Understanding StoRM: from introduction to internals

Luca Magnoni and Riccardo Zappi INFN-CNAF

SRM2.2 deployment workshop - Edinburgh

13 November 2007

- Storage Resource Manager
- The StoRM service
- StoRM components and internals
- Deployment configuration
- Authorization and ACLs
- Conclusions.

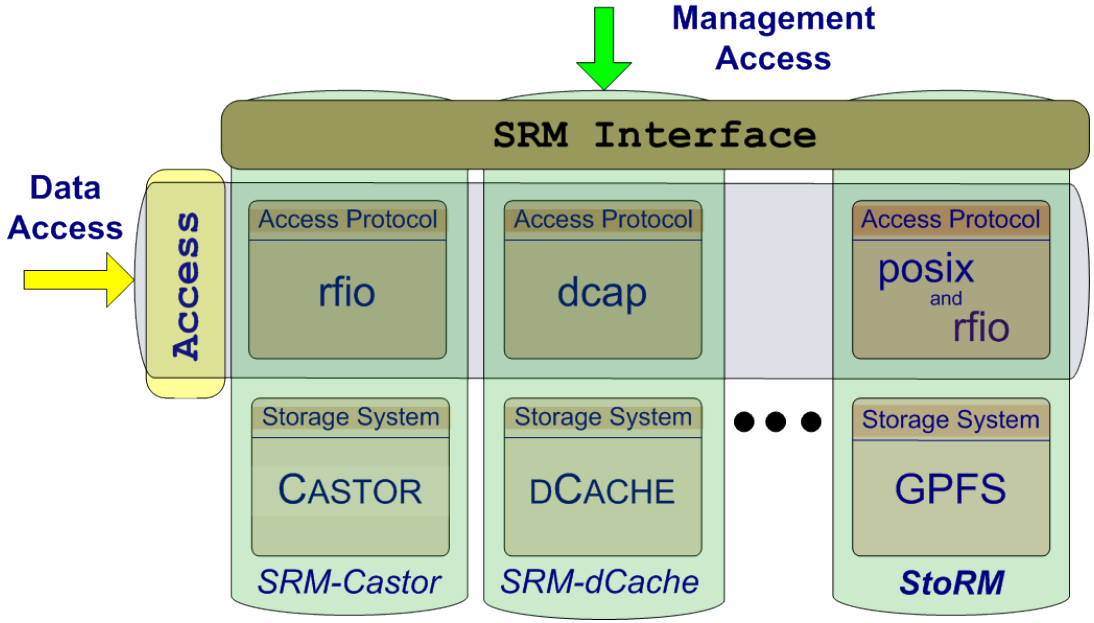
The SRM service

From bibliography:

- **Storage Resource Managers** are middleware services whose function is to provide dynamic space allocation and file management of shared storage components.
- Files are no longer permanent entities on the storage, but dynamic ones that can appear or disappear according to the user's specification.
- SRMs do not perform file transfers, but can invoke middleware components that perform this job (such as GridFTP)

The SRM interface

SRM services agree on a standard interface (**the SRM interface**) to hide storage characteristics and to allow interoperability.



SRM concepts

The SRM v2.2 is based on these concepts:

- **lifetime** of a file (volatile with a fixed lifetime, durable or permanent).
- **file pinning** (to ensure a file is not canceled during operation).
- **space pre-allocation** (to ensure the request space is available for the whole life of the application since the beginning).
- **storage classes** to identify different quality of storage resources.

SRM functionalities 1/2

The SRM functionalities can be grouped in the following categories:

- **Data management** provides capabilities to manage the data stored in a SRM, preparing envinronment for access operation (staging, etc.) and for receiving new data.
- **Space management** 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 for creating, moving and deleting directories and files.
- **Query functions**, this set of function is used to discover information on a specific SRM endpoint.

SRM functionalities 2/2

SRM service provides provides two classes of methods:

- **Asynchronous methods**, **non blocking call**. Return a token corresponding to the request, 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 methods**, **blocking call**. The control is returned to the client only when the request is completed. This is the case of directory management and space management functions.

SRM Storage Class

A Storage Class defines a classification of storage system in terms of quality of storage, expressed by:

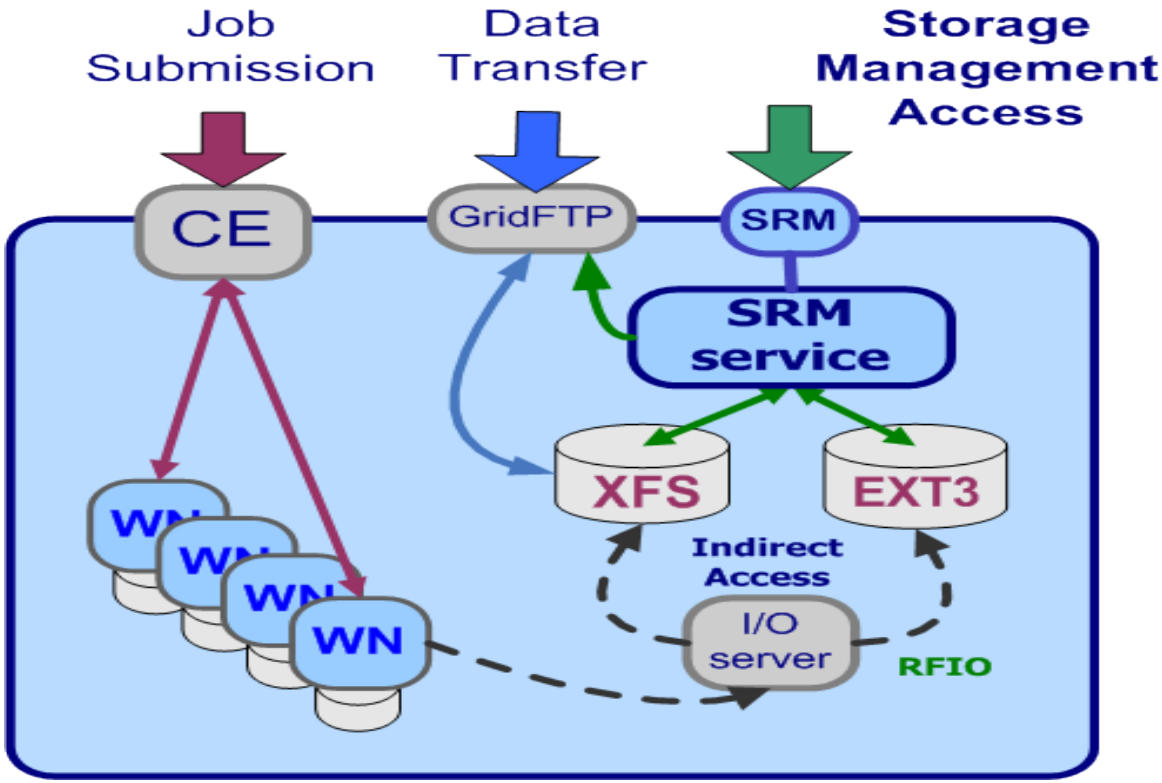
- **Retention Policy**, probability of data loss once stored. *[custodial, replica or output]*.
- **Access Latency**, the latency on data access operations. *[offline, nearline and online]*.

