StoRM configuration and deployment

Luca Magnoni INFN-CNAF

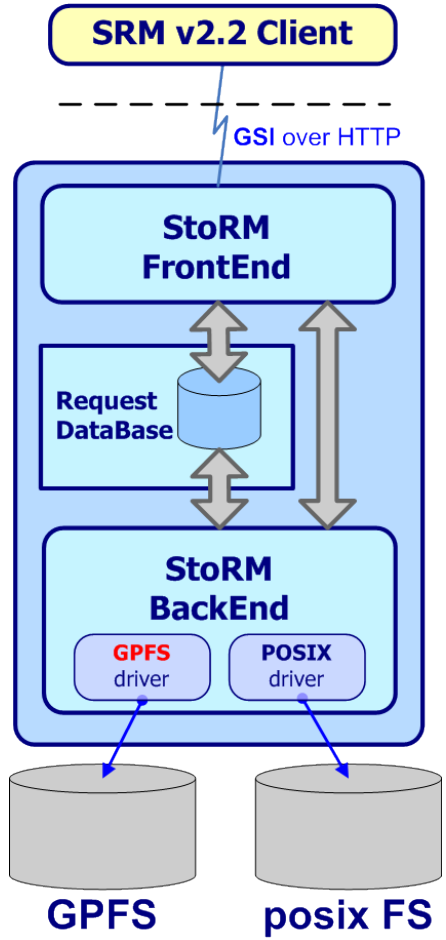
Pre-GDB Grid Storage Services

11 November 2008

- Architecture and internals
- Configuration
- Log analysis
- Deployment and layout

StoRM architecture

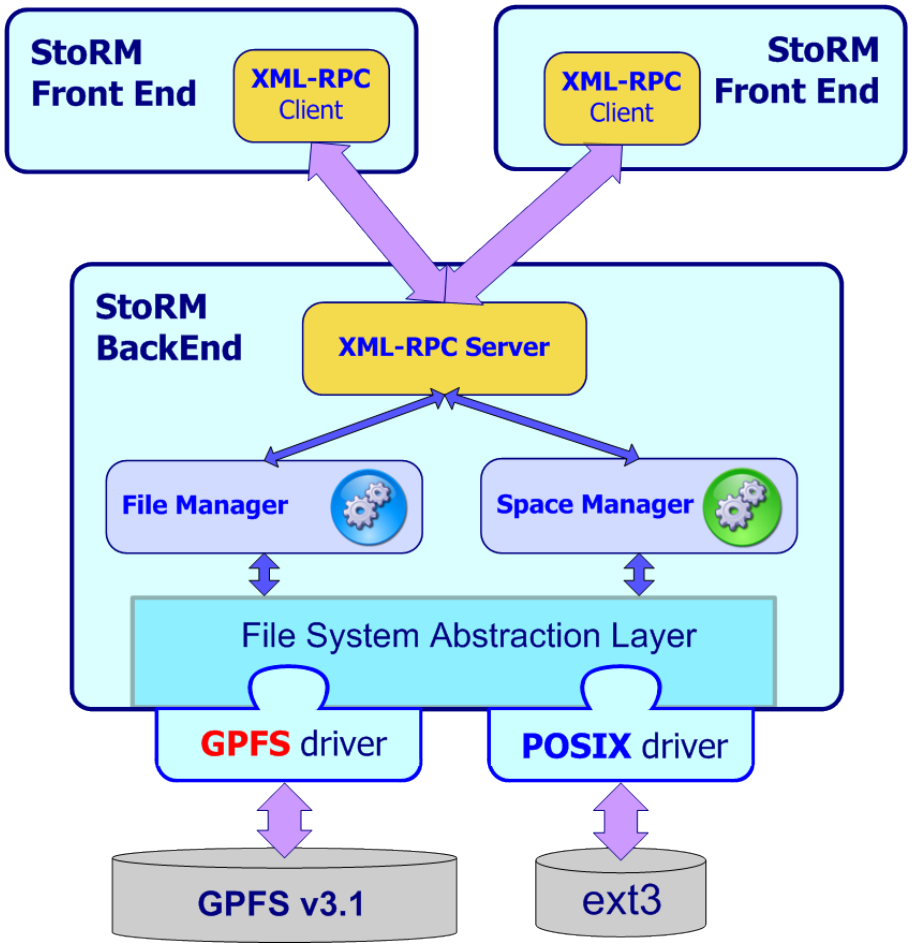
- Two main **stateless** components:
- The **Front End (FE)** exposes the service interface and manages user authentication
- Database is used to store SRM request data and the internal StoRM metadata
- The **Back End (BE)** is the core of StoRM, it executes all synchronous and asynchronous SRM request and interacts with file systems and with other Grid services.



