Architecture of the gLite Data Management System

Tony Calanducci

INFN Catania

CYCLOPS First Training Workshop

Bologna, 11h-13th April 2007

Outline

- Grid Data Management Challenge
- Storage Elements and SRM
- File Catalogs and DM tools
- File Transfer Service
- Metadata Service

Information Society and Media

The Grid DM Challenge

The Grid DM Challenge

- Heterogeneity

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

– Need common interface to storage resources

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)
- Need to keep track where data is stored

CYCLOPS

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)
- Need to keep track where data is stored
  - File and Replica Catalogs

Information Society and Media

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)
- Need to keep track where data is stored
  - File and Replica Catalogs
- Need scheduled, reliable file transfer

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)
- Need to keep track where data is stored
  - File and Replica Catalogs
- Need scheduled, reliable file transfer
  - File transfer service

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
    - Need common interface to storage resources
      - Storage Resource Manager (SRM)
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
    - Need to keep track where data is stored
      - File and Replica Catalogs
  - Data need to be moved between different locations
    - Need scheduled, reliable file transfer
      - File transfer service
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents
    - Need a way to describe files' content and query them

The Grid DM Challenge

- Heterogeneity
  - Data are stored on different storage systems using different access technologies
- Distribution
  - Data are stored in different locations – in most cases there is no shared file system or common namespace
  - Data need to be moved between different locations
- Data description
  - Data are stored as files: need a way to describe files and locate them according to their contents

- Need common interface to storage resources
  - Storage Resource Manager (SRM)
- Need to keep track where data is stored
  - File and Replica Catalogs
- Need scheduled, reliable file transfer
  - File transfer service
- Need a way to describe files' content and query them
  - Metadata service

Introduction

Introduction

- Assumptions:
  - Users and programs produce and require data
  - the lowest granularity of the data is on the file level (we deal with files rather than data objects or tables)
    - Data = files

Introduction

- Assumptions:
  - Users and programs produce and require data
  - the lowest granularity of the data is on the file level (we deal with files rather than data objects or tables)
    - Data = files

- Files:
  - Mostly, write once, read many
  - Located in Storage Elements (SEs)
  - Several replicas of one file in different sites
  - Accessible by Grid users and applications from “anywhere”
  - Locatable by the WMS (data requirements in JDL)

Introduction

- Assumptions:
  - Users and programs produce and require data
  - the lowest granularity of the data is on the file level (we deal with files rather than data objects or tables)
    - Data = files
- Files:
  - Mostly, write once, read many
  - Located in Storage Elements (SEs)
  - Several replicas of one file in different sites
  - Accessible by Grid users and applications from “anywhere”
  - Locatable by the WMS (data requirements in JDL)
- Also…
  - WMS can send (small amounts of) data to/from jobs: Input and Output Sandbox
  - Files may be copied from/to local filesystems (WNs, UIs) to the Grid (SEs)

gLite Grid Storage Requirements

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval

- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval

- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval
- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …
- Be able to manage different storage systems uniformly and transparently for the user (providing an SRM interface)

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval
- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …
- Be able to manage different storage systems uniformly and transparently for the user (providing an SRM interface)

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval

- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …

- Be able to manage different storage systems uniformly and transparently for the user (providing an SRM interface)

- Support basic file transfer protocols
  - GridFTP mandatory
  - Others if available (https, ftp, etc)

CYCLOPS

gLite Grid Storage Requirements

- The Storage Element is the service which allow a user or an application to store data for future retrieval
- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
  - HPSS, CASTOR, DiskeXtender (UNITREE), …
- Be able to manage different storage systems uniformly and transparently for the user (providing an SRM interface)
- Support basic file transfer protocols
  - GridFTP mandatory
  - Others if available (https, ftp, etc)
- Support a native I/O (remote file) access protocol
  - POSIX (like) I/O client library for direct access of data (GFAL)

Information Society and Media

SRM in an example

She is running a job which needs:

Data for physics event reconstruction

Simulated Data

Some data analysis files

She will write files remotely too

They are at CERN
In dCache

They are at Fermilab
In a disk array

They are at Nikhef
in a classic SE

SRM in an example

dCache

Own system, own protocols and parameters

gLite DPM

Independent system from dCache or Castor

Castor

No connection with dCache or DPM

You as a user need to know all the systems!!!

SRM in an example

dCache

Own system, own protocols and parameters

gLite DPM

Independent system from dCache or Castor

Castor

No connection with dCache or DPM

SRM

I talk to them on your behalf
I will even allocate space for your files
And I will use transfer protocols to send your files there

CYCLOPS

Storage Resource Management

Information Society and Media

CYCLOPS

Storage Resource Management

- Data are stored on disk pool servers or Mass Storage Systems

Information Society and Media

Storage Resource Management

- Data are stored on disk pool servers or Mass Storage Systems
- storage resource management needs to take into account
  - Transparent access to files (migration to/from disk pool)
  - File pinning
  - Space reservation
  - File status notification
  - Life time management

Storage Resource Management

- Data are stored on disk pool servers or Mass Storage Systems
- storage resource management needs to take into account
  - Transparent access to files (migration to/from disk pool)
  - File pinning
  - Space reservation
  - File status notification
  - Life time management
- The SRM (Storage Resource Manager) takes care of all these details
  - The SRM is a single interface that takes care of local storage interaction and provides a Grid outside world

