Status Report

Riccardo Zappi
INFN-CNAF, Bologna

CERN, 29 August 2006

Outline

1. **StoRM**
   - Origin and StoRM team
   - Quick overview of architecture
2. **Current Status**
   - Functionalities developed
3. **Timeline expected**
   - Functionalities incoming

StoRM Origin

Result of collaboration between:

- **INFN - Grid.IT Project :** to build a disk-based SRM service on top of high performance parallel file systems.

+

- **ICTP - EGRID Project :** to build a pilot national grid facility for research in Economics and Finance (www.egrid.it)

StoRM Development Team

INFN – CNAF

project coordinator : *A.Ghiselli*

developers : A.Forti, L.Magnoni, R.Zappi

eGRID - ICTP

coordinator :*A.Nobile, S.Cozzini*

developers : E.Corso, A.Messina, A.Terpin

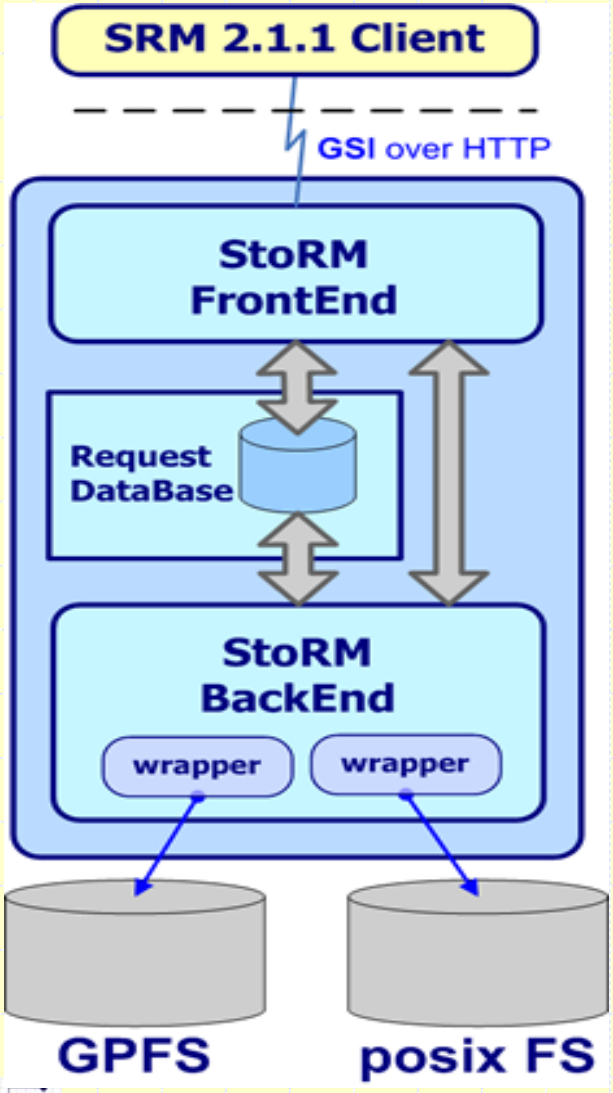
INFN-Bo

tester : V.Vagnoni (LHCb)

StoRM : General Features

- **StoRM is a disk based Storage Resource Manager which:**
  - **Production release implements SRM specification version 2.1.1**
  - **is migrating to the SRM version 2.2**
  - **is designed to support guaranteed space reservation.**
  - **supports direct access (native posix I/O calls).**
    - **Other access protocols remain available (e.g., rfio).**
  - **takes advantage of high performance Cluster File System with ACL support, such as GPFS.**
    - **Other posix file systems are supported (e.g., ext3)**
  - **Authentication and Authorization are based on VOMS certificates (LCMAPS).**