The synch way: XMLRPC communication

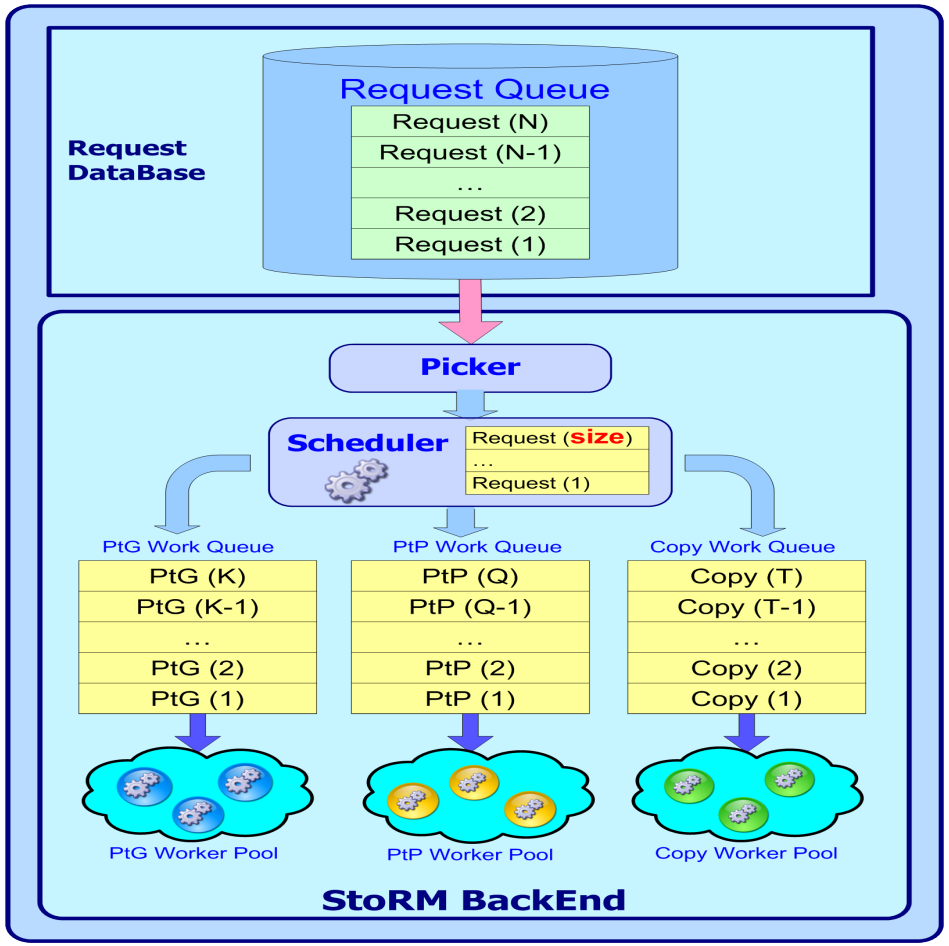
The Back End - Front End communication in case of **synchronous SRM request** is based on XML-RPC communication.

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



The asynch way: Request queues and thread pools

- The Picker, get a set of pending request from the DB,with a polling mechanism.
- The Scheduler manage 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 indipendent from the underlying file system using a **driver mechanism** for loading support:

- Driver specified at configuration time.
- Driver has a internal interface that make StoRM logic indipendent 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.

