StoRM: A SRM solution on disk based storage system.

E. Corso[2], S. Cozzini[2,4], A. Forti[1], A. Ghiselli[1], L. Magnoni[1], A. Messina[2], A. Nobile[2], A. Terpin[2], V. Vagnoni[3], R. Zappi[1]

[1] INFN-CNAF, Viale Berti Pichat 6/2, 40127, Bologna (BO), Italy
[2] The Abdus Salam ICTP, Strada Costiera 11, 34014, Trieste (TS), Italy
[3] INFN Sezione di Bologna, Via Irnerio 46, 40126, Bologna (BO), Italy
[4] CNR/INFM Democritos, via Beirut2/4, 34014, Trieste (TS), Italy

Abstract

Today's data intensive applications demand large storage systems capable of serving hundreds of terabytes of storage space. A Storage resource can be composed by different storage systems: disk only systems, tape archiving systems, or by a combination of both. Storage Resource Managers (SRMs) are middleware services whose function is to provide dynamic space allocation and file management of shared storage components. The access to storage resources needs to be fast. Even though tape servers can provide the desired capacity, they do not satisfy the performance requirements. The StoRM project is an implementation of the Storage Resource Manager interface version 2 for disk based storage solutions. StoRM allows applications to access files on a file-system directly via POSIX calls, whether scheduled via Grid or only through the local batch system, in a transparent way with the guarantee to find the required disk space. In a Grid environment security is a fundamental aspect. StoRM authentication is based on VOMS certificates, different information sources for authorization are supported (using plug-ins) and file system ACLs (Access Control Lists) are used to enforce permissions. The authorization model adopted by StoRM is also designed to cater for the interests of Economics and Finance as represented by the EGRID Project, given that security is an important driving requirement.

Here we present a solution based on GPFS file system from IBM. It is a parallel file system highly scalable in number of nodes and disks that offers POSIX semantics, allows for large volumes creation and provides extensive fault tolerance and recovery possibility.

1 Introduction

From the beginning of the Grid one of the main purposes was to provide a way to share geographically distributed heterogeneous resources, giving the illusion to the user to work on a local system. In a data Grid, in which more emphasis is given on data intensive applications that produce and read large volumes of data, the need to provide an efficient management of the storage resources becomes fundamental. A Storage resource can be composed by different storage systems: disk only systems, tape archiving systems, or by a combination of both. The basic logical entities of a storage resource are space and file. Space must be allocated when a new file has to be stored into a storage resource, and files could be dynamically removed to create the necessary space. This is the main goal of Storage Resource Managers (SRMs), middleware services whose function is to provide dynamic space allocation and file management of shared storage components. SRMs services agree on a standard interface to hide storage dependent characteristics and to allow interoperability between different storage systems.

In the HEP (High Energy Physics) community high performance storage resources based on disk storage solutions using parallel file systems are becoming increasingly important in order to provide reliability and high speed I/O operations needed by HEP analysis farms. In this article we describe the StoRM project, an implementation of the Storage Resource Manager interface version 2 for disk based storage solutions. StoRM is designed to take advantage from parallel file systems, but also standard POSIX file systems are supported. StoRM provides space reservation capabilities and uses native high performing POSIX I/O calls for file access. Besides the standard Grid protocols, also the file protocol is supported.

In the next section 2 we give a brief overview of the SRM specification v2.2. Section 3 describes StoRM, in particular section 3.1 briefly explains the architecture of StoRM, section 3.2 summarizes the main characteristics of parallel and cluster file systems stating how StoRM leverages their features and section 3.3 focuses on a use case with the GPFS file system (IBM). In section 4 we describe the security framework of StoRM, that is an important driving requirement. Finally the conclusions in section 5.

2 Storage Resource Managers

Storage Resource Managers [9] (SRM) are middleware services whose function is to manage the dynamic use of space and files in a storage resource on the Grid. From one side a client can request to the SRM to reserve space and to manage files and directories, from the other side a SRM can dynamically decide which files to keep in the storage space and which files to remove (e.g. when free space is needed). Therefore, files are no longer permanent entities on the storage, but dynamical ones that can appear or disappear according to the user specification.

SRM relies on some basic concepts concerning space and files. A file can be volatile, permanent or durable. A *volatile* file is a temporary file with a lifetime associated. When the lifetime expires the file can be deleted by the garbage collector of the SRM. Therefore, a client can *pin* a file for a specific amount of time (lifetime of the pinning) and be sure to have the file available for the whole operation time. A *permanent* file can only be removed by the owner. A *durable* file has a lifetime associated with it, but has a different behaviour when the lifetime expires. It can be removed by the garbage collector only after confirmation by the owner. In Grid a file is identified by several terms: the

Logical File Name (LFN) is a user friendly (and user defined) name and there can be several LFNs (aliases) for one file. The *Storage URL* (SURL) points to the physical stored version of a file on a grid storage element. The *Transport URL* (TURL), or Transfer URL is an URL that a SRM issues for a SURL which can be used to retrieve or store data (e.g. with GridFTP). Finally the *file catalog* is the grid middleware server which maps LFNs to SURLs, i.e. tracks the real physical locations of the logical file names.

