STORM, AN SRM IMPLEMENTATION FOR LHC ANALYSIS FARMS

E. Corso[1],S. Cozzini[2],F. Donno[3],A. Ghiselli[4],L. Magnoni[4], M.Mazzucato[4],
R. Murri[1],P.P. Ricci[4],H. Stockinger[5],A. Terpin[1], V. Vagnoni[6],R. Zappi[4]
1) The Abdus Salam ICTP, Trieste, Italy - 2) CNR-INFM Democritos, Trieste, Italy
3) INFN, Pisa, Italy and CERN, Geneve, Switzerland - 4) INFN/CNAF, Bologna, Italy
5) University of Vienna, Vienna, Austria - 6) INFN, Bologna, Italy

Abstract

In production grids, high performance disk storage solutions using parallel file systems are becoming increasingly important to provide reliability and high speed I/O operations needed by HEP analysis farms. Today, Storage Area Network solutions are commonly deployed at LHC center and and parallel file systems such as GPFS and Lustre allow for reliable, high-speed native POSIX I/O operations in parallel fashion.

In this article we describe the status of the StoRM project, an implementation of the Storage Resource Manager (SRM) standard interface version 2.1.1 for disk based storage solutions. StoRM is designed to work over native parallel filesystems, provides for space reservation capabilities and uses native high performing POSIX I/O calls for file access. StoRM takes advantage of ACL support provided by the underlying filesystem to implements the security model.

StoRM is also designed to cater for the interests of Economics and Finace as represented by the EGRID Project, given that security is an important driving requirement.

We report on the tests performed on a dedicated test bed to prove basic functionality and scalability of the system together with GPFS performance on write and read operation from multiple nodes.

INTRODUCTION

LHC [1] analysis farms - present at sites collaborating with LHC experiments - have been used in the past for analyzing data coming from an experiment's production center. With time such facilities were provided with high performance storage solutions in order to respond to the demand for big capacity and fast processing capabilities. With the advent of Grid technologies, existing LHC analysis facilities have to face the problem of adapting current installations with Grid requirements to allow users to run their applications both locally and from the Grid in order to provide efficient usage of the resources.

Parallel file systems deployed on 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 access on files from different nodes. Filesystem such as GPFS [2] and Lustre [3] with such characteristic inside a grid environment allow for reliable, high-speed native POSIX I/O [4] operations.

The need to provide grid applications with consistent and efficient wide-area access to heterogeneous storage resources drives to use the Storage Resource Manager (SRM) [5] interface as the common standard for storage management. As of today SRM implementations exist for storage managers such as Castor, d-Cache and LCG DPM. However, such solutions manage the entire storage space allocated to them and force applications to use custom file access protocols such as *rfio* and *d-cap*, sometimes penalizing performance and requiring changes in the application.

StoRM [6] is a disk-based storage resource manager that implements SRM interface v.2.1.1. It is designed to work over native parallel filesystems or with standard POSIX filesystem, providing space reservation capabilities and allowing to use native high performing POSIX I/O calls for file access.

Moreover StoRM is also designed to cater for the interests of Economics and Finace as represented by the EGRID Project, given that security is an important driving requirement.

StoRM takes advantage of ACL support provided by the underlying filesystem to implements the security model.

In this article, we describe the StoRM project and the features provided by the current release. We report on the tests performed on a dedicated test bed to prove basic functionality and scalability of StoRM together with GPFS performance in standard POSIX operation from multiple nodes.

STORM SERVICE

Architecture

Parallel file systems offer high scalability and high availability by allowing multiple servers and multiple disks to serve the same file system, providing to application a single view of a unique file system.

StoRM (acronym for Storage Resource Manager) is an SRM server that provides functionalities for space and file management of disk based storage systems. The current release of StoRM provides a subset of the SRM v2.1.1 functionalities:

- **Data transfer**: srmPrepareToPut(), srmPrepareToGet(), smrCopy(), srmStatus*(), srmPutDone().
- **Space management**: srmReserveSpace(), srmGetMetaDataSpace().
- **Directory management**: srmLs() with recursive option, srmRm(), srmRmdir(), srmMkDir().