Services configuration

Introduction to the following service configuration:

- **StoRM Frontend**
- **StoRM Backend**
- **LCMAPS plugin**
- **GridFTP**

Frontend configuration

- The Frontend installation directory is */opt/srmv2storm/*.
- In StoRM v1.4: /opt/storm/frontend
- The Frontend configuration is based on:
  - file */opt/srmv2storm/etc/sysconfig/srmv2storm.nsconfig*
  - contains the information for database connection:
    db_user/db_password
    @frontend_host
- There are also command line options for advanced tuning that can be used in:
  - /opt/srmv2storm/etc/init.d/srmv2storm
- In StoRM v1.4 options available also through a new configuration file

Backend configuration

- The Backend installation directory is */opt/storm/*.
- In StoRM v1.4: /opt/storm/backend
- The Backend configuration is based on two files in */opt/storm/etc*:
  - **storm.properties**: service configuration.
  - **namespace.xml**: namespace configuration.

LCMAPS and GridFTP configuration

- LCMAPS
  - LCMAPS provides **the mapping between Grid credential and local users**.
  - StoRM has an ad-hoc configuration in:
    - /opt/storm/etc/lcmaps.db
- GridFTP
  - VDT version 1.6.1
  - Globus GridFTP server v.2.3
  - Service configuration:
    - Command line parameter/shell variables check in **/etc/init.d/globus-gridftp**.
    - LCAS/LCMAPS/**opt/glite/etc/lcmaps/lcas-lcmaps.db**.

Configuring the service: storm.properties

The **storm.properties** is a standard JAVA properties file, a set of **key=value** entries. An example:

- *storm.service.hostname=storm01.cr.cnaf.infn.it*
- *storm.service.port=8444*
- *storm.service.inQueryForm=true*
- *storm.service.endpoint=/srm/managerv2*
- *persistence.db.host = localhost*
- *persistence.db.username = storm*
- *persistence.db.passwd = storm*
- *...*

Properties

StoRM properties:

- **SRM service details** StoRM machine names and IPs, port, service endpoint path.
- **Front End and Back End communication** XMLRPC details, number of threads.
- **Database details** DB hostname, username, pwd.
- **Multithread SRM requests scheduler** Number of thread, queues size.
- **Requests garbage collector** Polling time, expiration time.