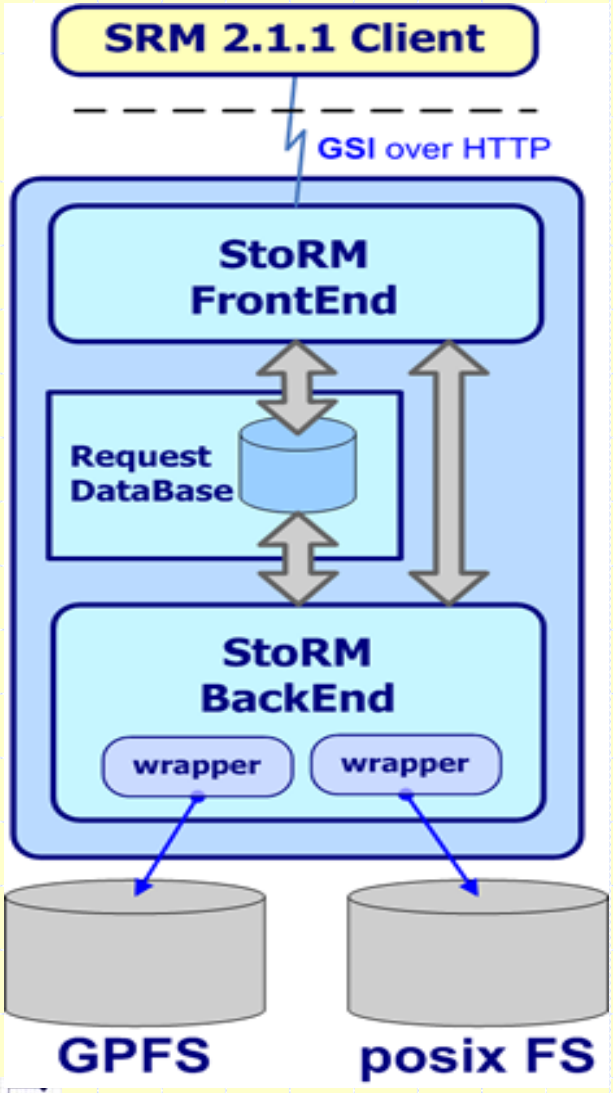
StoRM architecture



Front end (FE) has responsibilities of:

- expose a web service interface
- manage connection with authorized clients
- store asynchronous request into data base
- retrieve asynchronous request status
- co-operate with backend directly for synchronous call
- manage user authentication
- *co-operate with external authorization service to enforce security policy on service*

StoRM architecture



Data Base :

- Store SRM request and status
- Store application data

Back end (BE) has responsibilities of:

- execute all synchronous (active) action
- get asynchronous request from data base
- execute all asynchronous action
- bind with underlying file systems
- enforce authorization policy on files
- manage SRM file and space metadata

Migration Status to SRM v2.2 of StoRM

- **We started the migration from the released version of StoRM implementing SRM 2.1.1_modified (begin of July).**
- **The followed strategy is based on adapting the 2.1.1 functions already developed.**

Status Summary

Status	Functionality
Currently available	srmPing, srmGetProtocols srmMkdir, srmRmdir, srmRm, srmLs srmReserveSpace, srmGetSpaceMetadata srmStatusOf[*]
available in the next week / mid September	srmPrepareToGet, srmPrepareToPut, srmPutDone, srmCopy *(push mode)*
available by mid September	srmBringOnLine, srmReleaseFile, srmMv
Available in near future	srmReleaseSpace, srmChangeSpaceForFile, srmPurgeFromSpace

Status details (1/4)

Directory functionality

- **`srmMkdir`** : Done.
- **`srmRmDir`** : Done.
- **`srmRm`** : Done.
- **`srmLs`** : Done.
  - Only synchronous version is provided
  - Limit on returned entries configurable (admin. config)
- ***`srmMv`*** *:* Not provided yet **(available by the end of Sept. or before)**

Status details (2/4)