CYCLOPS

Storage Resource Management

- Data are stored on disk pool servers or Mass Storage Systems
- storage resource management needs to take into account
  - Transparent access to files (migration to/from disk pool)
  - File pinning
  - Space reservation
  - File status notification
  - Life time management
- The SRM (Storage Resource Manager) takes care of all these details
  - The SRM is a single interface that takes care of local storage interaction and provides a Grid outside world
- In gLite, interactions with the SRM is hidden by higher level services (DM tools and APIs)

Information Society and Media

gLite SE types

- gLite 3.0 data access protocols:
  - File Transfer: GSIFTP (GridFTP)
  - File I/O (Remote File access):
    - gsidcap
    - insecure RFIO
    - secured RFIO (gsirfio)

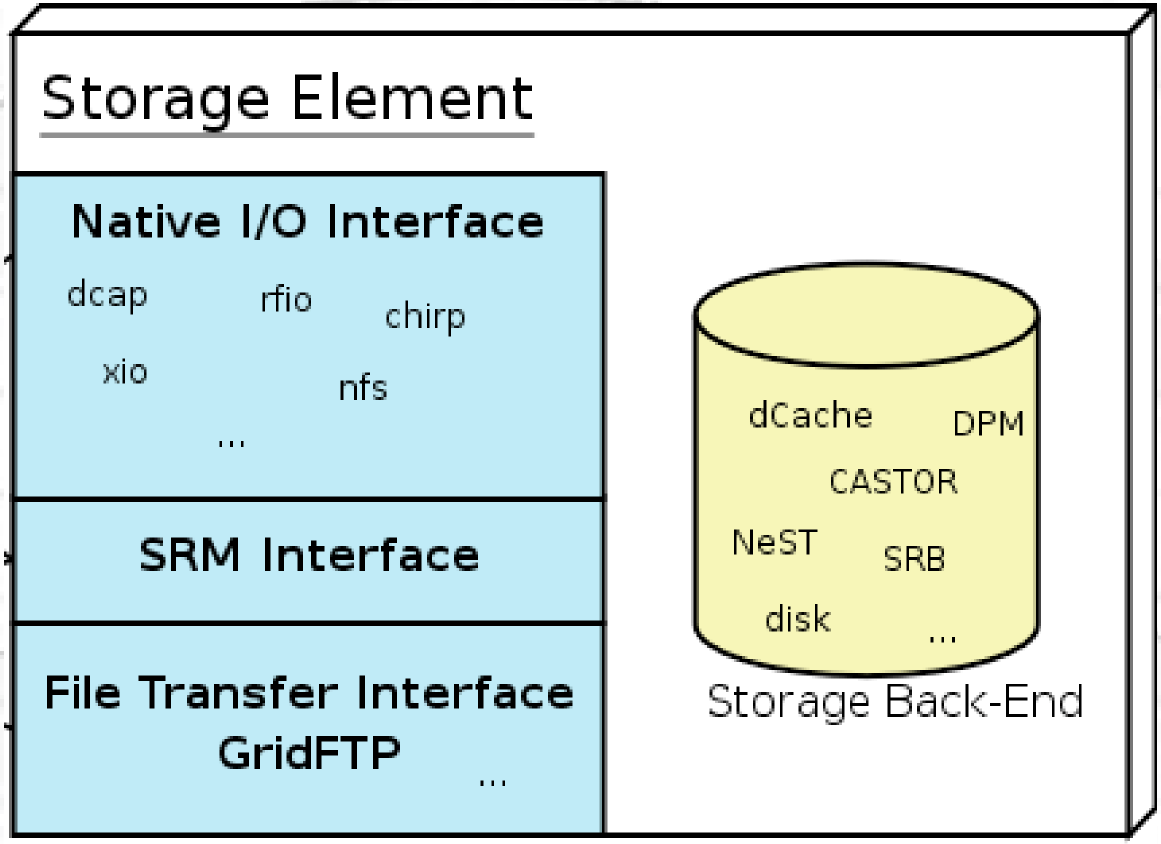
- Classic SE:
  - GridFTP server
  - Insecure RFIO daemon (rfiod) – only LAN limited file access
  - Single disk or disk array
  - No quota management
  - Does not support the SRM interface

gLite SE types (II)

- Mass Storage Systems (Castor)
  - Files migrated between front-end disk and back-end tape storage hierarchies
  - GridFTP server
  - Insecure RFIO (Castor)
  - Provide a SRM interface with all the benefits
- Disk pool managers (dCache and gLite DPM)
  - manage distributed storage servers in a centralized way
  - Physical disks or arrays are combined into a common (virtual) file system
  - Disks can be dynamically added to the pool
  - GridFTP server
  - Secure remote access protocols (gsidcap for dCache, gsirfio for DPM)
  - SRM interface