Asking for a certain storage class users can specify the desired storage system characteristics to store data. In WLCG has been decided mapping the quality of storage together with the combinations of storage devices.

- **Custodial-Nearline**, In WLCG **T1D0**.
- **Custodial-Online**, In WLCG **T1D1**.
- **Replica-Online**,In WLCG **T0D1**.

SRM role in a site

In a **Grid Storage Element**, the SRM service provides the functionality to manage file and spaces.



StoRM overview

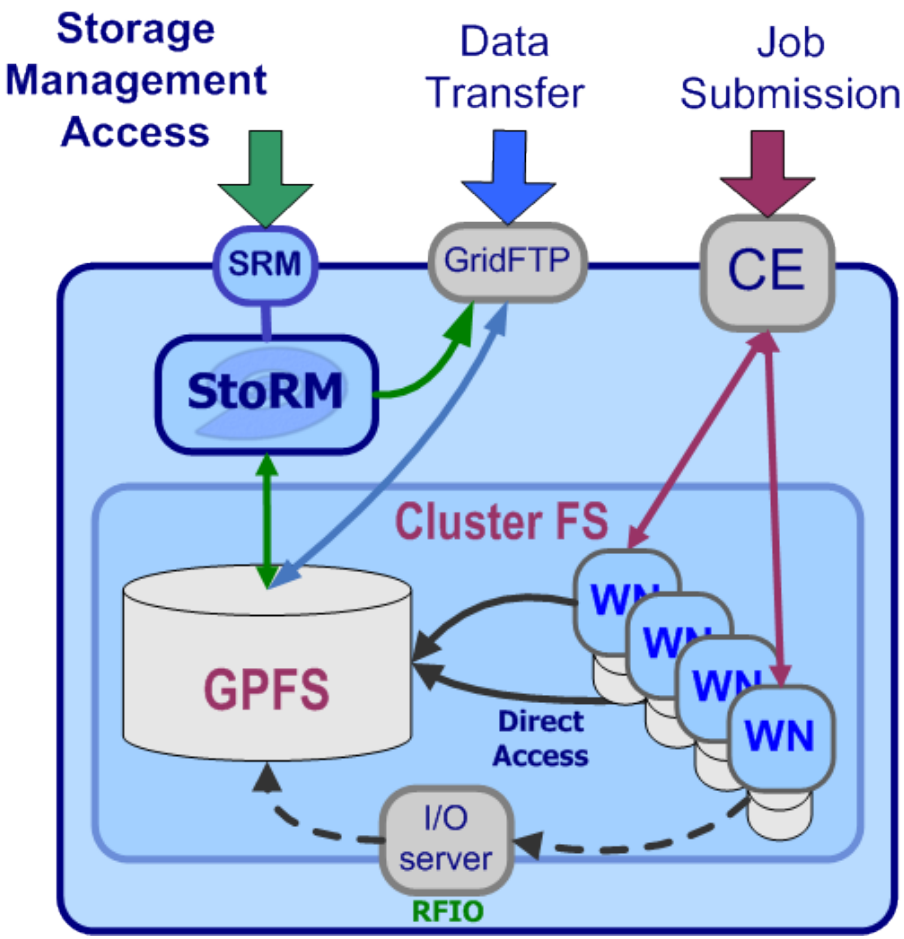
StoRM is a storage resource manager for disk based storage systems, implementing the SRM interface v2.2.

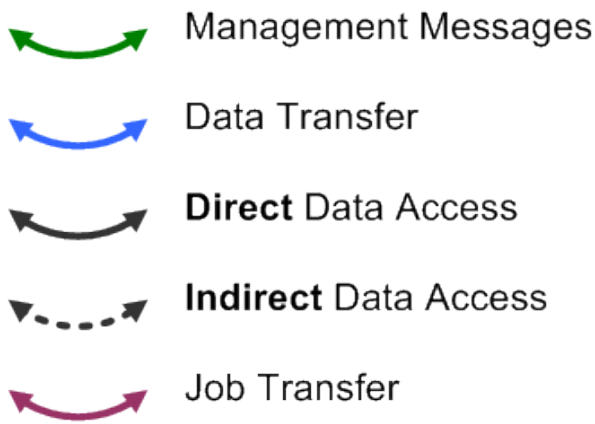
- It is designed to take advantage from **high performing cluster file system**, as GPFS from IBM and Lustre from ClusterInc., but it supports also every standard POSIX FS.
- It allows **direct access** (through the protocol *file://*) to the storage resource, as well as other standard grid protocol as *gsiftp* and *rfio*.
- Authentication and authorization are based on the **VOMS** credential.
- Permission enforcing are based on setting **physical ACLs** on files and directories.