Data Transfer functionality

- **`srmPrepareToGet`**: Almost done. **(by mid Sept or before).**
- **`srmPrepareToPut`**: Almost done. **(by mid Sept or before).**
- **`srmPutDone`** : Almost done. **(by mid Sept or before).**
- **`srmCopy`** : Almost done. **(by mid Sept or before).**
  - Push mode only.
  - The management of **SRM_BUSY** is not completely tested.
- ***`srmBringOnLine`*** **:** Not provided yet **(available by the end of Sept.)**
  - We need more analysis for disk only solution (no "state" or "copies" of SURL within StoRM…)
- *"Other asynch func."* **:** Not provided yet **(no schedule)**

Status details (3/4)

Status functionality

- **`srmStatusOf`** [PtP/PtG/Copy/Bol] : Done.

Discovery functionality

- **`srmPing:`** Done.
  - otherInfo[] : ? … some example could be useful.
- **`srmGetProtocols:`** Done.
  - Return all protocols managed by StoRM instance (admin. config)
  - Protocol 'file:///' is supported

Status details (4/4)

Space Management functionality

- **`srmReserveSpace:`** Done.
  - Guaranteed space is managed only for GPFS through space allocation mechanism.
- **`srmGetSpaceMetadata:`** Done.
- ***`srmReleaseSpace`*** : Not provided yet **(no scheduled)**
- ***`srmChangeSpaceForFiles`*** : Not provided yet **(no scheduled)**
- ***`srmPurgeFromSpace`*** : Not provided yet **(no scheduled)**

Timeline summary

As soon as possible (by second week of Sept):

- StoRM with v2.2 endpoint available for interoperability tests.

By mid of September or before:

- All functionalities tagged as "almost done" will be provided. (`PtG`, `PtP`, `Copy`, `PutDone`).

By the end of September or before:

- `BringOnLine` and `ReleaseFiles`

Other functionalities:

- The schedule will follow the agreement ...

Last minute issues

- **retentionPolicyInfo** for srmReserveSpace:
  - is mandatory in SRM v2.2 spec. but
  - it is optional in WLCG usage agreement and within S-2 test suite.
- **Concurrent PrepareToPut on the same SURL**:
  - SRM V2.2 spec. not mentioned
  - WLCG agreement introduce SRM_FILE_BUSY for such use case

Which is the right interpretation?

Questions

?!