CYCLOPS

gLite Storage Element

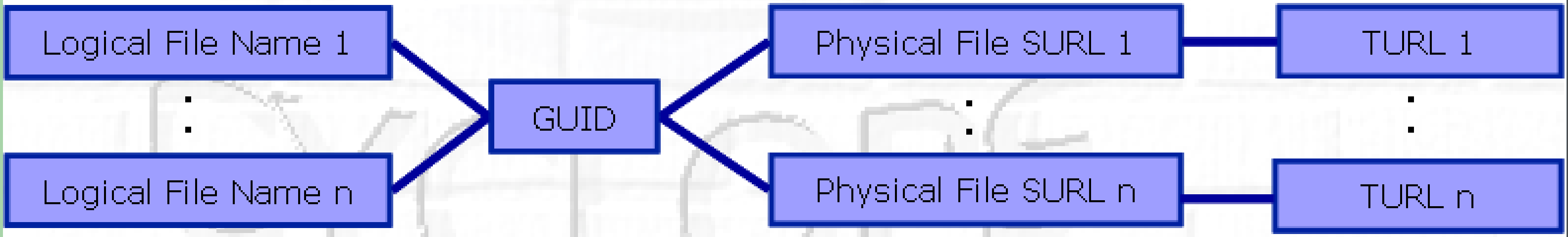


Information Society and Media

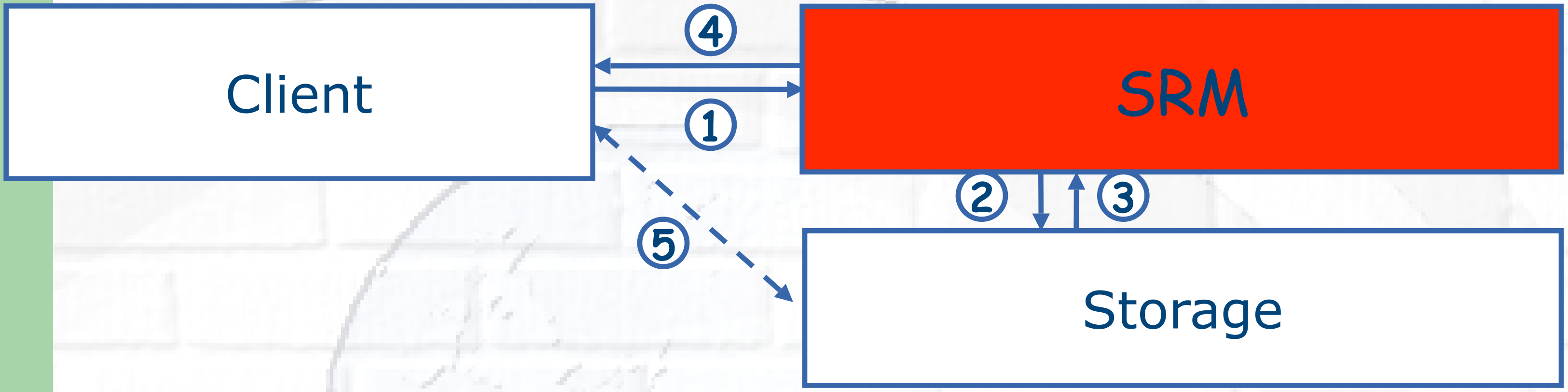
Files Naming conventions

- Logical File Name (LFN)
  - An alias created by a user to refer to some item of data, e.g. “lfn:/grid/gilda/20030203/run2/track1”
- Globally Unique Identifier (GUID)
  - A non-human-readable unique identifier for an item of data, e.g. “guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6”
- Site URL (SURL) (or Physical File Name (PFN) or Site FN)
  - The location of an actual piece of data on a storage system

    e.g. “srm://grid009.ct.infn.it/dpm/ct.infn.it/gilda/output10_1” (SRM) “sfn://lxshare0209.cern.ch/data/alice/ntuples.dat” (Classic SE)
- Transport URL (TURL)
  - Temporary locator of a replica + access protocol: understood by a SE, e.g. “rfio://lxshare0209.cern.ch//data/alice/ntuples.dat”