StoRM and cluster fs

- StoRM takes advantage of aggregation functionalities provided by dedicated systems, such as parallel and cluster file systems.
- A cluster file system allows large numbers of disks attached to multiple storage servers to be configured as a single file system.
- A cluster file system provides:
  - Transparent parallel access to storage devices while maintaining standard UNIX file system semantics.
  - High-speed file access to applications executing on multiple nodes of a cluster.
  - High availability and fault tolerance.

StoRM role in a site





Current StoRM features 1/3

Current StoRM features:

- Support for different file system provided by a **driver** mechanism. Easy to expands.
- It's able to works on different file system type **at the same time**.
- Support for **file** protocol.
- As well as for other standard protocol as **rfio** and **gridftp**.
- Guaranteed dynamic space reservation (using underlying file system functionalities)

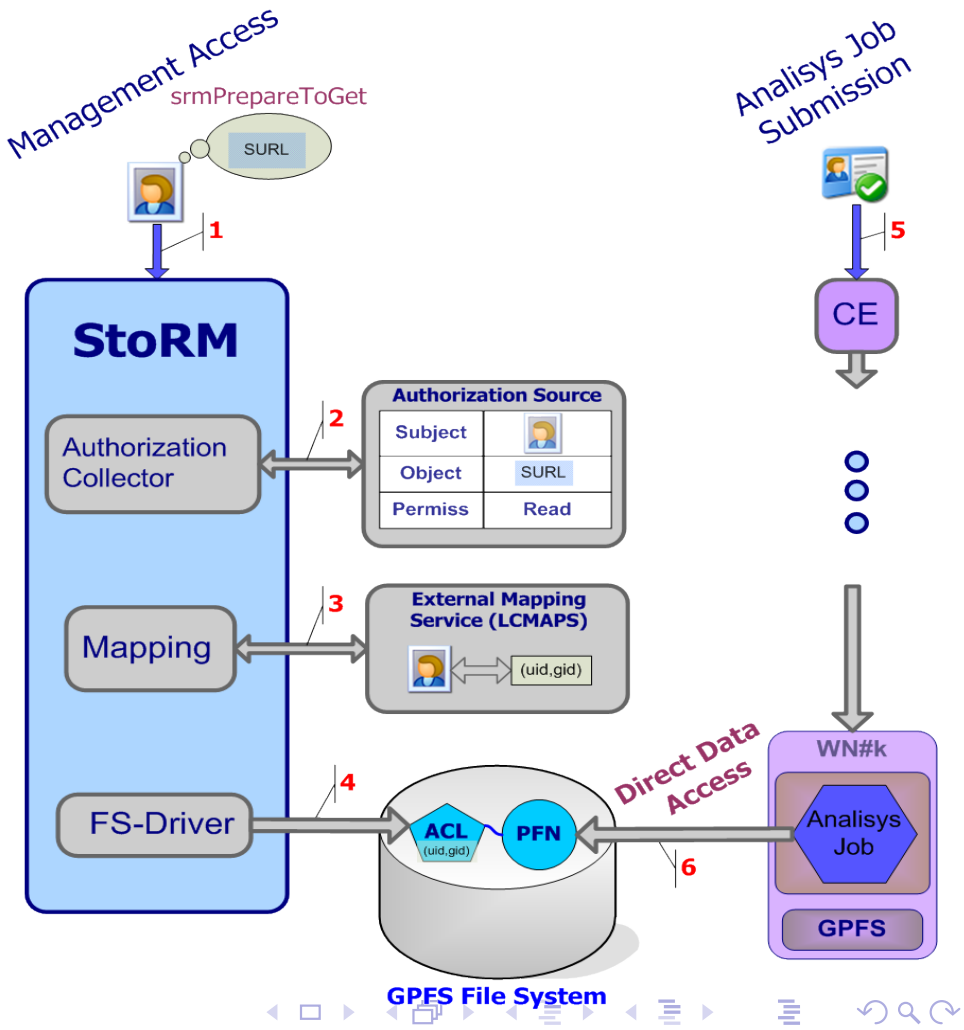
Current StoRM features 2/3

- **Storage Area (SA)**:
  - Storage Area can be defined editing the StoRM namespace configuration.
  - Each SA is addressed by path and by **Storage Area token**.
- **Quota**, relying on the underlying file system capabilities (as GPFS).
- Space and file garbage collector.

Current StoRM features 3/3

Layered security mechanism:

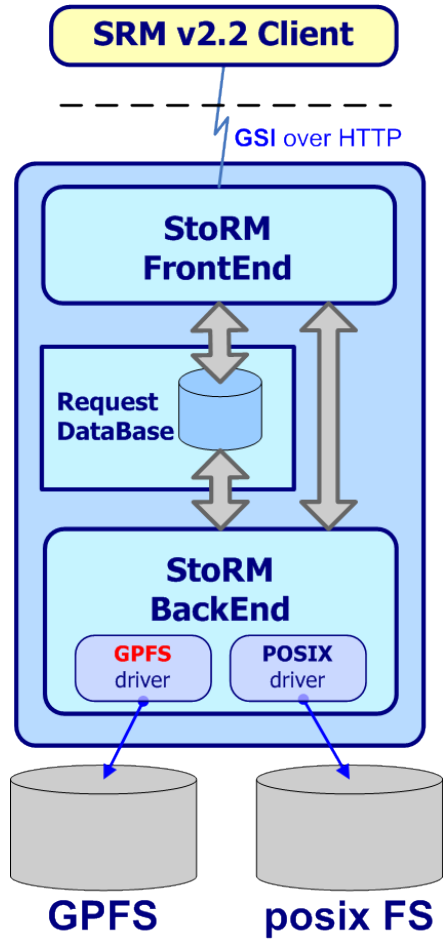
- **VOMS** compliant.
- **StoRM retrieves authorization information** from:
  - External services (as the LFC catalogue).
  - Local configuration.
- Enforcing of **file system ACL** on file and directory at user/group level.



