet.pl/cgw06/