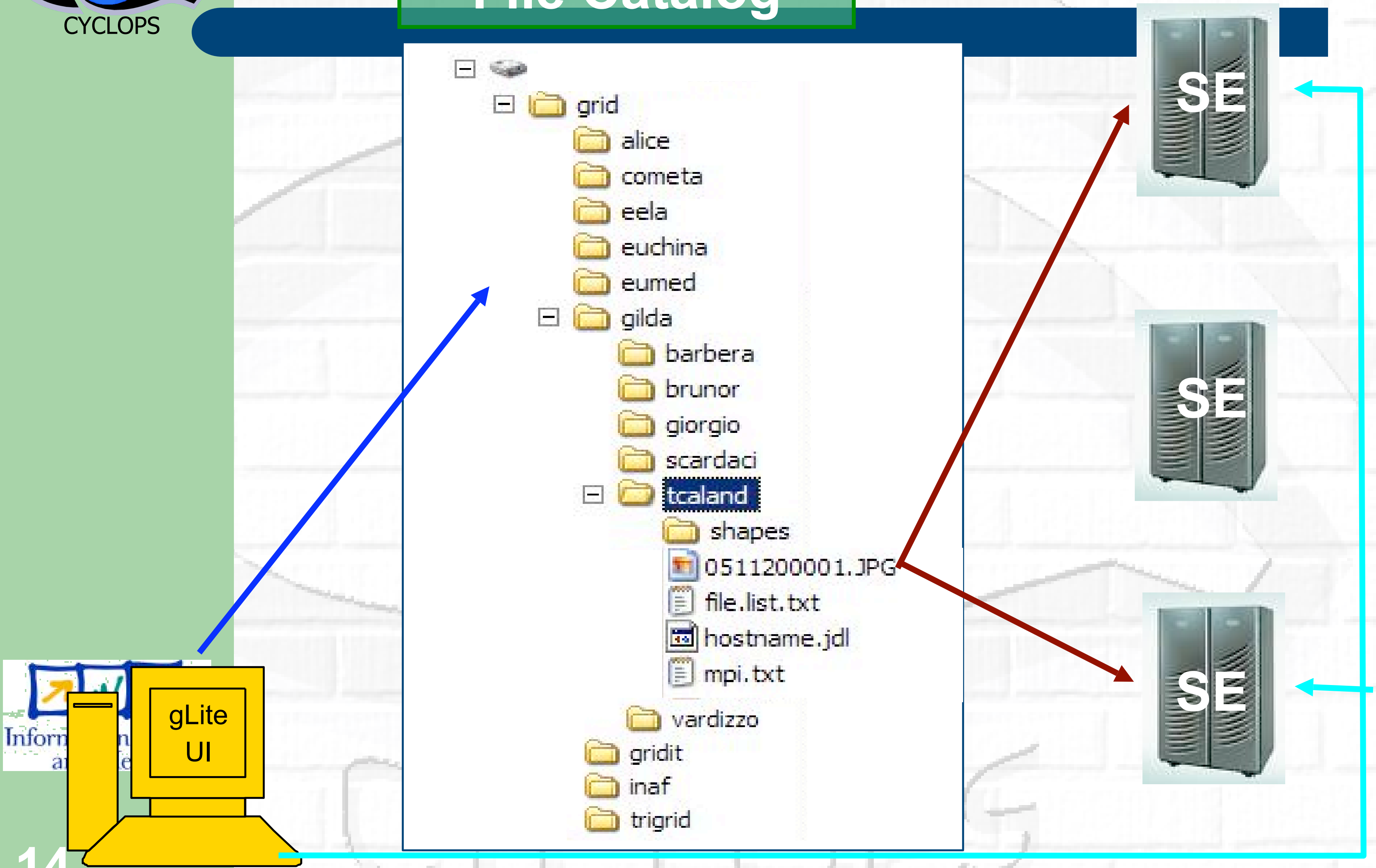
SRM Interactions



1. The client asks the SRM for the file providing an SURL (Site URL)
2. The SRM asks the storage system to provide the file
3. The storage system notifies the availability of the file and its location
4. The SRM returns a TURL (Transfer URL), i.e. the location from where the file can be accessed

What is a file catalog

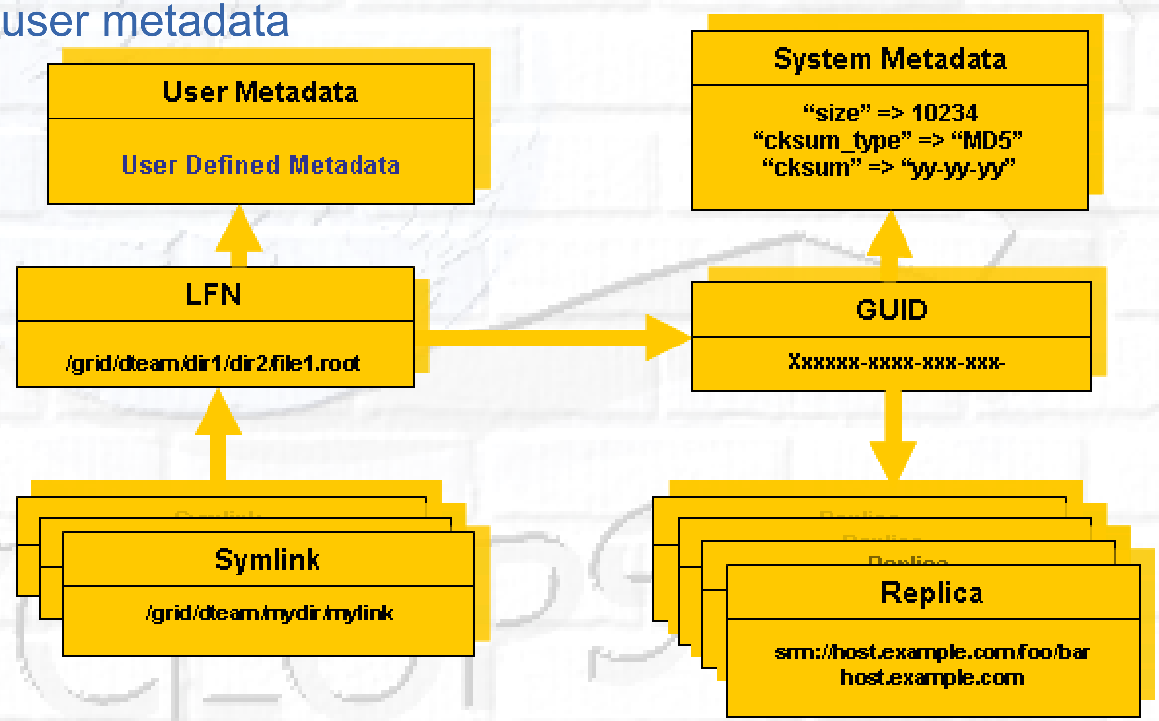
File Catalog



- grid
  - alice
  - cometa
  - eela
  - euchina
  - eumed
  - gilda
    - barbera
    - brunor
    - giorgio
    - scardaci
    - tcaland
      - shapes
      - 0511200001.JPG
      - file.list.txt
      - hostname.jdl
      - mpi.txt
    - vardizzo
  - gridit
  - inaf
  - trigrid

The LFC (LCG File Catalog)

- It keeps track of the location of copies (replicas) of Grid files
- LFN acts as main key in the database. It has:
  - Symbolic links to it (additional LFNs)
  - Unique Identifier (GUID)
  - System metadata
  - Information on replicas
  - One field of user metadata