SRMs provide *space reservation* in order to support the request planning and request execution steps of running jobs on the Grid by guarantying a desired amount of space in the storage.

The concept of *storage classes* is used to distinguish different properties of the storage systems managed by a SRM. For example a user can choose to create a file in a slow but reliable tape based system or in a faster disk based system.

SRMs are implemented as web services and they agree on a standard interface to hide storage characteristics and to allow interoperability between different implementations.

3 StoRM

StoRM [4] is a storage resource manager for disk based storage systems implementing the SRM interface version 2.2 [5]. It is designed to support guaranteed space reservation and direct access (native POSIX I/O calls) to the storage, supporting as well other standard access libraries like RFIO [6]. This allows applications to perform a standard POSIX operation without interacting with any external service that emulates data access, with the result of improving the performance when the underlying file system is efficient.

StoRM takes advantage from high performance parallel and cluster file systems, as described in section 3.2 and 3.3. More generally StoRM works on top of any standard POSIX file system with ACL (Access Control List) support, like XFS and ext3. Indeed, StoRM uses the ACLs provided by the underlying file systems to implement the security model.

3.1 StoRM Architecture

StoRM has a multilayer architecture made by two main components: the *frontend* and the *backend*. The frontend exposes the SRM web service interface, manages user authentication and stores the requests data into the database. The backend is the core of StoRM service since it executes all synchronous and asynchronous SRM functionalities. It takes care of file and space metadata management, enforces authorization permissions on files and interacts with file transfer services. Moreover the backend is able to use advanced functionalities provided by the file system (for example by GPFS and XFS) to accomplish space reservation requests. The backend logic is decoupled from the wrapper component that provides file system support. This plug-in mechanism allows to easily add new support for different file systems.

In StoRM the database is used to store request information and metadata of files and space. One or more instances of the StoRM components can be deployed on different machines using a centralize database service. StoRM has a light and flexible namespace mechanism that leverages the underlying file system structure (i.e. it is the file system that knows where files are stored and this information is not replicated into a database), hence StoRM does not need to query the database to know the physical location of a file that can be deduced from the requested SURL.

3.2 Cluster File System

A cluster file system allows large numbers of disks attached to multiple storage servers to be configured as a single file system, providing:

- Transparent parallel access to storage devices while maintaining standard UNIX file system semantics.
- High-speed file access to applications executing on multiple nodes.
- High availability and fault tolerance.

StoRM takes advantage from aggregation functionalities provided by such kind of dedicated storage systems. It is designed to implement the SRM functionalities allowing the user to leverage the direct access and the performance improvement provided by parallel file systems.

3.3 StoRM and GPFS

IBM General Parallel File System (GPFS) [7] is a high-performance shared-disk cluster file system. GPFS distinguishes itself from other cluster file systems by providing concurrent high-speed file access to applications executing on multiple nodes. With GPFS a single file system containing Petabytes of data can be built on dedicated hardware in a Storage Area Network (SAN) configuration. A SAN is a network designed to attach computer storage devices to servers with high-performance connections, like fiber channels. In a Grid site, where every worker node in the cluster cannot have SAN access to disks, GPFS makes use of a capability called NSD (Network Shared Disk), which provides a software simulation of a SAN. GPFS uses NSD to provide a capability to access data at high speed to applications running on compute nodes which do not have a SAN attachment.

The picture in fig. 1 shows a grid site with a GPFS cluster and StoRM as a storage resource manager. The standard way to read and write data during the computational process has many drawbacks (Site A in fig. 1):

- In write operations usually a job produces data in the local worker node disk and then it has to copy the data into the storage element. During this process, several problems can arise preventing the operation to complete successfully: not enough free space on the local disk, local disk filled up by different jobs running on the same worker node, a network failure during the transfer operation, data removed by some other job, etc.