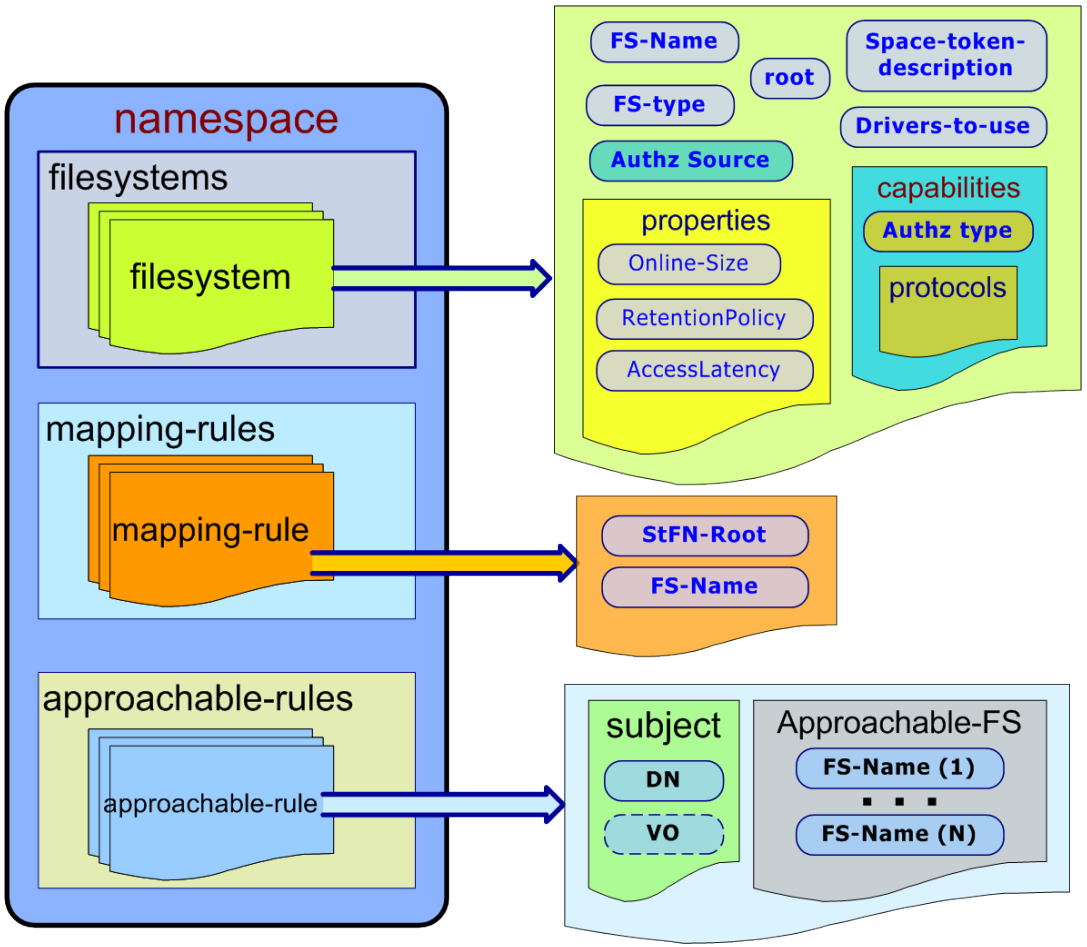
Configuring the service: namespace.xml

Mapping The mapping functionality is the process of retrieving or building the transport URL (TURL) of a file addressed by a **Site URL (SURL)** and a **grid user credential**.

In StoRM, the mapping functionality is provided by the **namespace component (NS)**.

- The Namespace component (NS) works without a database.
- The Namespace component is based on an XML configuration.
- It relies on the physical storage structure.

Namespace configuration elements



High view of the namespace main elements.

1. File system
2. Mapping rule
3. Approachable rule

Frontend Log

- The Frontend log:
  - **/opt/srmv2storm/var/log/srmv2storm.log**.

  reports information on each request received by the SRM.
- The log verbosity is a command line option.
  - **Warning, In case of authentication error at GSOAP level, nothing is logged!**
  - In case of problem, enable the **CGSI_GSOAP tracefile**, as reported in the FAQ section of StoRM website.

StoRM Backend Log

The Backend log are in :

- **/opt/storm/var/log/storm-backend.[log-stdout-stderr]**: SRM requests log file.
- **/opt/storm/var/log/heartbeat.log**: report on StoRM status and SRM requests summary.

storm-backend.log

In the file *storm-backend.log* you can find all log information

- Log example: **2007-11-09 19:21:04,565 INFO [pool-1-thread-5] Log message...**
- The Backend uses *log4j* to structure the log operations.
  - Different level of logging: **DEBUG, INFO, WARN, ERROR, FATAL**.
  - Configured by the file *log4j.properties*.
  - **DEBUG** level can help a lot in case of error, but costs in term of performance and log cleanness.

The *storm-backend.stderr* file contains standard error output.

hearthbeat.log

The file *hearthbeat.log* contains a summary of the SRM operations performed by StoRM. The log are structured:

- **[2008-06-26 16:03:32,610]** : Logging date
- **[...171 lifetime=2:52.00]**: Lifetime
- **Heap Free:34471264**: JVM Heap free
- **SYNCH [165]** : Synch requests executed from **last beat**.
- **ASynch [PTG:806 PTP:2159]**: Asynch requests executed from **START TIME**.
- **Last: [PTG=1 OK=1 M.Dur.=229]**: N. of PtG managed in last beat with **n. of success** and **avg. msec**.
- **[PTP=16 OK=16 M.Dur.=283]** : N. of PtP managed in last beat with **n. of success** and **avg. msec**.

LCMAPS and GridFTP Log

- GridFTP server:
  - Log files:
    - **/var/log/globus-gridftp.log**.
    - **/var/log/gridftp-session.log**
  - The default level is **really poor of information** and **logs only transfers executed with success**.
  - Command line option/shell variables can be used to enable log details in:
    - */etc/init.d/globus-gridftp*
  - Verbose logging (-d all).
- LCMAPS
  - Log information on mapping operation here:
    - **/opt/storm/var/log/lcmaps.log**.
  - Logs are really hard to read, but can became useful in case of problem.