Links

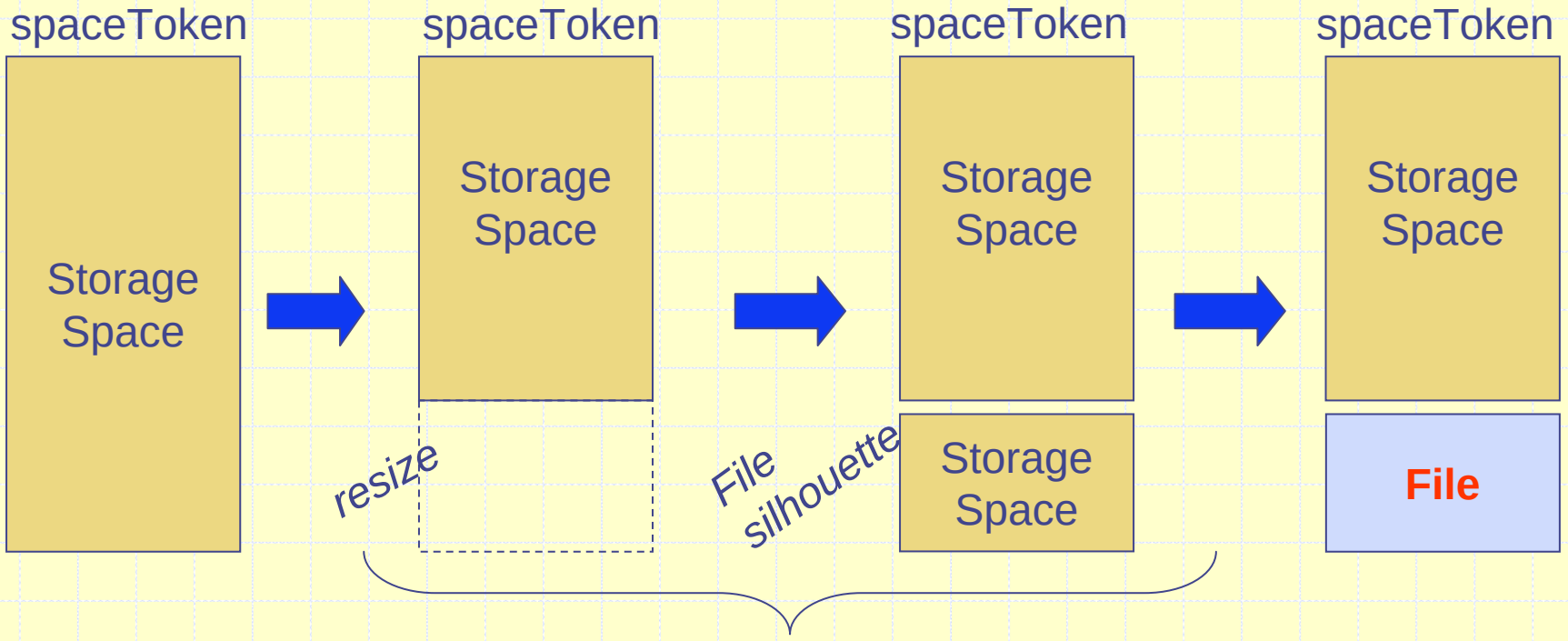
- **Hepix Spring meeting 2006** (3-7 April 2006) **:** Slides
- **CHEP'06** (13-17 February 2006) : Poster and Paper
- **Fourth EGEE Conference** (23-28 October 2005) : Slides

- **StoRM web sites :**
  - http://grid-it.cnaf.infn.it/storm
  - http://www.egrid.it/sw/storm/
  - http://forge.cnaf.infn.it/projects/storm/
- **Source code**
  - http://www.egrid.it/cvs/storm?root=cvs

Backup slides

Characteristics of Space Reservation in StoRM

Space guarantee is performed through a Space allocation (feature available only in StoRM on GPFS)