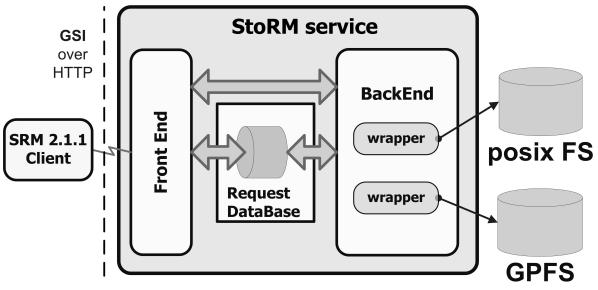

Figure 1: StoRM Architecture

The StoRM architecture can be identified by two main tiers named Front-End and Back-End:

- **Front-End (FE)** exposes the service interface. It receives client requests, manages the client communication, and forwards a processed structure to server.
- **Back-End (BE)** is the core of StoRM. It processes the requests, executes necessary tasks to perform operations concerning security, metadata, space and file management and returns the result to the client.

In order to provide access to different filesystem, StoRM makes use of wrappers, called by the Back-End. These wrapper also hide the complexity of filesystem specific functions. However, StoRM takes also advantage of special features provided by underlying filesystems like ACL support, and block pre-allocation.

From the beginning of the StoRM project a particular attention has been given to the space reservation functionality. Using file systems specific features, such as the block pre-allocation available in GPFS, StoRM is able to provide a guaranteed space reservation to application, allocating empty block which will be used for write data.

The current release of StoRM allows to use two kinds of protocols: *gsiftp* and *file*. Support for the file protocol allows applications to perform a standard POSIX operation (returning a special URL) without interacting with any external service that emulates data access, improving performance, when the underlying file system is efficient. Therefore, an application executed on a grid system can be adapted without any change in data access pattern.

Service configuration

For a grid site the possibility of configuring a storage management system in specific way is fundamental. StoRM provides an easy way to configure parameters regarding both the SRM interface and StoRM's internal features, allowing to create an ad-hoc installation for any site. StoRM can be easy configured to manage file namespace within a VO domain (at the site level) by editing a single configuration file. Many other specific parameters regarding technical behavior can be set up to tune the service in behavior and performance.

Status

At the end of November 2005, a first alpha release of StoRM was released for a functionality test. Moreover, a web site was established describing the StoRM project [7] containing documentation and a user guide. The StoRM site supports user management, so it is possible download service RPMs previous to site subscription. The mechanism of registration was chosen to allow an efficient support of small groups of a selected certification team. Currently, an improved version is available to download. Develop and bug fixing are on-going as well as the improvement of the documentation. The current distribution of StoRM includes simple command line clients to test the service.

SECURITY MODEL

General aspects

There are several storage security issues: data integrity, confidentiality, authentication, non-repudiation and authorization are only some examples. Authorized access on storage resource, such as file and represents only the first step to provide a minimal security context. Anyway, the physical file on storage represents the final target of principal.

Secure access to a physical file in grid context involves many steps. Firstly, the principal has to own a valid identity defined within a virtual organization. She has to be authorized to query file catalogs in order to retrieve *logical file names (LFNs)* and *Storage URLs (SURLs)*. Both are terms clearly defined in the SRM standard. The knowledge of SURL allows pointing to the right storage and the SRM service that holds and manages it. The principal has to be authorized to access SRM service to obtain the TURL pointer.

Authorization policies are metadata bound with SURLs, live within catalogs affiliated with the VO domain. The LHC File Catalog (LFC) is one example, but other catalogs can also be used. Moreover, security subjects within authorization policies are in terms of grid identities.

Enforcing of permission on a physical file completes the entire security access scenario. Several security aspects belong to the virtual organization domain while others belongs to the site administrative domain. Principal identity definition, secure catalogs access and service authorization policies definition belong to the VO domain. Subject authentication, service authorization, and enforcing of access authorization policies on physical file belong to site domain. Every aspect must be taken into account at different levels and virtual organizations to build an effective secure context.

StoRM security model

The security model adopted by StoRM is qualitative, i.e. it permits or denies the access (intended to perform whatever action) to a particular resource (i.e. StoRM service, single file or directory and space). StoRM security acts at site level, in particular regarding authorization to storage The authorization model adopted by StoRM is addressed to allow local access to storage resource in secured fashion. StoRM utilizes a security mechanism in two levels as documented in the following paragraphs:

First level. Authorization is based on subject authentication. StoRM uses the Grid Security Infrastructure (GSI) for authentications. Some FQANs (short form for Fully Qualified Attribute Name) could be present within X.509 proxy certificates, as defined by the Virtual Organization Membership Service (VOMS) [8].