StoRM YAIM profiles

Two node-type profiles:

- **ig_SE_storm_backend**: StoRM backend component
- **ig_SE_storm_frontend**: StoRM frontend component

and the GridFTP profile:

- **ig_GRIDFTP** : GridFTP server

Profiles can be combined for a **all-in-one** installation or can be used alone for a **cluster or distributed** configuration.

Variables categories

The YAIM configuration manges:

- Service properties.
- Storage Area configuration.
- Access and transfer protocols.
- Information published.

Variable name	Type	Description	>= ig-yaim version
STORM_<SA>_ACCESSPOINT	O	Path exposed by the SRM into the SURL. Default value: /<sa>.	4.0.2-9
STORM_<SA>_ACLMODE	O	ACL enforcing mechanism. Available values: [aot|jit] (use aot for WLCG experiments). Default value: aot.	4.0.2-9
STORM_<SA>_DEVICE	O	GPFS device on which the quota is enabled. It is mandatory if STORM_<SA>_QUOTA variable is set. Default value: $STORM_<SA>_ROOT.	4.0.2-9
STORM_<SA>_FILESET	O	GPFS FileSet on which the quota is enabled. It is mandatory if STORM_<SA>_QUOTA variable is set. Default value: <SA>.	4.0.2-9
STORM_<SA>_FILE_SUPPORT STORM_<SA>_GRIDFTP_SUPPORT STORM_<SA>_RFIO_SUPPORT STORM_<SA>_ROOT_SUPPORT	O	Enable the corresponding protocol. Default value: $STORM_INFO_<PROTOCOL>_SUPPORT.	4.0.2-9
STORM_<SA>_FSTYPE	O	File System Type. Available values: [posixfs|xfs|gpfs]. Default value: $STORM_FSTYPE.	4.0.2-9
STORM_<SA>_GRIDFTP	O	Gridftp server for the Storage Area. If not specified, the STORM_HOST variable is used. Default value: $STORM_HOST.	4.0.2-9
STORM_<SA>_QUOTA	O	Enables the quota management for the Storage Area and it works only on GPFS filesystem.	4.0.2-9
STORM_<SA>_RFIO	O	RFIO server for the Storage Area. If not specified, the STORM_HOST variable is used.	4.0.2-9
STORM_<SA>_ROOT	O	Physical storage path for the VO. Default value: $STORM_DEFAULT_ROOT/<sa>	4.0.2-9
STORM_<SA>_STORAGECLASS	O	Storage class type. Set different from the default value T0D1.	4.0.2-9
STORM_<SA>_TOKEN	O	Storage Area token. Default value: <SA>_TOKEN	4.0.2-9

SE StoRM Frontend

Configuration file

Specific variables are in:

- /opt/glite/yaim/examples/siteinfo/services/ig-se_storm_frontend

Please copy and edit that file in your <confdir>/services directory (have a look to "*YAIM configuration files*").