StoRM architecture 1/2

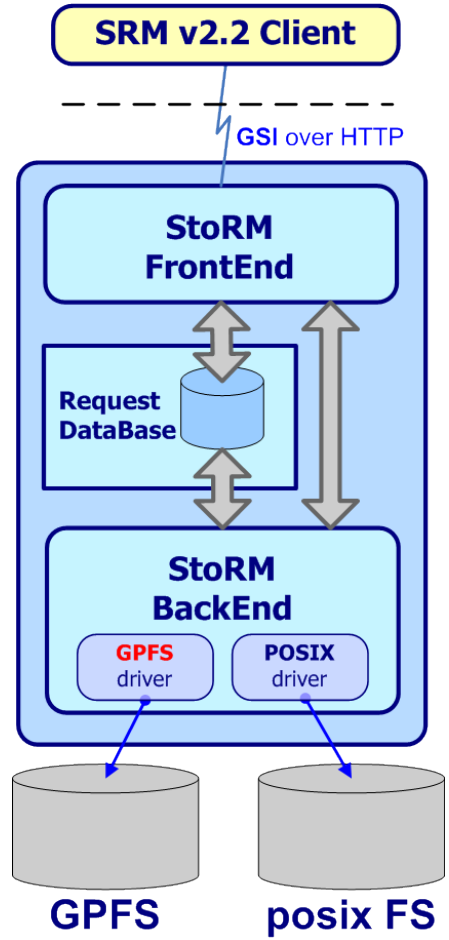
StoRM has a multilayer architecture.
The **Frontend (FE)** component:

- exposes the web service interface.
- manages user authentication.
- manages connection with clients.
- store asynchronous request into data base
- direct communication with Backend for synchronous request.
- retrieve request status.



StoRM architecture 2/2

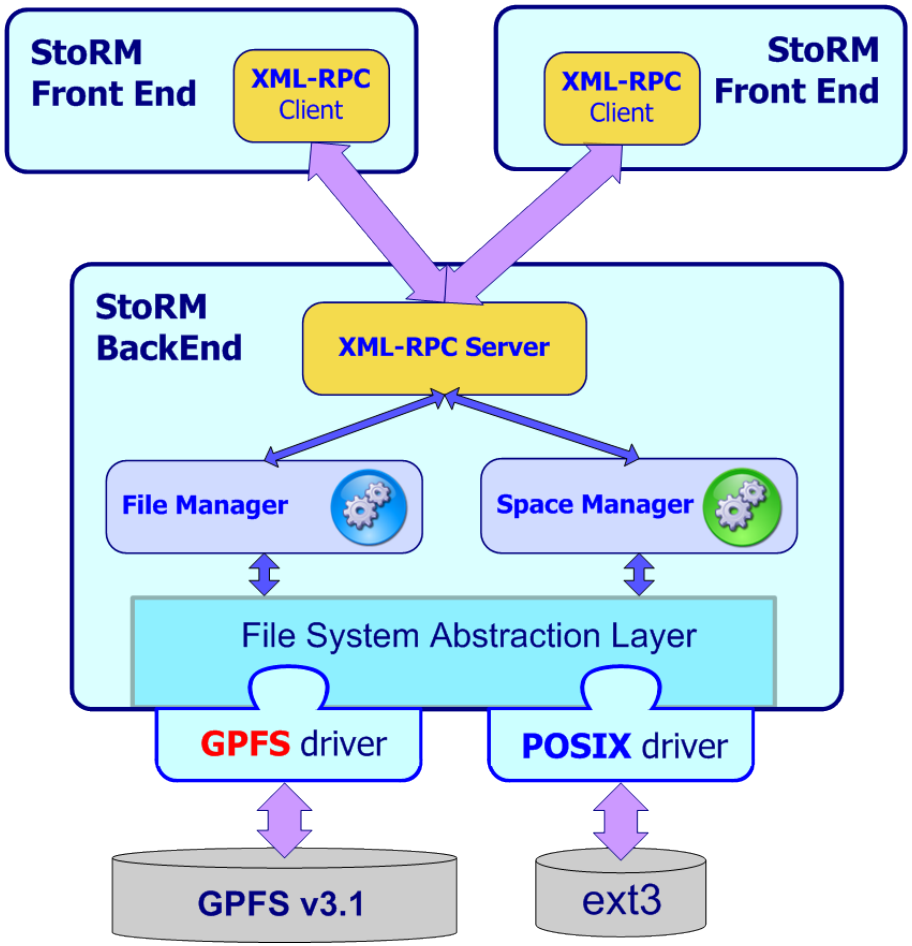
- **Database** is used to store SRM request data and the internal StoRM metadata.
- The **Backend (BE)** is the core of StoRM.
- it executes all synchronous and asynchronous SRM request
- manages user authorization
- enforces permissions
- interacts with other grid services
- provides support to file systems through a **driver mechanism**