The first level engages the StoRM service itself as resource, and the possible actions on the StoRM resource are exactly the SRM function exposed. This first level relies on external authorization service. It is planned to integrate the Front End of the StoRM service with an external authorization service, like G-PBox [9].In this pattern G-PBox acts as policy decision point and StoRM acts as policy enforcement point. It is possible define authorization policy based on subject attribute (VOMS attributes are welcome) and usage metrics. In this way is possible to prevent unauthorized requests.

The information of an entity provided by VOMS attributes is not used actively within Front End layer to make authorization decisions about service utilization. Every attributes about principal is sent to Back End tier for checking authorization on file resources and management of namespace translation.

Second level. The StoRM security mechanism refers to an external metadata catalog as authorization source. Every authorization source is queried with a specified order, and the results are collected by StoRM with a deny override algorithm fashion. Authorization policies on file and space live on external catalogs, like the LHC File Catalog (LFC) or others. Authorization policies collected by StoRM are expressed in terms of grid identities and SURLs.

StoRM takes advantages of the security mechanism provided by the underlying file system to enforce permissions. StoRM sets up ACLs on the physical file to allow direct access from a worker node. The access control restriction to storage resources is based on mapping from grid identities to local user IDs, and on definition of ACLs. The grid identities mapping is accomplished by a call out to LCMAPS.

There are two way to define ACL on files. The *Just in Time (JiT)* model defines and enforce access control entries when access request is done while the *Ahead of Time (AoT)* model defines and enforces access control entries (ACEs) when authorization policies are defined. In the JiT model an ACL is removed when access is terminated or when life time of access policy is expired. In the AoT model the ACE is removed when authorization policies expires.

Simple access use case

The figure below shows a typical scenario of data access in a site with StoRM and a disk based storage system. The principal requires an SRM operation (e.g. SrmPrepareToGet), to obtain the file pinned and ready to be accessed. A validity check of the proxy certificate (held by the requester) occurs within the Front End. User attributes are used to delegate the external policy decision point (PDP, like a G-PBox) to validate the Principal attributes are sent to the Back End layer to complete request. Then, StoRM queries LCMAPS to obtain a local user corresponding to the grid identity of the requester. The physical name is derived by an SURL and user attributes (Virtual organization space).

The file system wrapper enforces permissions managing the ACL on the physical file using the JiT or the AoT model depending on user attributes and the configuration of StoRM. Finally, the user job can be executed into the worker node of a specific computing element. The end user application can perform a POSIX call to access the into/from the storage system.

TEST BED AND RESULTS

Test bed architecture

In this paragraph we present the hardware environment that has been used in the next test phase. We decided to use the hardware that will be used in production at our tier 1 center in Italy since the main idea is develop an high performance system that can be used in realistic activity phases of HEP experiments where critical performance on I/O bandwidth are required. In figure 2 we summarize the main parts of our hardware test bed.

The GPFS cluster employed as storage test bed is realized with the architecture described below. The disk storage (lower part of the figure) is composed by roughly 40 TBs. It is provided by 20 logical partitions of one dedicated StorageTEK FLexLine FLX680 disk array storage, aggregated by GPFS (version 2.3.0-10).

The FLX680 contains 96 blade units composed of 2 interconnected 250 GB drives. We created the 20 logical partitions using RAID 5 over 9 disks for the best performance/reliability ratio. The 2 Fibre Channel (FC) redundant controllers are connected using four 2 Gbps to our central Fabric Director FC Switch Brocade 24000 on 4 different hot-swappable switching modules for the best availability. As disk-server we decided to use 4 high performance Sun Microsystems SunFire V20Z servers with dual Opteron 2.6 GHz, 4 GB RAM and RAID 1 over the 2 x 74 GB SCSI system disks. The 4 servers are connected to the FC Switch through 2 Qlogic 2340 2 Gbps FC Host Bus Adapter (HBA) on 2 different switching modules. The