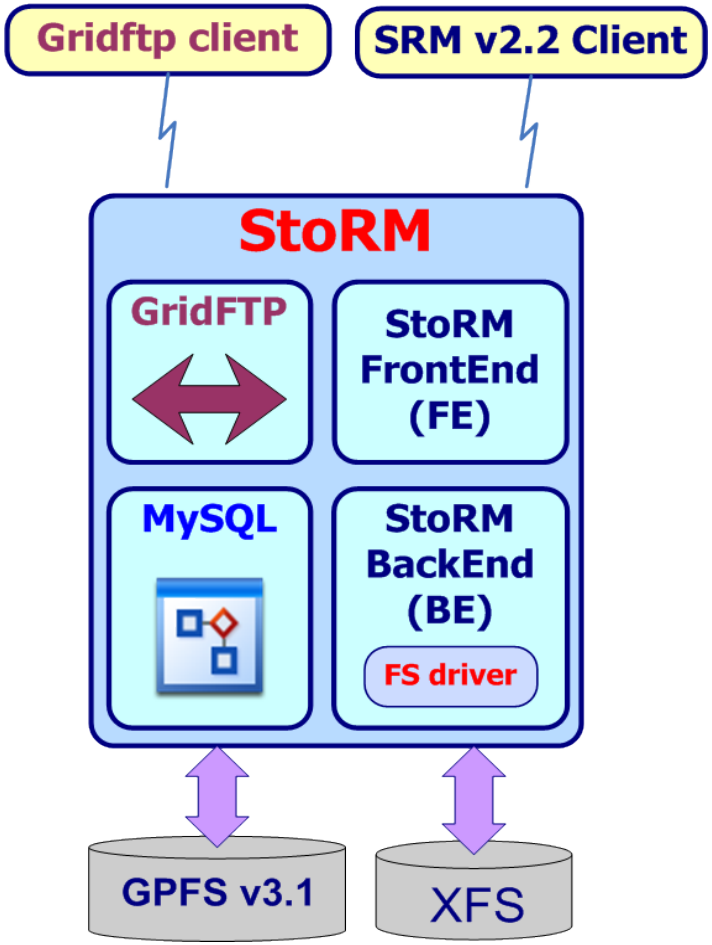
Variable name	Type	Description	>= ig-yaim version
STORM_DB_HOST	C	Host for database connection (StoRM Backend).	4.0.2-9
STORM_DB_PWD	C	Password for database connection.	4.0.2-9

Deployment on single site

All the component:

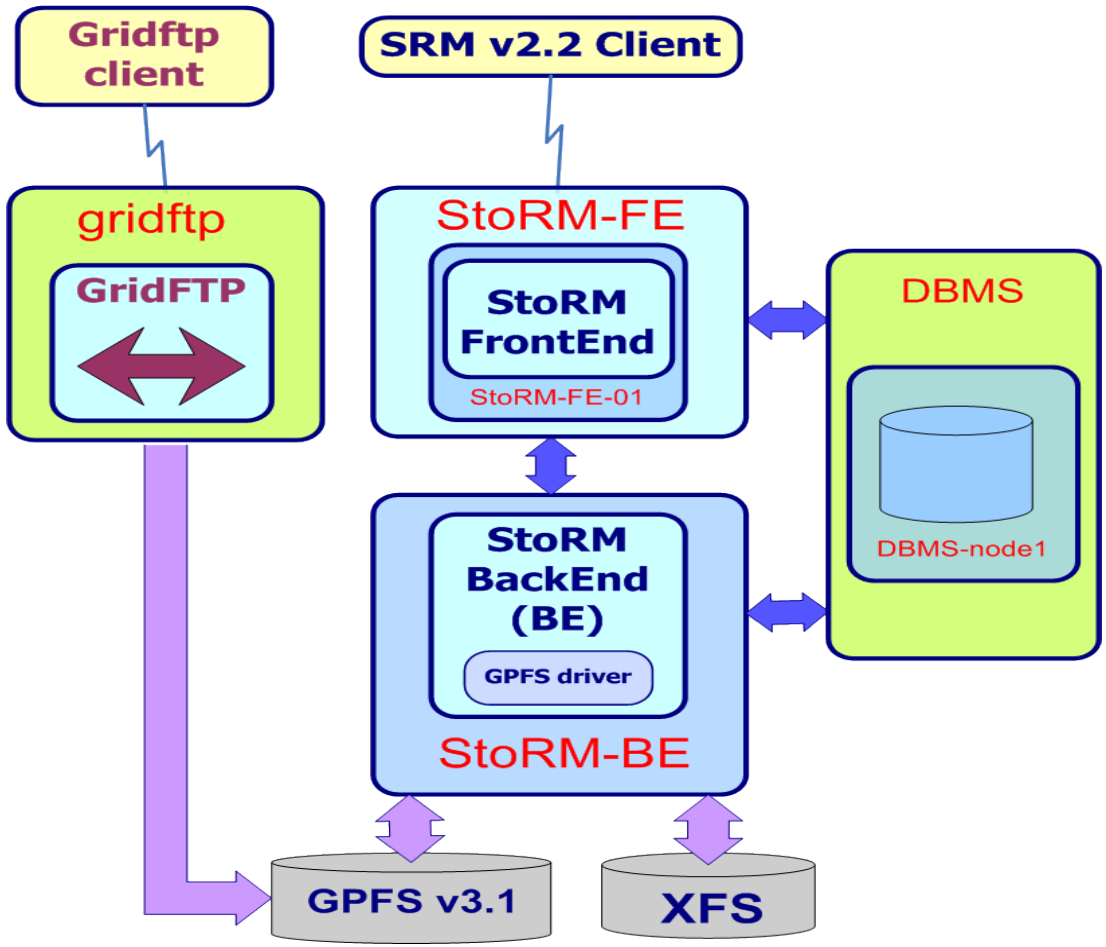
- **StoRM Frontend**
- **StoRM Backend**
- **MySQL**
- **GridFTP**