The XMLRPC communication

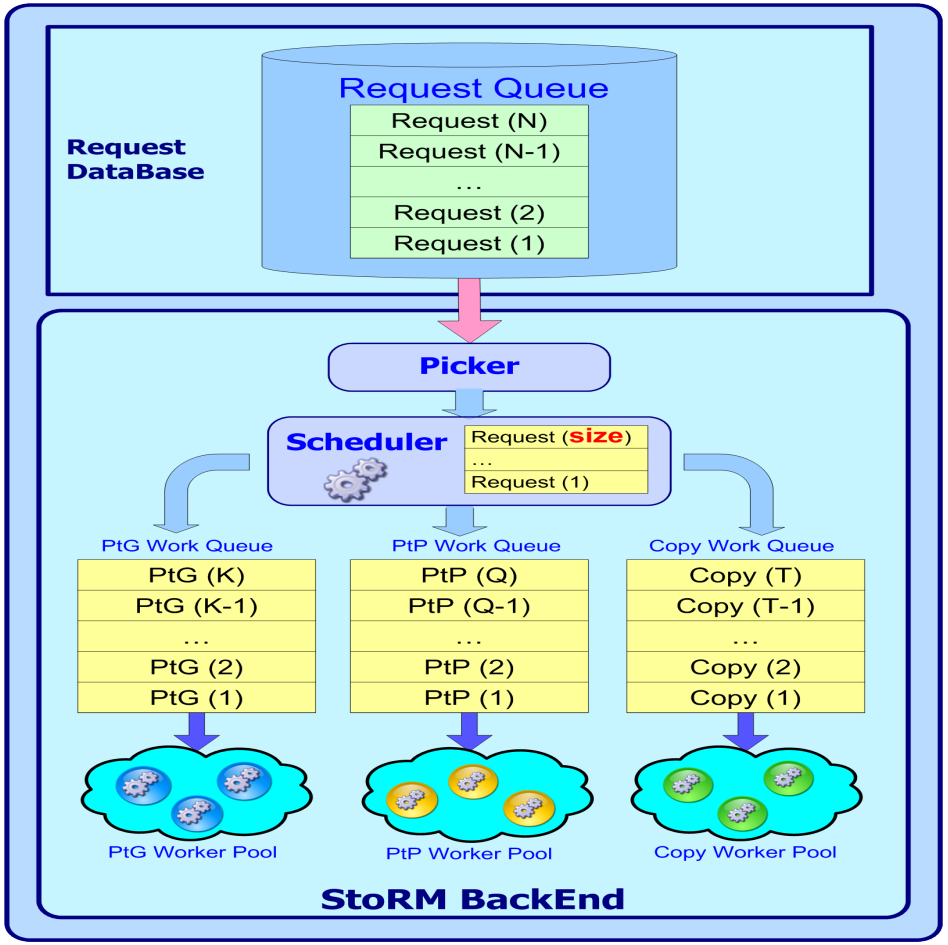
The Backend - Frontend communication in case of **synchronous SRM request** is based on XML-RPC communication.

- Standard.
- Host independent.
- Simpler than SOAP, appropriate for our purpose.
- Currently, one pool of thread for all SRM synchronous request.



Request queues and thread pools

- The Picker, get a set of pending request from the DB,with a polling mechanism.
- The Scheduler mange the set of request, forwarding them to the right pools.
- There are a queue and a thread pool for each type of asynch SRM request, PtP, PtG and Copy.



File system driver

StoRM is designed to be strongly independent from the underlying file system used. This is done using a driver mechanism for loadin support:

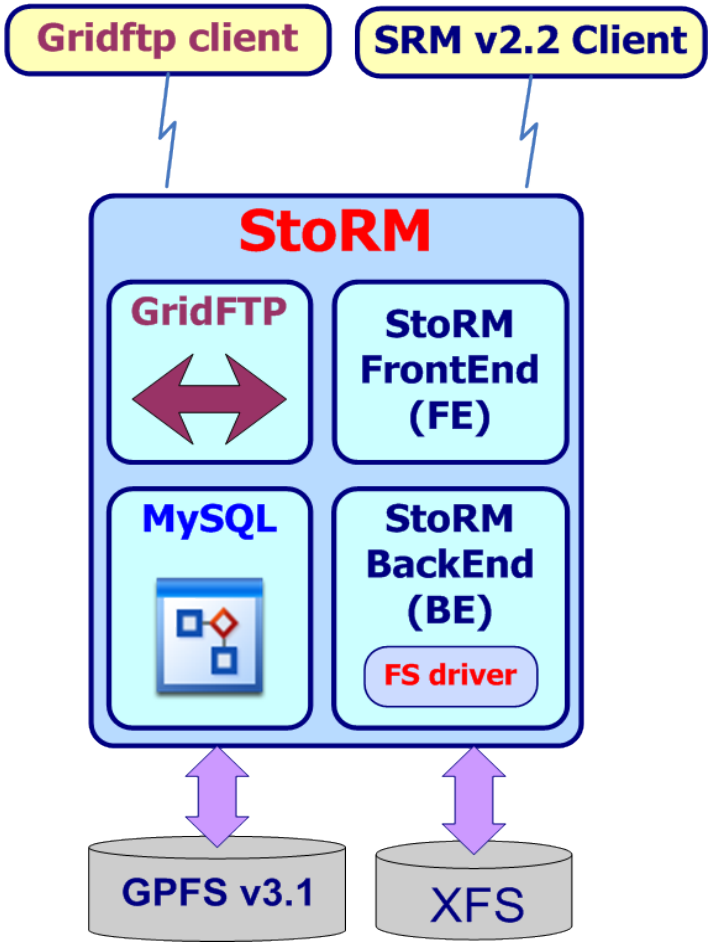
- Driver specified at configuration time.
- Driver has a internal interface that make StoRM logic independent from file system characteristics.
- support for new file system or new version (as the case of GPFS) can be easy added to StoRM only developing the specific driver.