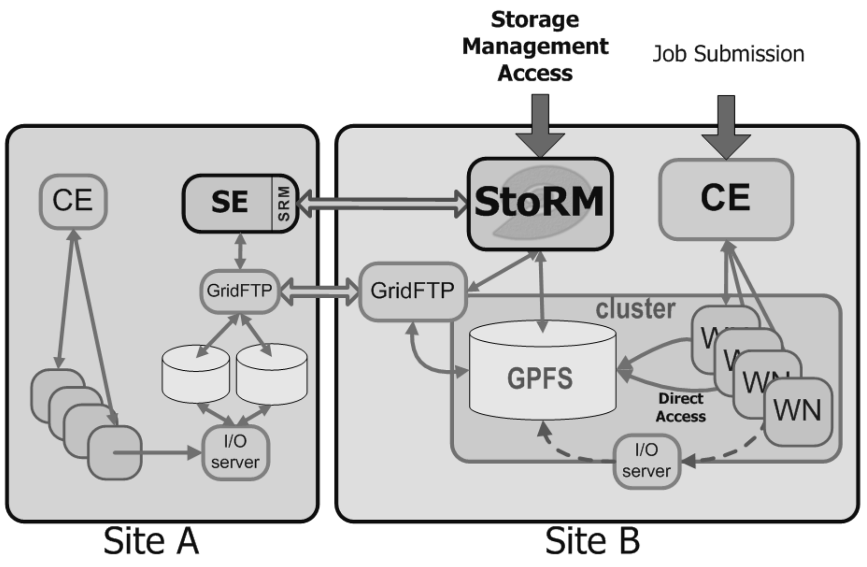

Fig. 1: Grid site with GPFS and StoRM.

- To read a file without copying it locally, the only way is to use some external I/O server that provides a remote POSIX-like access to the data.

The proposed solution (Site B in fig. 1) provides an efficient, secure and reliable way to access data through standard POSIX calls. A job running on a worker node belonging to the GPFS NSD has direct access to the desired file into the Storage Element. In this scenario StoRM assures that the user data will be available for all the time the access operation goes on. Pinning the file for a desired amount of time (lifetime) avoids early removals and on the other side space reservation guarantees the space availability for the whole writing process. StoRM also provides a garbage collector to clean up unused files with expired lifetime.

4 Security aspects

In the sharing resources among multi-domain context, the security aspect is a lite motif. Data integrity, confidentiality, identification, data encryption, non-repudiation, auditing, authorization, and authentication are only some storage security issues. Anyhow, the concept of identification, authentication, and authorization are quite central to storage security. In all these concepts the most frequently used terms are subjects and objects. A security subject is an active entity (a user or a running process, sometimes also called a principal) that consumes resources and manipulates passive objects in the system, whilst an object is a passive entity that represents some resource. Subject identification is about stating who users are and authentication is about proving the identification claim. Once authenticated, the authorization mechanism decides what users or programs can do on the storage resource (i.e. security object) and what they cannot. In grid context, secure access to a physical file involves many steps. Firstly, the principal have to own a valid identity defined into a virtual organization. Then the subject has to be authorized to query the VO file catalog in order to get authorization information about the wished file and to obtain the SURL. The knowledge of the SURL allows pointing to the right storage element and to the SRM service that holds and manages it. At last, the grid user must be authorized to access the SRM service to obtain the TURL pointer. Nevertheless, security must be enforced also at file level to prevent a malicious user to bypass the SRM secure level and access the physical file directly (e.g. knowing the URL). Several security aspects belong to the virtual organization domain, whilst other belongs to the site administrative domain. Principal identity definition, secure catalogues access and service authorization policies definition belong to the VO domain. Subject authentication, service authorization, and enforcing of access authorization policies on physical files belong to the site domain. Each aspect must be taken into account at different levels, site and virtual organization, in order to build an effective secure context.

4.1 StoRM security model

The security model adopted by StoRM is qualitative since it permits or denies the user to access (intended as perform whatever actions) a particular file or directory and space. StoRM security model relies on the security model implemented in the underlying file system. Since the physical file on storage represents the final target of the principal and since an end user can bypass StoRM security implementation performing a direct access to files, the security mechanism provided by the file system to enforce permission represents the foundation and the ultimate security defence.

StoRM service behaves as a proactive actor. This means that StoRM has the ability to find out which actions the subject can perform on a storage resource even before presenting a way to access it. In addition the underlying secured file system operates in a reactive authorization mode; users have to try to access a resource to determine whether they are authorized for it. Moreover, to allow more complex security scenario, StoRM requires more than a single access control entry for every file. The ACLs on disk's filesystem looks like the natural mechanism for enforcement. StoRM requires file systems with ACL support to provide a complete security access scenario.

4.2 Security Subject: Grid user and local user

The Grid user is the subject who is authenticated and authorized. Operating systems today have no specific knowledge of "Grid Users" and in the Grid environment an application runs on a machine as a local user, but on behalf of a Grid user. Since the final access enforcement is based on the local user, StoRM interacts with an external mapping service to map the Grid user to a local user having the same authorization permissions.