LFC Features

- Cursors for large queries
- Timeouts and retries from the client
- User exposed transactional API (+ auto rollback on failure)
- Hierarchical namespace and namespace operations (for LFNs)
- Integrated GSI Authentication + Authorization
- Access Control Lists (Unix Permissions and POSIX ACLs)
- Checksums

LFC commands

Summary of the LFC Catalog commands

lfc-chmod	Change access mode of the LFC file/directory
lfc-chown	Change owner and group of the LFC file-directory
lfc-delcomment	Delete the comment associated with the file/directory
lfc-getacl	Get file/directory access control lists
lfc-ln	Make a symbolic link to a file/directory
lfc-ls	List file/directory entries in a directory
lfc-mkdir	Create a directory
lfc-rename	Rename a file/directory
lfc-rm	Remove a file/directory
lfc-setacl	Set file/directory access control lists
lfc-setcomment	Add/replace a comment

lfc-ls

Listing the entries of a LFC directory

lfc-ls *[-cdiLlRTu] [--class] [--comment] [--deleted] [--display_side] [--ds] path…*

where *path* specifies the LFN pathname (mandatory)

- Remember that **LFC has a directory tree structure**
- /grid/<VO_name>/<you create it>

LFC Namespace

Defined by the user

- All members of a VO have read-write permissions under their directory
- You can set LFC_HOME to use relative paths

-l : long listing
-R : list the contents of directories recursively: Don't use it!

lfc-mkdir

Creating directories in the LFC

lfc-mkdir [-m mode] [-p] path...

- Where *path* specifies the LFC pathname
- Remember that while registering a new file (using lcg-cr, for example) the corresponding destination directory must be created in the catalog beforehand.
- Examples:

```
> lfc-mkdir /grid/gilda/tony/demo
```

You can just check the directory with:

```
> lfc-ls -l /grid/gilda/tony
    drwxr-xr-wx   0 19122    1077                0 Jun 14 11:36 demo
```

lfc-ln

Creating a symbolic link

```
lfc-ln   -s  file   linkname
lfc-ln   -s  directory   linkname
```

Create a link to the specified *file* or *directory* with *linkname*

- *Examples:*

```
> lfc-ln -s   /grid/gilda/tony/demo/test   /grid/gilda/tony/aLink
```

Original File

Symbolic link

Let’s check the link using lfc-ls with long listing (-l):

```
> lfc-ls  -l
lrwxrwxrwx   1 19122  1077   0 Jun 14 11:58 aLink ->/grid/gilda/tony/demo/test
drwxr-xrwx   1 19122  1077   0 Jun 14 11:39 demo
```

LFC C API

Low level methods (many POSIX-like):

lfc_access
lfc_aborttrans
lfc_addreplica
lfc_apiinit
lfc_chclass
lfc_chdir
lfc_chmod
lfc_chown
lfc_closedir
lfc_creat
lfc_delcomment
lfc_delete
lfc_deleteclass
lfc_delreplica
lfc_endtrans
lfc_enterclass
lfc_errmsg
lfc_getacl
lfc_getcomment
lfc_getcwd
lfc_getpath
lfc_lchown
lfc_listclass
lfc_listlinks
lfc_listreplica
lfc_lstat
lfc_mkdir
lfc_modifyclass
lfc_opendir
lfc_queryclass
lfc_readdir
lfc_readlink
lfc_rename
lfc_rewind
lfc_rmdir
lfc_selectsrvr
lfc_setacl
lfc_setatime
lfc_setcomment
lfc_seterrbuf
lfc_setfsize
lfc_starttrans
lfc_stat
lfc_symlink
lfc_umask
lfc_undelete
lfc_unlink
lfc_utime
send2lfc

CYCLOPS

GFAL: Grid File Access

Interactions with SE require some components:

- → File catalog services to locate replicas
- → SRM
- → File access mechanism to access files from the SE on the WN

GFAL does all this tasks for you:

- → Hides all these operations
- → Presents a POSIX interface for the I/O operations
  - → Single shared library in threaded and unthreaded versions

    libgfal.so, libgfal_pthr.so
  - → Single header file: gfal_api.h
- → User can create all commands needed for storage management
- → It offers as well an interface to SRM

Supported protocols:

- → file (local or nfs-like access)
- → dcap, gsidcap and kdcap (dCache access)
- → rfio (castor access) and gsirfio (dpm)

Information Society and Media

CYCLOPS

GFAL: File I/O API (I)

```
int gfal_access (const char *path, int amode);
int gfal_chmod (const char *path, mode_t mode);
int gfal_close (int fd);
int gfal_creat (const char *filename, mode_t mode);
off_t gfal_lseek (int fd, off_t offset, int whence);
int gfal_open (const char * filename, int flags, mode_t mode);
ssize_t gfal_read (int fd, void *buf, size_t size);
int gfal_rename (const char *old_name, const char *new_name);
ssize_t gfal_setfilchg (int, const void *, size_t);
int gfal_stat (const char *filename, struct stat *statbuf);
int gfal_unlink (const char *filename);
ssize_t gfal_write (int fd, const void *buf, size_t size);
```

Information Society and Media

GFAL: File I/O API (II)

```
int gfal_closedir (DIR *dirp);

int gfal_mkdir (const char *dirname, mode_t mode);

DIR *gfal_opendir (const char *dirname);

struct dirent *gfal_readdir (DIR *dirp);

int gfal_rmdir (const char *dirname);
```