Deployment on single site

All the component:

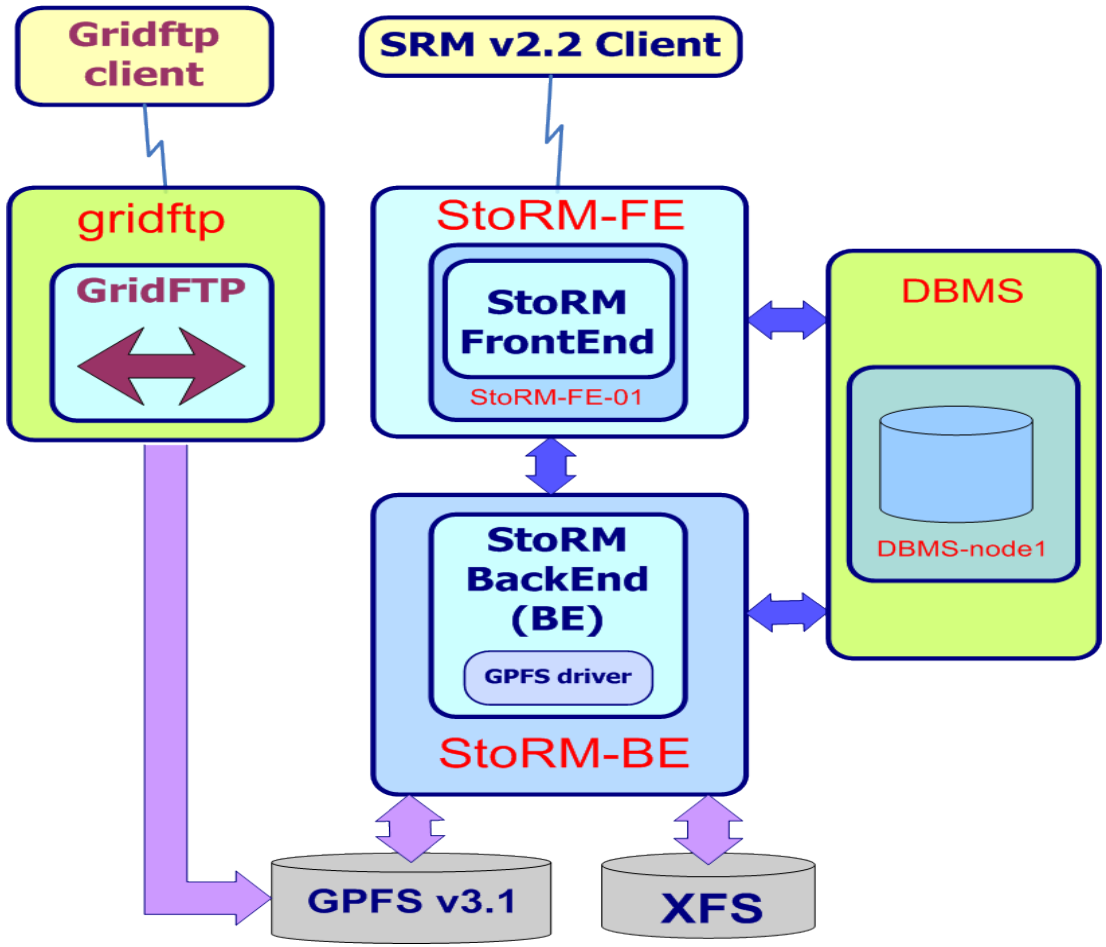
- **StoRM Frontend**
- **StoRM Backend**
- **MySQL**
- **GridFTP**