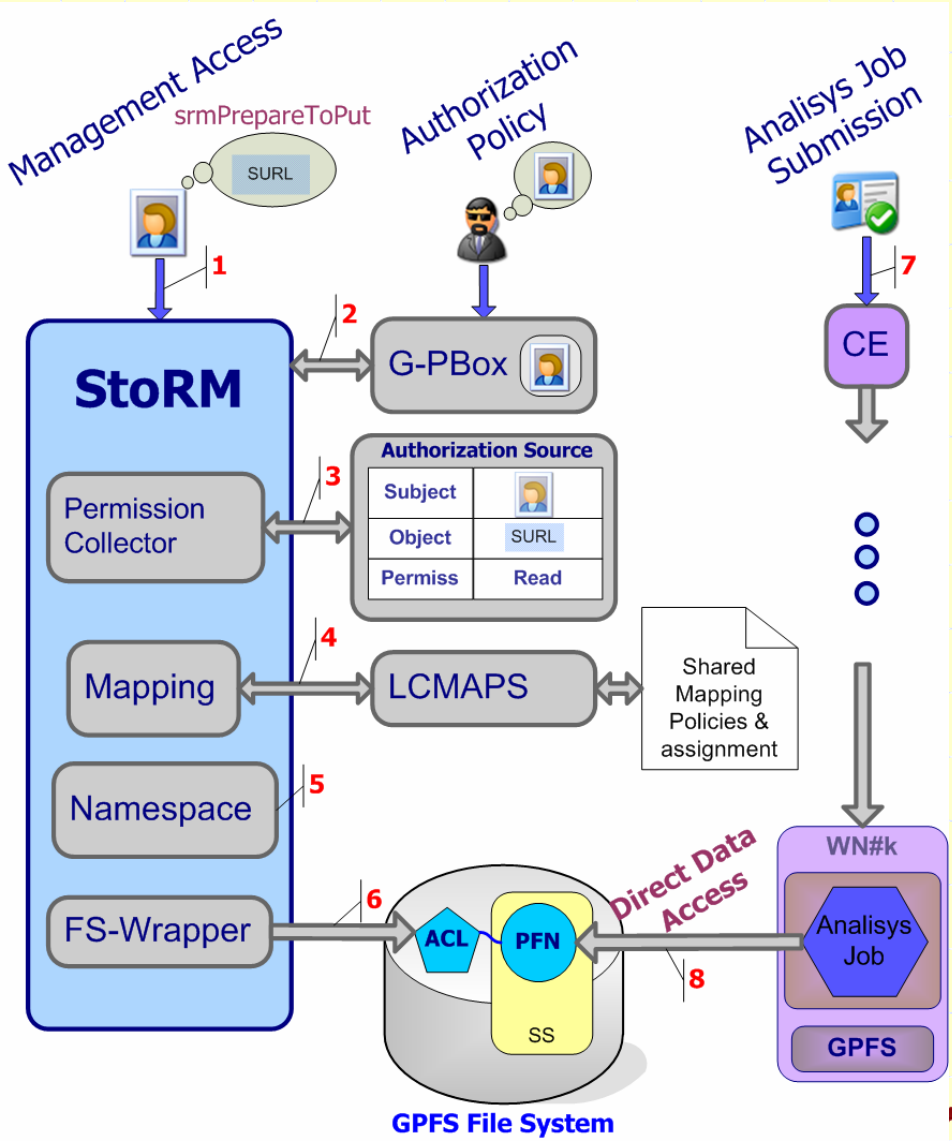
It is not an atomic operation.. but it's very fast. Inverse operation is possible..