4.2.1 Grid user authentication

StoRM uses the Grid Security Infrastructure (GSI) [8] for authentication. Some Fully Qualified Attribute Name (FQANs) could be present within the X.509 proxy certificates, as defined by the Virtual Organization Membership Service (VOMS) [1]. VOMS is an Attribute Authority (AA), in which users can be organized in a hierarchical structure with groups and subgroups. Upon a request from a grid user belonging to a managed Virtual Organization (VO), the VOMS service produces a VOMS proxy certificate, that is a X.509 Attribute Certificate Extension. The user must have a valid proxy certificate to submit the SRM request to the StoRM server. In this scenario users are authenticated and authorized to use the resources against their proxy certificate, their role and the VO they belong to.

4.2.2 Mapping grid user to local user

The access restriction to storage resources is based on the mapping from a grid identity to a local user identifier (ID). The grid identities mapping is accomplished by a call to the LCMAPS [10] service, so at runtime every Grid user is associated to a local user ID.

4.3 Security Object and permission enforcement

StoRM interacts with different external authorization services (authorization sources) to verify if the user can perform the specified operation on the requested resources. Typically, authorization policies within the authorization sources are expressed in terms of Grid identities and SURLs. This kind of information can be managed by dedicated services (as the INFN project G-PBox [2], a policy management framework for Grid environments) or resides in a more general purpose catalog (as LFC, the LCG File Catalog [3]). Each authorization source is queried with a specific order and the results are evaluated with a deny-override algorithm. Once the user request has been authorized, StoRM enforces the permission on the data by setting a file system ACL for the corresponding user on the specified resource. StoRM sets up ACLs on the physical file to allow direct access from a worker node. A new access control entry (ACE) of the ACL is built from the local user to whom Grid credentials are mapped and with the permission returned by the authorization source. The ACE added will be removed once the data access completes or when the access operation lifetime expires.

The StoRM security model contemplates two ways to manage the ACLs on files. The *"Just in Time"* (JiT) model defines and enforces the ACEs when a new access request arrives and the ACE is removed when the access is terminated or when the lifetime of the access policy is expired. The *"Ahead of Time"* (AoT) model defines and enforces the ACEs when an authorization policy is defined and the ACE is removed only when the authorization policy expires.

5 Conclusions

StoRM is a lightweight implementation of the standard SRM interface specification. It is designed to work on top of any POSIX file system and to take advantage from special features offered by parallel and cluster file systems (e.g. GPFS). Parallel and cluster file systems deployed on a Storage Area Network (SAN) are capable to serve multi-terabyte storage, offering a scalable, high performance system with reliability in case of disk failures, and with the possibility of great performance on multiple accesses on files from different nodes. Differently from other SRM implementations existing today StoRM decouples the management of storage resources (leaving to the file system the aggregation of different storage volumes) and the implementation of SRM functions and does not force applications to use custom file access protocols (e.g. RFIO and d-cap). The authorization mechanism based on file systems ACLs allow standard Grid applications and protocols (e.g. GridFTP, RFIO, etc.) to be used without been modified. Security is an important aspect on Grid storage management. StoRM supports VOMS certificates and has a flexible authorization framework allowing to use plug-ins for different authorization models.

References

1. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lörentey, and F. Spataro. Voms: An authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference*, Santiago de Compostela, 2003.
2. A. Caltroni, V. Ciaschini, A. Ferraro, A. Ghiselli, G. Rubini, and R. Zappi. G-PBox: A Policy Framework for Grid Environments. In *Proceedings of the International CHEP 2004*, Interlaken, Switzerland, 2004.
3. LCG File Catalog. https://uimon.cern.ch/twiki/bin/view/lcg/lfcadminguide.
4. E. Corso, S. Cozzini, F. Donno, A. Ghiselli, L. Magnoni, M. Mazzucato, R. Murri, P.P. Ricci, H. Stockinger, A. Terpin, V. Vagnoni, and R. Zappi. StoRM, an SRM Implementation for LHC Analysis Farms, Computing in High Energy Physics. In *Proceedings of the International CHEP 2006*, Mumbai, India, Feb 2006.
5. Storage Resource Managament Working Group. http://sdm.lbl.gov/srm-wg/.
6. RFIO/IB Project. http://hikwww2.fzk.de/hik/orga/ges/infiniband/rfioib.html.
7. F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System For Large Computing Clusters. *FAST'02*, pages 28–30, January 2002.
8. Globus Toolkit Security. http://www.globus.org/toolkit/docs/4.0/security/.
9. A. Shoshani, A. Sim, and J. Gu. Storage resource managers. In *Grid Resource Management*, chapter 20, pages 321–340. Kluwer Academic Publishers, Norwell, MA, USA, 2004.
10. M. Steenbakkers. Guide to lcmaps version 0.0.23. http://www.dutchGrid.nl/DataGrid/wp4/lcmaps/edg-lcmaps_gcc3_2_2-0.0.23/, 2003.