GFAL: Catalog API

int **create_alias** (const char *guid, const char *lfn, long long size)

int **guid_exists** (const char *guid)

char ***guidforpfn** (const char *surl)

char ***guidfromlfn** (const char *lfn)

char ****lfnsforguid** (const char *guid)

int **register_alias** (const char *guid, const char *lfn)

int **register_pfn** (const char *guid, const char *surl)

int **setfilesize** (const char *surl, long long size)

char ***surlfromguid** (const char *guid)

char ****surlsfromguid** (const char *guid)

int **unregister_alias** (const char *guid, const char *lfn)

int **unregister_pfn** (const char *guid, const char *surl)

GFAL: Storage API

int **deletesurl** (const char *surl)

int **getfilemd** (const char *surl, struct stat64 *statbuf)

int **set_xfer_done** (const char *surl, int reqid, int fileid, char *token, int oflag)

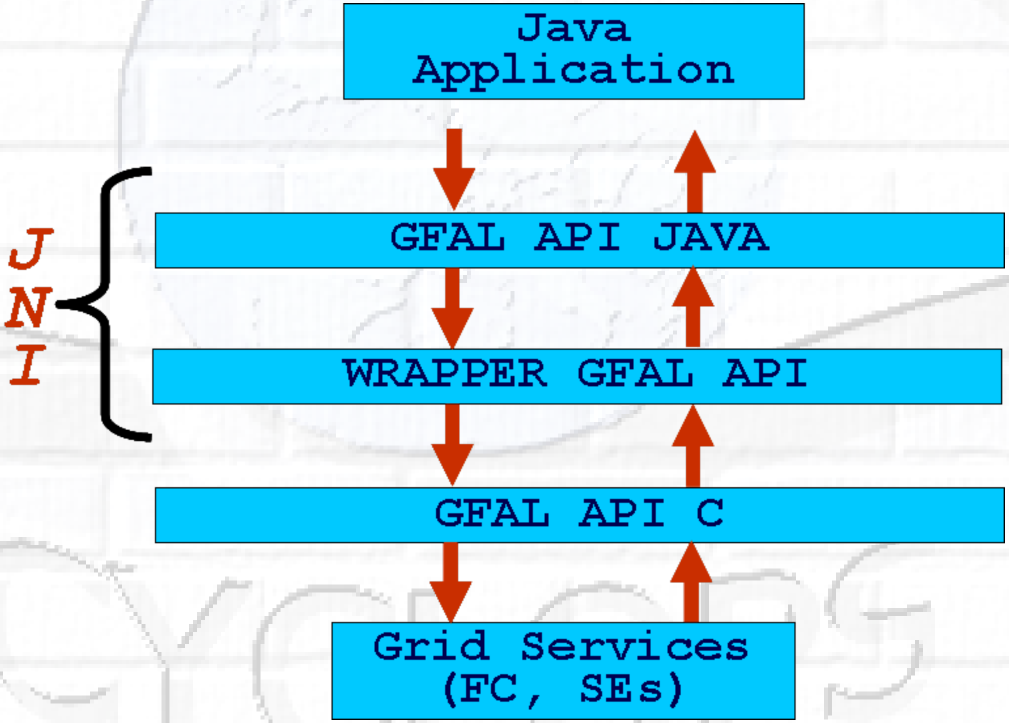
int **set_xfer_running** (const char *surl, int reqid, int fileid, char *token)

char ***turlfromsurl** (const char *surl, char **protocols, int oflag, int *reqid, int *fileid, char **token)

GFAL Java API

- GFAL API are available for C/C++ programmers
- We wrote a wrapper around the C APIs using Java Native Interface and a the Java APIs on top of it
- More information can be found here:

  https://grid.ct.infn.it/twiki/bin/view/GILDA/APIGFAL



lcg-utils DM tools

- High level interface (CL tools and APIs) to
  - Upload/download files to/from the Grid (UI,CE and WN <---> SEs)
  - Replicate data between SEs and locate the best replica available
  - Interact with the file catalog

- ***Definition***: A file is considered to be a Grid File if it is both physically present in a SE and registered in the File Catalog

- lcg-utils ensure the consistency between files in the Storage Elements and entries in the File Catalog