are **deployed on the same host**



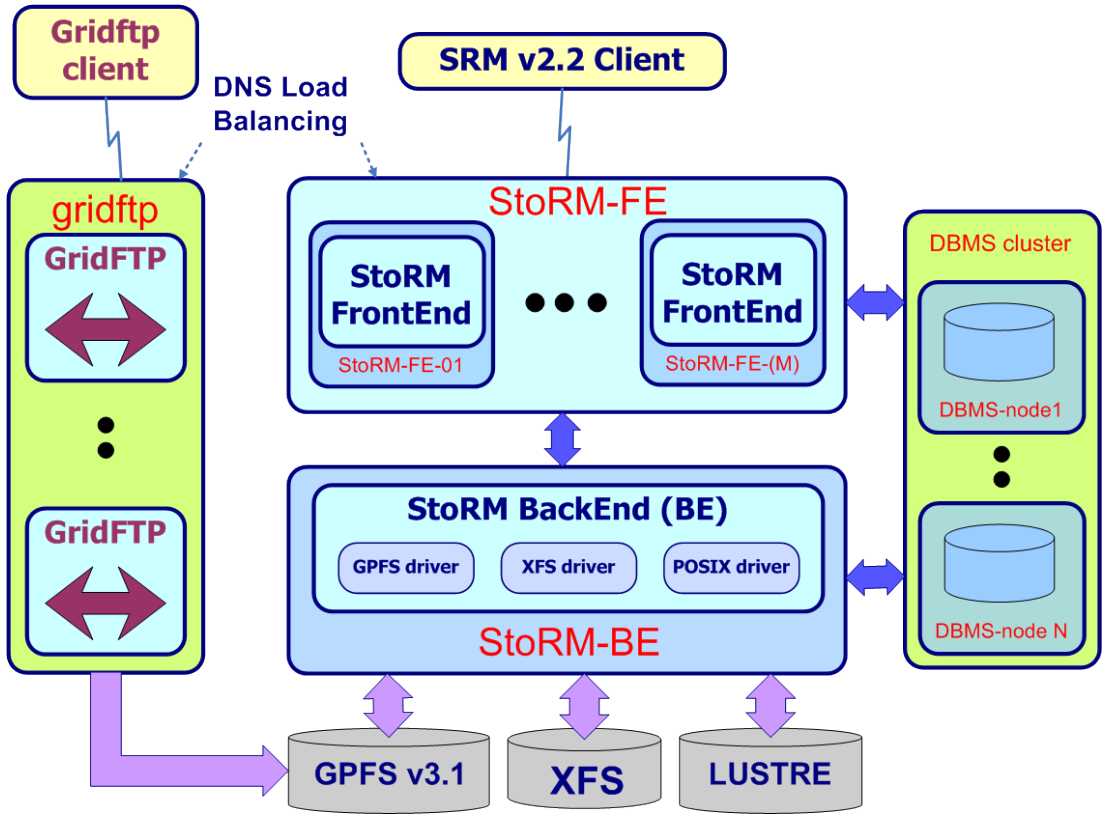
Deployment on different host

- StoRM architecture allow to **exchange information** over network .
- Each component can be deployed on a dedicated host.



Deployment on cluster

- StoRM supports **component replication**
- This allow to satisfy the **high availability and scalability** requirements.

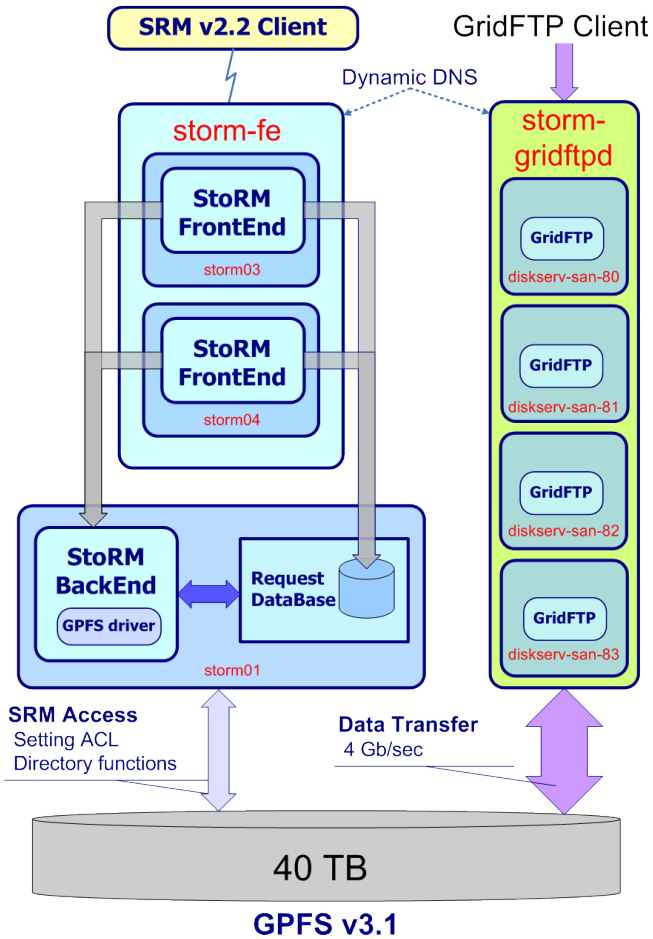


StoRM at CNAF Tier 1

The **CNAF T1** use StoRM as SRM v2.2 endpoint for the **T0D1** storage class.
We have set up a clustered configuration to make StoRM working in a production scale scenario:

- **2 FE** hosts (dual AMD Opteron 2.2 GHz, 4 GB).
- **1 BE** host (dual Intel Xeon 2.4 GHz, 2GB), shared with the MySQL DBMS.
- **40 TB** of disks available for experiment tests.
- **4 GPFS disk server** (dual Intel Xeon 1.6 GHz, 4 GB), with the *gridftpd* server in a dynamic DNS configuration.

StoRM at CNAF Tier 1



StoRM at CNAF Tier 1 - Some number on performances

Tests made by LHCb to validate their analysis scenario with StoRM and GPFS.

- Data transfer from CERN through CNAF via *gridftp*. Before and after each transfer a set of SRM requests are performed to StoRM.
- Average bandwidth: 240 MB/s.
- Bandwidth peak: 370MB/s.
- Stress test on SRM request on StoRM:
  - 100k handled.
  - At least 400k interactions (PtP files, statusPtP, Ls, Rm, etc.).
  - 600 parallel process that submit request to StoRM
  - Low failure rate ¡ 0-2%.
  - Execution rate of 40 request per seconds

This tests help to male optimization on database interactions.

StoRM status

StoRM general status:

- Latest stable release: v**1.3.18**
- Included in the INFN Grid release.
- Available by **YAIM**, **APT** - **rpm** or **Quattor**.
- File system driver currently available:
  - GPFS v2.3
  - GPFS v3.x (Improved ACL management using the new GPFS API)
  - XFS
  - generic PosixFS (as Lustre, Ext3). ACL enforcing made through Linux *setf/getfacl()* functionalities.
- StoRM web site: *http://storm.forge.cnaf.infn.it* for documentation, installation guide and client tutorial.