Space Allocation in GPFS

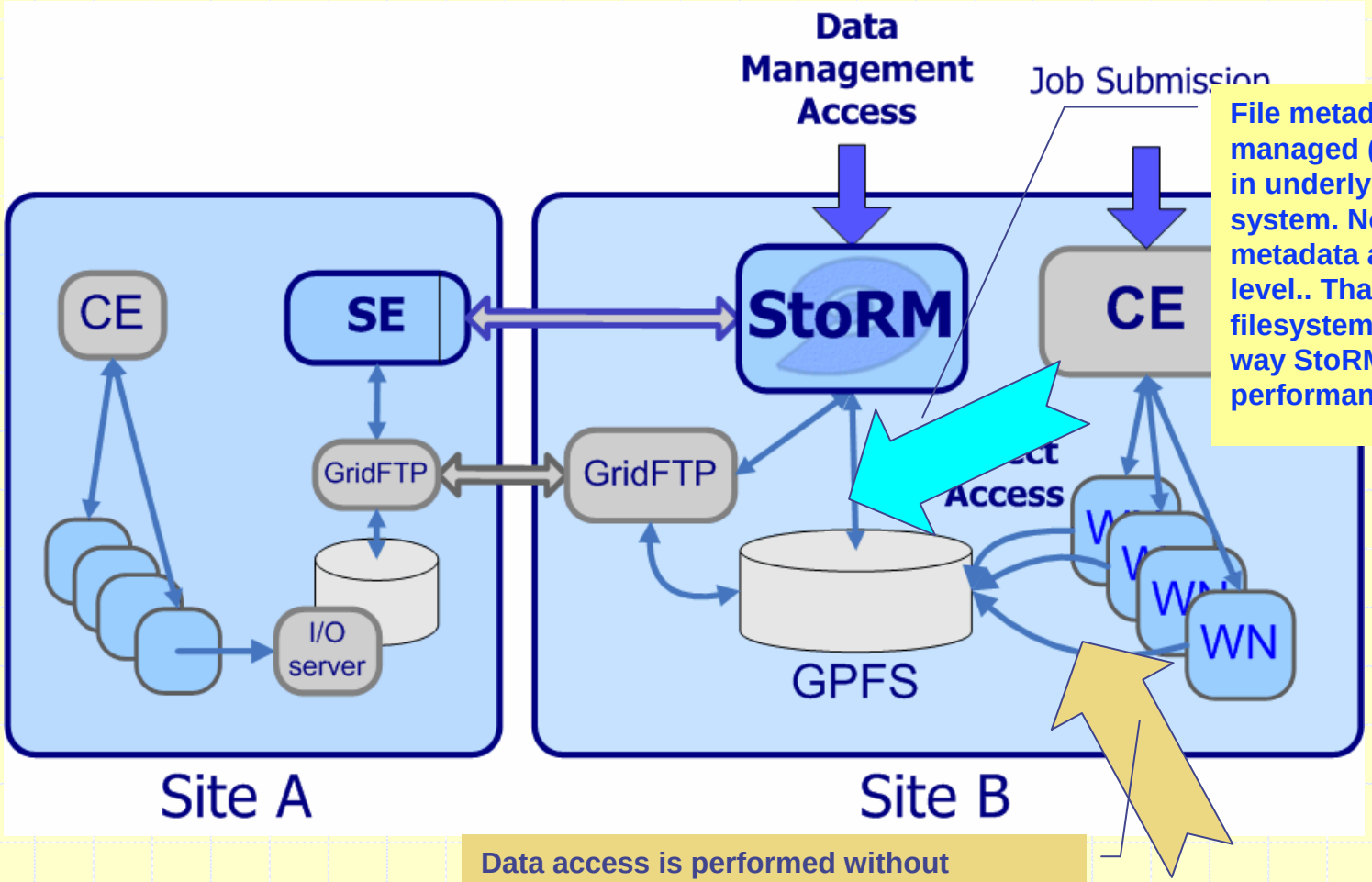
- **GPFS exposes a programming interface.**
- **gpfs_prealloc**( ) subroutine for storage space allocation
  - The **gpfs_prealloc( )** function is used to preallocate disk storage for a file that has already been opened, prior to writing data to the file.
  - It assigns a number of empty block to I-Node pointed by a parameter.
  - The preallocation of disk space for a file provides an efficient method for allocating storage without having to write any data.
  - This can result in faster I/O compared to a file which gains disk space incrementally as it grows.
  - Existing data in the file will not be modified. Reading any of the preallocated blocks will return zeroes.

Security aspects

1. User perform srmPrepareToPut
2. StoRM verifies if the principal holds a valid proxy certificate and delegates the external policy decision point to validate the request.
3. StoRM then queries the Authorization Sources to verify if the user can perform the specified operation on the SURL
4. StoRM queries LCMAPS to obtain local user account corresponding to the grid identity of the requestor
5. Physical file name derives by SURL and user attributes (Virtual organization name space)
6. The file system wrapper enforces permissions by setting a new ACL on the physical file.
7. The user job can be executed into the worker node
8. The application can perform a standard POSIX call to access the file into/from the storage system.



StoRM: SE model



File metadata are managed (and stored) in underlying file system. No replica of metadata at application level.. That is a filesystem job! In this way StoRM gain in performance.

Data access is performed without interacting with an external service, with great performance improvemnet. (by POSIX call) . Moreover, in case of Parallel File System, StoRM permits to make the most of the provided performance!!

StoRM configurability

Service configuration (XML based)

Per VO configuration

- Default values (Space size, lifetime, ...)
- Namespace (relative path names)
- Protocols allowed (and other capabilities)
- Authorization model adopted
- Drivers for wrapping underlying Filesystem

Service customization (Property file)

- Service properties (port, tuning parameters, ...)
  - Properties file that overwrite default service configuration