Data Management

Erwin Laure
EGEE Deputy Middleware Manager

www.eu-egee.org

Information Society

Outline

- The Grid Data Management Challenge
- Overview of Data Management services
- File Storage Systems
- File and Replica Catalogs
- File I/O
- File Movement

- Disclaimer: data can be stored in files or as structured data in databases – in the following we deal only with files as this is the most common usecase in the HEP community

The Grid DM Challenge

- **Heterogeneity**
  - Data is stored on different storage systems using different access technologies
  - Need common interface to storage resources
    - Storage Resource Manager (SRM)
- **Distribution**
  - Data is stored in different locations – in most cases there is no shared file system or common namespace
  - Data needs to be moved between different locations
  - Need to keep track where data is stored
    - File and Replica Catalogs
  - Need scheduled, reliable file transfer
    - File transfer and placement services
- **Different Administrative Domains**
  - Data is stored at places you would normally have no access to
  - Security and auditing implications
  - Need a common security model
    - ACLs enforcement based on Grid identities – DNs

Data Management Services

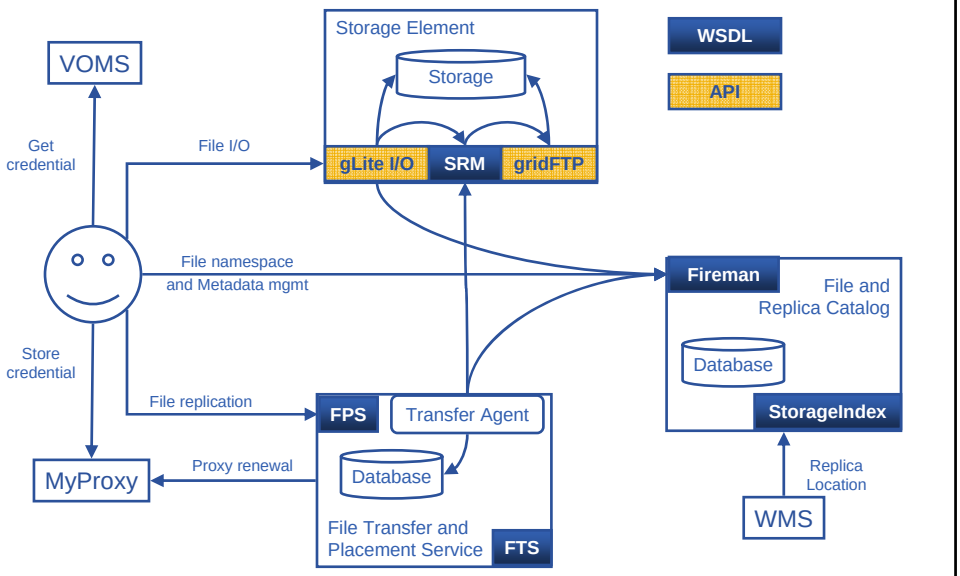
- **Storage Element** – common interface to storage
  - Storage Resource Manager — Castor, dCache, DPM, …
  - POSIX-I/O — gLite-I/O, rfio, dcap, xrootd
  - Access protocols — gsiftp, https, rfio, …
- **Catalogs** – keep track where data is stored
  - File Catalog
  - Replica Catalog
  - File Authorization Service
  - Metadata Catalog

  → gLite File and Replica Catalog; Globus RLS; Application specific catalogs
- **File Transfer** – scheduled reliable file transfer
  - Data Scheduler — (only designs exist so far)
  - File Transfer Service (manages physical transfer) — gLite FTS and glite-url-copy; Globus RFT, Stork
  - File Placement Service — gLite FPS

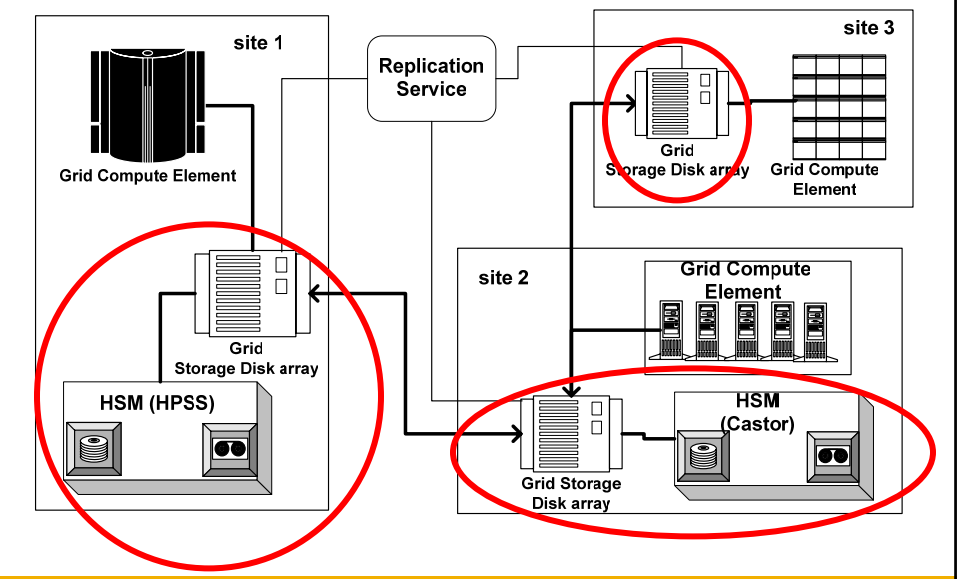
  (FTS and catalog interaction in a transactional way)