lcg-utils commands

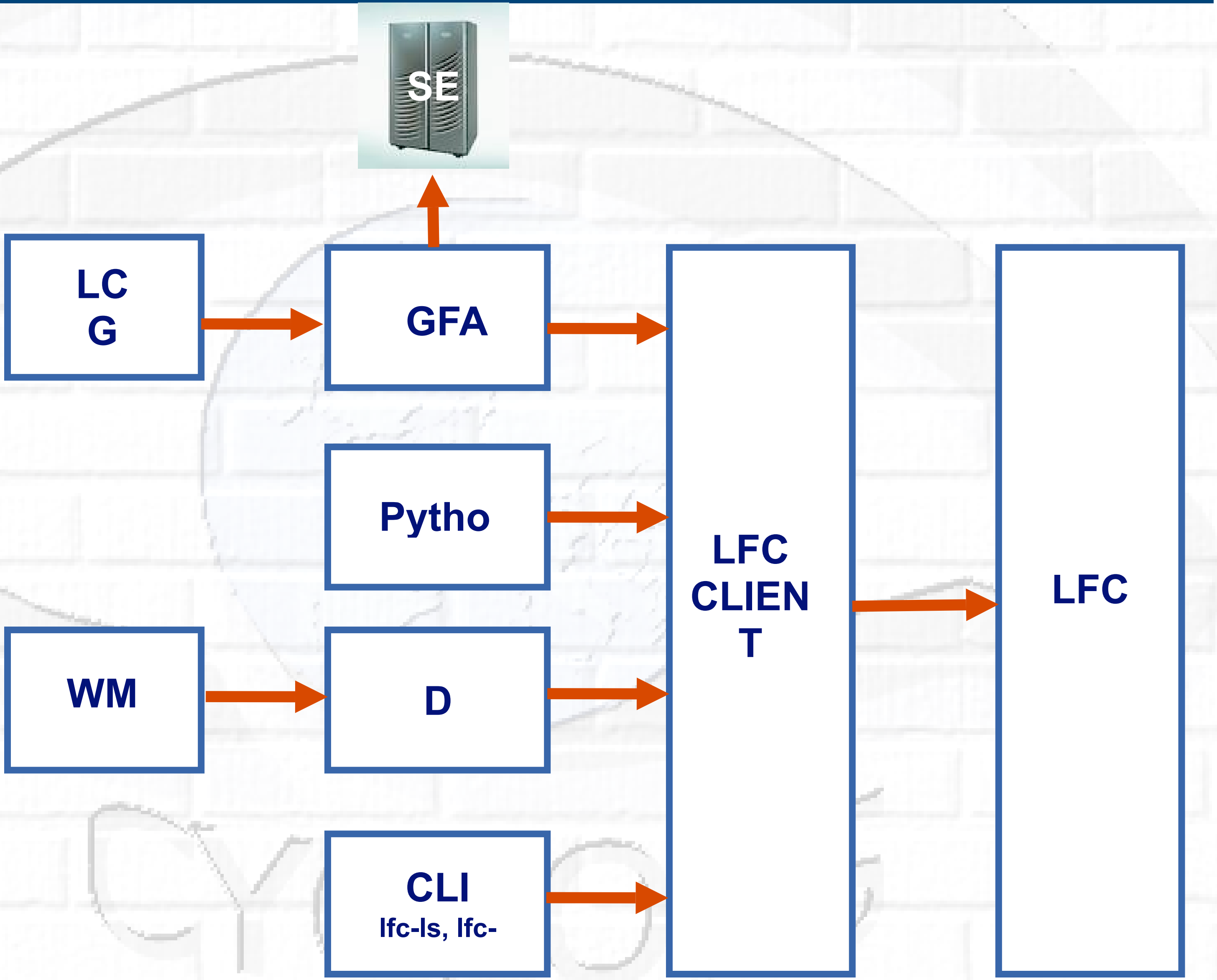
Replica Management

lcg-cp	Copies a grid file to a local destination
lcg-cr	Copies a file to a SE and registers the file in the catalog
lcg-del	Delete one file
lcg-rep	Replication between SEs and registration of the replica
lcg-gt	Gets the TURL for a given SURL and transfer protocol
lcg-sd	Sets file status to “Done” for a given SURL in a SRM request

File Catalog Interaction

lcg-aa	Add an alias in LFC for a given GUID
lcg-ra	Remove an alias in LFC for a given GUID
lcg-rf	Registers in LFC a file placed in a SE
lcg-uf	Unregisters in LFC a file placed in a SE
lcg-la	Lists the alias for a given SURL, GUID or LFN
lcg-lg	Get the GUID for a given LFN or SURL
lcg-lr	Lists the replicas for a given GUID, SURL or LFN

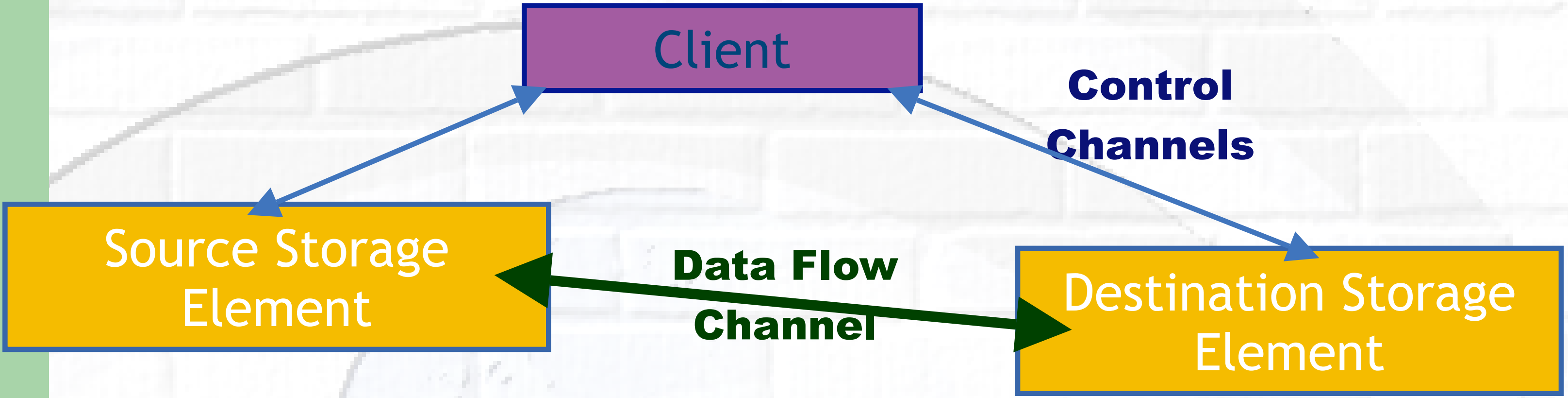
LFC interfaces



Data movement introduction

- Grids are naturally distributed systems
- The means that data also needs to be distributed
  - First generation data distribution mainly concentrated on copy protocols in a grid environment:
    - gridftp
    - http + mod_gridsite
- But copies controlled by clients have problems…