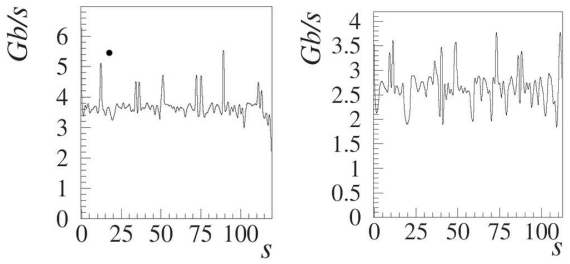

Figure 2: Test results: Read throughput and write throughput

Qlogic SANsurfer software tool has been used to fully implement fail-over paths over the disk storage to give the maximum availability in case of major failures of each component of the system (HBA, switching module, disk storage controllers), and a hardware problem on each component has been simulated during effective I/O on the logical partition for testing the fail-over and recover capability of the system. The 4 servers are connected through on-board Gigabit Ethernet interfaces to one central Gigabit switch with 4 uplink to our tier 1 central router giving a theoretical 4 Gbps Ethernet bandwidth to the disk server. The disk servers run GridFTP daemons as well to provide an external access to the file system.

Tests results

We have done extensive performance tests on the storage system described above. A short snapshot is presented here:

- **Write test** srmPrepareToPut() with implicit reserveSpace of 1 GB files. globus-url-copy from local source to the returned TURL. 80 simultaneous client processes.
- **Read test** srmPrepareToGet() followed by globus-url-copy from the returned TURL to a local file (1 GB files). 80 simultaneous client processes.

The two tests are meant to validate the functionality and robustness of the srmPrepareToPut() and srmPrepareToGet() functions provided by StoRM, as well as to measure the read and write throughput of the underlying GPFS file system. We have recorded on our testbed sustained read and write throughput of about 4 Gbps and 3 Gbps, respectively.

CONCLUSION

StoRM is designed with the aim of answering to a set of requests coming from various applications to allow direct POSIX file system access to files in local environment with high performance. In grid site modern parallel file systems like GPFS and Lustre on disk pool provide high scalability and reliability with great performance. A parallel file system together with StoRM as storage resource manager create a grid environment in which applications can perform access on shared storage resource without interact with any I/O service.

The StoRM server runs on top of any POSIX file system on Linux or UNIX, but can take advantage of the special features of the underlying file system (e.g. GPFS add-ons). StoRM allows data transfer functions, directory management functions and explicit space reservation with native POSIX access to files. The StoRM security model allows either grid access and local access comply with grid security requirements.

Our performance tests have show that StoRM provides a satisfactory throughput for demanding application scenarios. The development is on-going and early end-users are integrated in the software development cycle to improve both, the functionally as well as the performance of the system.

ACKNOWLEDGEMENTS

StoRM design and developing result from a collaboration between the INFN-CNAF in grid.IT Project, and the Abdus Salam ICTP in EGRID Project for Economics and Finance research.

Many thanks to INFN-CNAF Tier 1 for support and availability of storage infrastructure. Moreover, we acknowledge the people in INFN-Bari for their support.

REFERENCES

[1] The Large Hadron Collider. http://lcg.web.cern.ch/LCG, 2006.

[2] F. Schmuck, R. Haskin, GPFS: A Shared-Disk File System For Large Computing Clusters, FAST'02, Monterey, CA, 28-30 January 2002.

[3] Lustre filesystem, http://www.lustre.org/, 2006.

[4] ISO/IEC 9945-1:1996 (IEEE Std 1003.1, 1996 Edition), Information Technology - Portable Operating System Interface (POSIX).

[5] Storage Resource Management Working Group. http://sdm.lbl.gov/srm-wg, 2004.

[6] F. Donno, A. Ghiselli, L. Magnoni, R. Zappi. StoRM: Grid Middleware for Disk Resource Management. Chep 2004, Interlaken, Switzerland, 26 Sept -1 Oct 2004.

[7] StoRM project. http://grid-it.cnaf.infn.it/storm, 2006

[8] V. Ciaschini, A. Frohner. VOMS Credential Format. http://edgwp2.web.cern.ch/edg-wp2/security/voms/edg-voms-credential.pdf

[9] G-PBox, http://infnforge.cnaf.infn.it/projects/pbox/, 2006.