are **deployed on the same host**



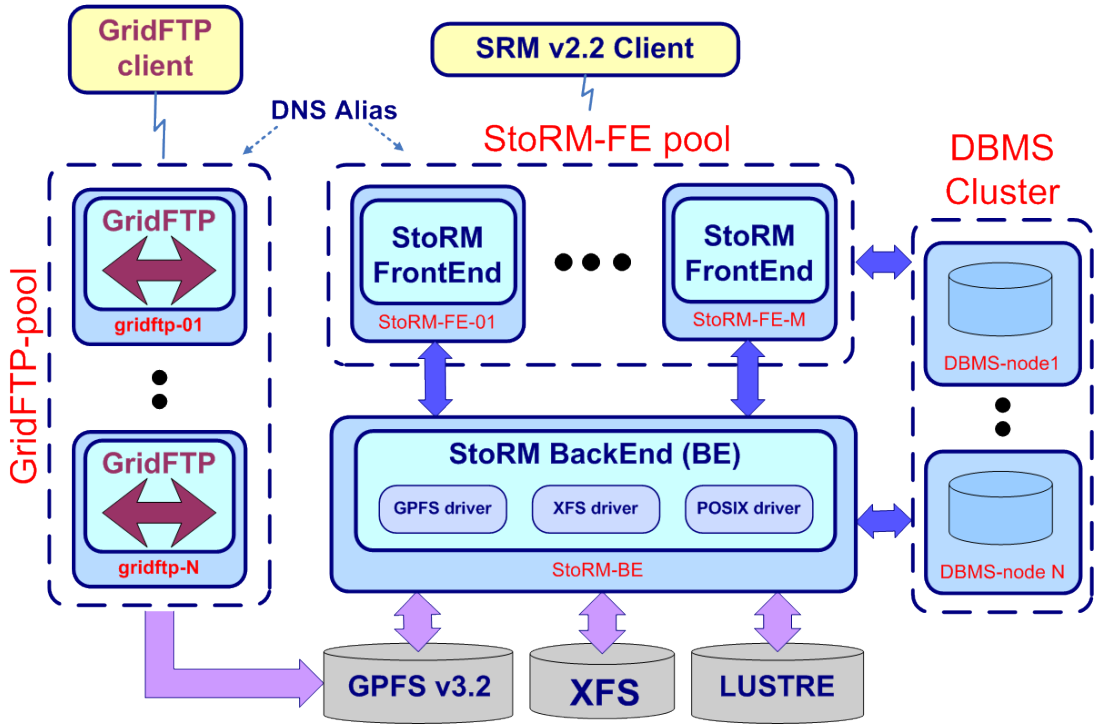
Deployment on different host

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

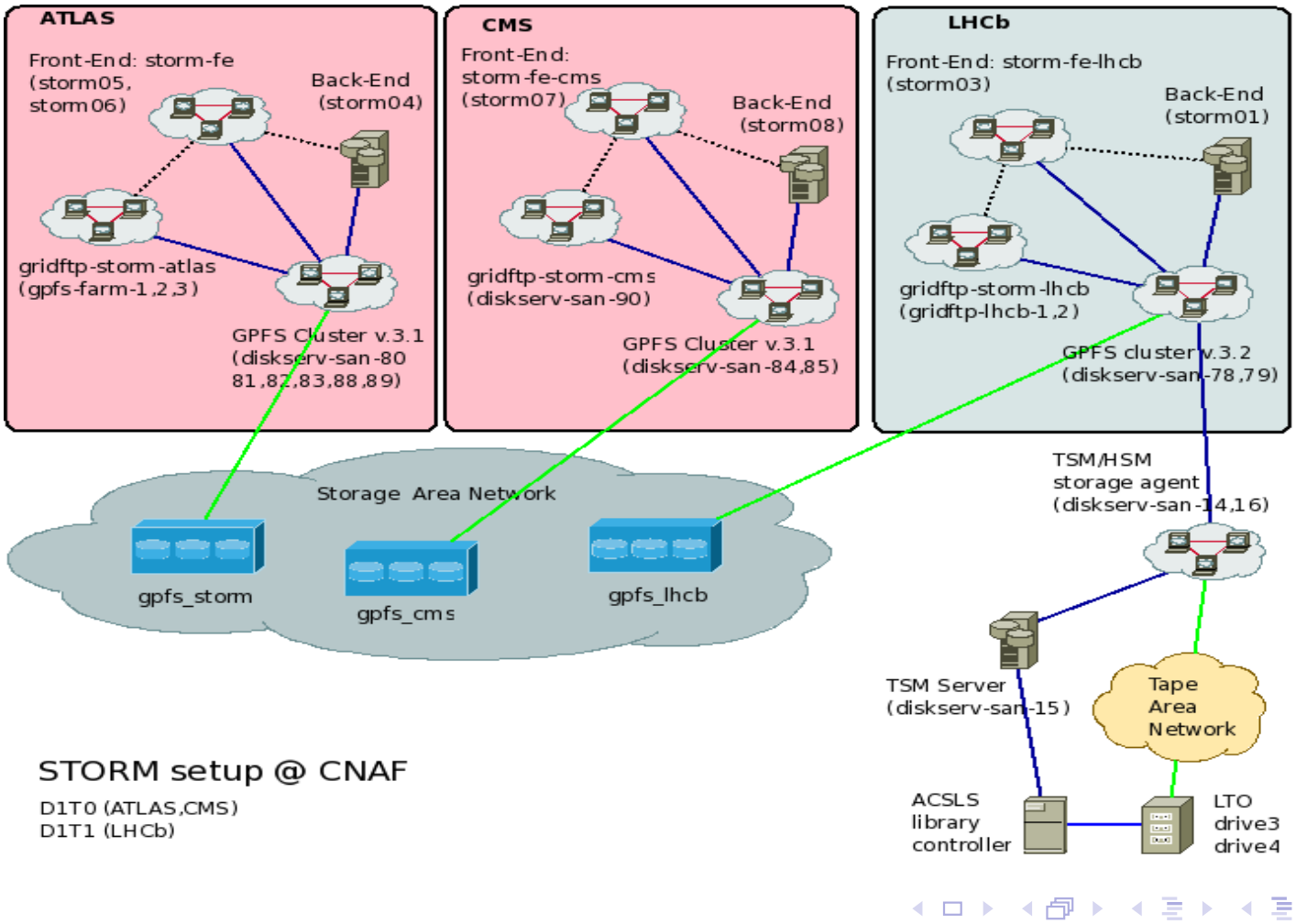


Deployment on cluster

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



StoRM layout at the INFN CNAF Tier 1



Guidelines for deployment

- GridFTP server on separated hosts (expect 70 MB/s of throughput for standalone Gbit machine)
- For clustering StoRM:
  1. Separate Front End on dedicated machine (GSI Auth is cpu intensive)
  2. Replicate Front End instance
  3. Separate MySQL on dedicated machine

About StoRM performance

Statistics from logs analysanalysis off production instance.

- Cluster configuration: 2 Front End and 1 Back End
- **Average** number of SRM requests managed:
  - Synchronous requests (Ls, PutDone, Rm, etc): about 600 per minute
  - Asynchronous requests (PtG, PtP): about 200 per minute
- **Maximum** number of requests managed:
  - Synchronous requests (Ls, PutDone, Rm, etc): $>$ **1600** per minute
  - Asynchronous requests (PtG, PtP): $>$ **600** per minute

Questions

- Questions ?