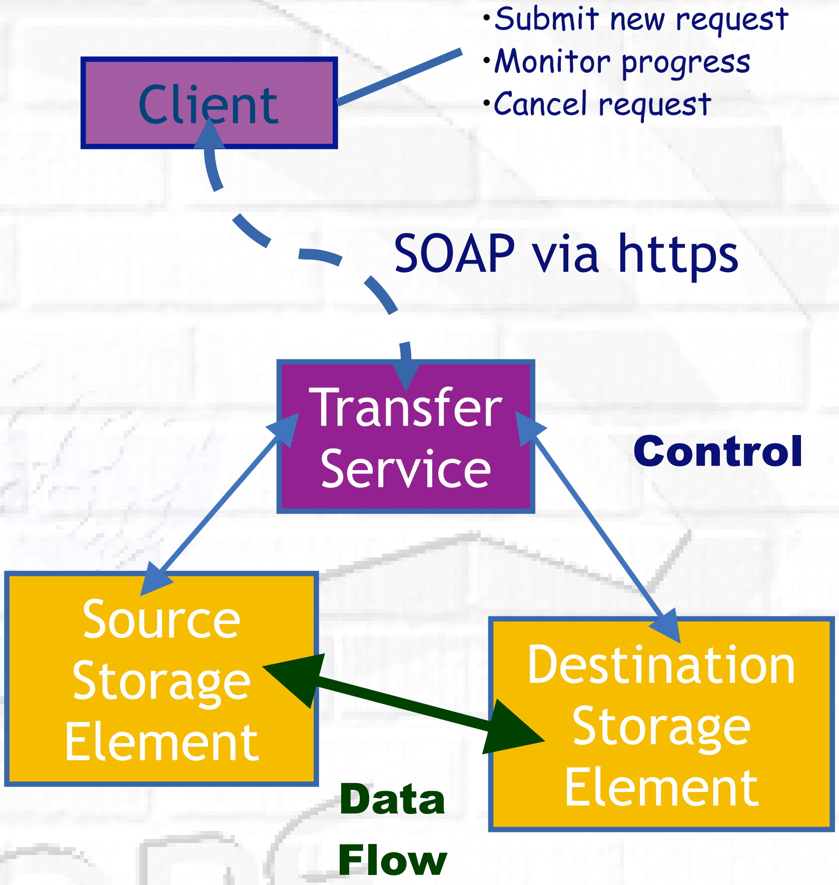
Direct Client Controlled Data Movement



- Although transport protocol may be robust, state is held inside client – inconvenient and fragile.
- Client only knows about local state, no sense of global knowledge about data transfers between storage elements.
  - Storage elements overwhelmed with replication requests
  - Multiple replications of the same data can happen simultaneously
  - Site has little control over balance of network resources - DOS

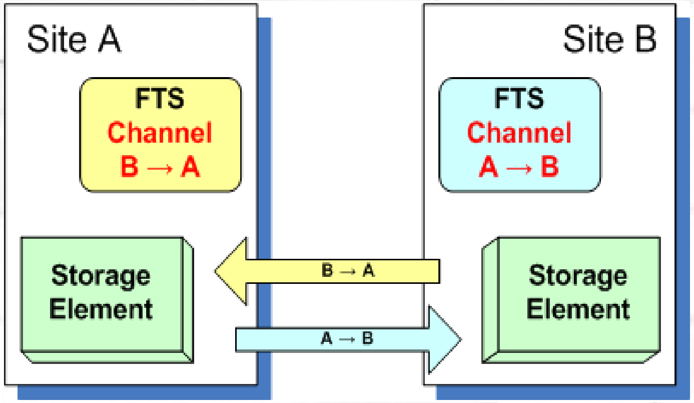
Transfer Service

- Clear need for a ***service*** for data transfer
  - Client connects to service to submit request
  - Service maintains state about transfer
  - Client can periodically reconnect to check status or cancel request
  - Service can have knowledge of global state, not just a single request
    - Load balancing
    - Scheduling



gLite FTS: Channels

- FTS Service has a concept of ***channels***
- A channel is a ***unidirectional*** connection between two sites
- Transfer requests between these two sites are assigned to that channel
- Channels usually correspond to a dedicated network pipe (e.g., OPN) associated with production
- But channels can also take wildcards:
  - * to MY_SITE : All incoming
  - MY SITE to * : All outgoing
  - * to * : Catch all



- Channels control certain transfer properties: transfer concurrency, gridftp streams.
- Channels can be controlled independently: started, stopped, drained.

Data Management Services Summary

- Storage Element – save date and provide a common interface
  - Storage Resource Manager (SRM) Castor, dCache, DPM, …
  - Native Access protocols rfio, dcap, nfs, …
  - Transfer protocols gsiftp, ftp, …
- Catalogs – keep track where data are stored
  - File Catalog
  - Replica Catalog

  LCG File Catalog (LFC)
  - Metadata Catalog AMGA Metadata Catalogue
- Data Movement – schedules reliable file transfer
  - File Transfer Service gLite FTS

    (manages physical transfers)

References

- gLite documentation homepage
  - http://glite.web.cern.ch/glite/documentation/default.asp
- DM subsystem documentation
  - http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/doc.htm
- LFC and DPM documentation
  - https://uimon.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation

Questions…