SRM v2.2 command line client

Together with StoRM a command line client for the SRM v 2.2 is available:

- Written in C++ using the GSOAP toolkit.
- Compatible with every SRM v 2.2 implementation.
- Provides the SRM syntax in a classic UNIX style.
- Example and tutorial available on the StoRM site.
- Usage example:
  clientSRM ptp -e httpg://ibm139.cnaf.infn.it:8444 -s srm://ibm139.cnaf.infn.it:8444/dteam/test111 -p

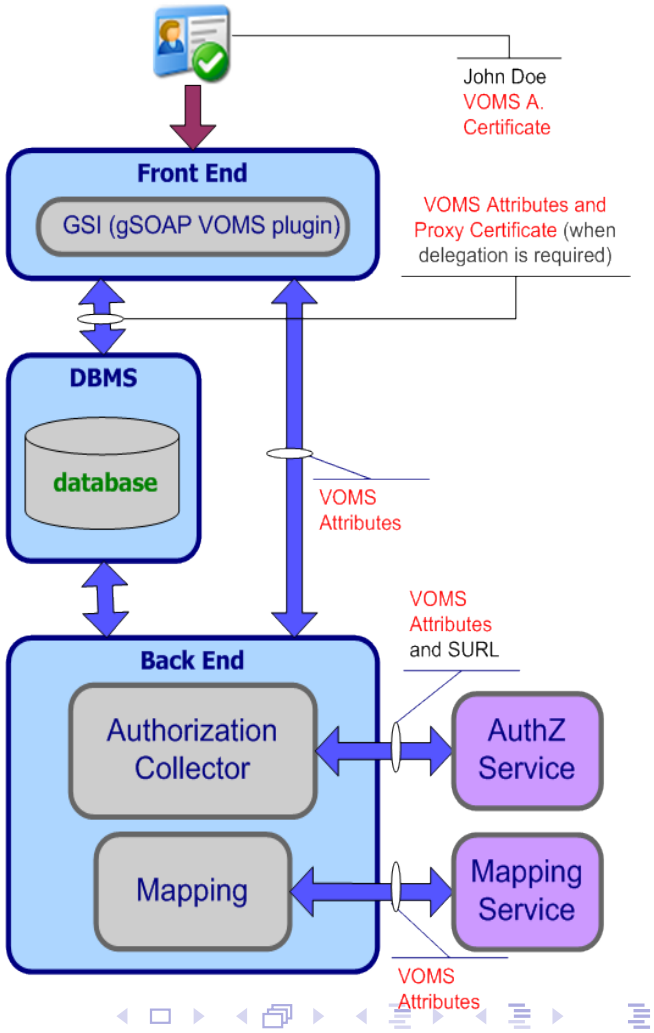
StoRM support

StoRM is integrated in the INFNGRID release from 3_0_0 update 35. For user support we provides:

- Installation and configuration guides and FAQ on StoRM site.
- Ticketing system.
- storm-user@cnaf.infn.it Mailing list.
- working on storm-support@cern.ch, that should be the official low level support for StoRM.

VOMS attributes in StoRM

- The VOMS attributes are retrieved from the user proxy by the StoRM Frontend through the CGSI_GSOAP plugin.
- The attributes are stored into dedicated tables in the StoRM catalog, to represents the user credential and the requestor identities.
- The attributes are evaluated for **authorization** operation by the StoRM Backend.



Approachable rules

- StoRM is able to represents the **authorization rule** for the SA, or in other words which VO can approach (or view) the SA, by a configuration features named **approachable rule**.
  - Approachable rules are defined per Storage Area and they are expressed by **regular expression in terms of FQAN**.
  - The SURLs within a request will be considered well formed if and only if the requestor is compliant with the specific App-rule.
  - The default value for app-rules is ".*", so all FQAN can approach every SA.

Authorization sources

StoRM is able to interact with external authorization service (named **authorization sources**) to perform the authorization decision.

- OGSA AuthZ WG is defining a standardized AuthZ Service Interface.
- Waiting for a standardized interface, we suppose that external service answer question like *"Can USER perform the ACTION on this RESOURCE"*.

We distinguish two kind of authorization sources:

- **Local**: based on configuration file or local information.
- **Global**: external service.

Local authorization sources

Local authorization source holds AuthZ policies for file or directories.

- Basic policy: Permit/Deny All .
- **Regular expression per path and FQANs** (permissions are equals on the same path) A simple XML file which defines AuthZ policies on the bases of directories (i.e. files within a directory will hold the same ACL).

Global authorization source

Global authorization source holds AuthZ policies valid for all storage resources accessible by VO-users.

- Currently StoRM can uses ECAR, a client for the **LFC catalogue**, to retrieve AuthZ information based on the ACLs on LFN.
- In the future version StoRM will have a Policy Enforcement Point (PEP) bound with external tools as the G-PBox. (G-PBox is an highly distributed policy management and evaluation framework).

ACL files and directories.

StoRM uses **ACL on file and directories** to enforce authorization decision.
StoRM interacts with the **LCMAPS** service to retrieve the **local uid** and **gid**. Two approach for ACL enforcing:

- **Just in Time (JiT)**: ACL enforced for **local user_id**, removed when the SrmPutDone/SrmReleaseFiles operation take place.
- **Ahead of Time (AoT)**: ACL enforced for **local group_id**. ACL will remain in place until the file or directory exists.

Conclusion 1/2

StoRM:

- leverages on cluster and parallel file system advantages in a Grid environment.
- support direct access on data.
- support Storage Area, Token and Description concepts.
- is heavily configurable, to satisfy the different site requirements
- StoRM is used at **CNAF T1** to provide T0D1 storage class.
- INFN-GRID integration will improve the user support and releasing process.
- The lack of SRM interface v1.1 in StoRM have penalized its diffusion. This should not be a problem anymore since the high level tools compliant with SRM v2.2 interface are finally available.

Conclusion 2/2

Regarding authorization and access permission StoRM provides:

- A layered and pluggable security mechanism.
- Approachable rule, then regular expression in terms of VO and DN to restrict file system access.
- Interaction with external authorization services (as the LFC catalogue).
- Enforcement of physical ACLs on file and directory for the local user identity corresponding to Grid credentials.

The resulting security framework address the strict requirement coming from the financial Grid community and allow the Grid application to perform a secure and efficient direct access to the storage resource.

StoRM

http://storm.forge.cnaf.infn.it

Antonia Ghiselli
Alberto Forti
Luca Magnoni
Riccardo Zappi

Ezio Corso