We do not cover security in this lecture

DM Interaction Overview



Grid Storage Devices



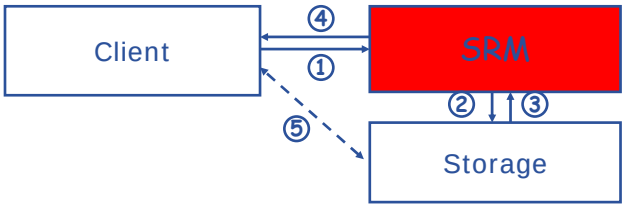
Grid Storage Requirements

- **Manage local storage and interface to Mass Storage Systems like**
  - HPSS, CASTOR, DiskeXtender (UNITREE), …
- **Provide a unique interface**
- **Support basic file transfer and access protocols**
  - GridFTP, FTP, POSIX-like…

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
- SRM (Storage Resource Manager) takes care of all these details
  - SRM is a Grid Service that takes care of local storage interaction and provides a Grid interaface to outside world
- Interactions with the SRM is typically hidden by higher level services – will not be exercised

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
5. The client interacts with the storage using the protocol specified in the TURL

The File Naming Problem

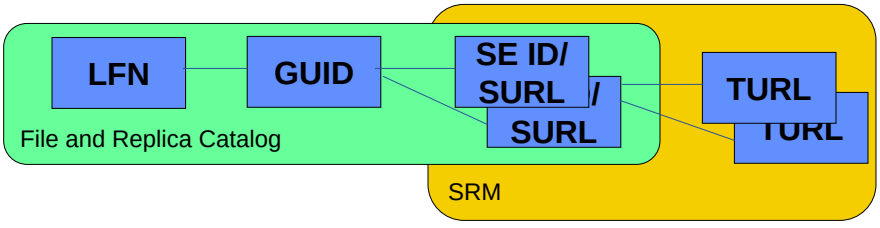
- **File names typically have only a local meaning**
  - `/home/csc/csc05/students.dat` (Unix)
  - `srm://castorgrid.cern.ch:8443/srm/managerv1?SFN=/castor/cern.ch/file1` (SRM Site URL – SURL)
- **The local storage system may transform filenames e.g. an SURL cannot be accessed directly, it needs to be transformed into a Transfer URL (TURL) by an SRM:**
  - `gsiftp://se05.cern.ch/scratch/file05`
- **In order to locate files on the Grid we need mechanisms to abstract from local file naming and provide a grid-wide view on files**

The File Naming Problem II

- **Logical File Names (LFN)**
  - Provide a human readable identifier for files on the Grid level
  - Can be arbitrary URIs
  - Need to be unique
  - `lfn:///glite/myVO.org/production/run/07/123456/calibration/cal/cal-table100`
- **Global Unique Identifiers (GUID)**
  - LFNs may be created in a distributed fashion – hard to guarantee uniqueness
  - Assigning a GUID to each file when it is created allows to always uniquely identify it and thus conflict resolution in LFNs
  - Drawback: not human readable
  - `004c3326-0daf-126d-87f9-898a04b4beef`

File and Replica Catalogs

- **Allow to find where files are stored on the Grid**
- **May implement additional semantics in the LFN namespace (directories, ACLs, …)**
- **Allow to locate the location of replicas (i.e. identical copies of files)**



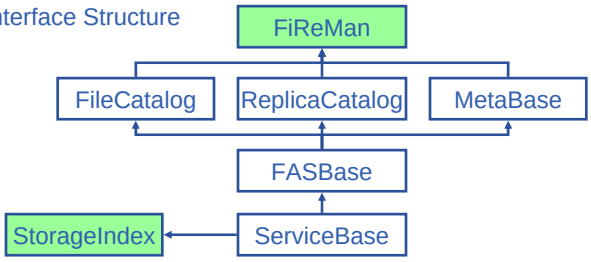
LFN – Logical File Name
GUID – Global Unique Identifier
SURL – Site URL
TURL – Transfer URL

gLite FiReMan Catalog details

Enabling Grids for E-sciencE

- **Web Service interface (WSDL)**
- **Mostly Bulk operations**
- **Stateless interaction**
- **No transactions outside Bulk**

Interface Structure



- StorageIndex: file location for broker
- FAS: File Access Service (ACLs)
- File Catalog: directory structure in LFN namespace
- Replica Catalog: location of replicas
- Meta: additional (user defined metadata)

Implemented on top of Oracle and MySQL

Main FiReMan Commands I

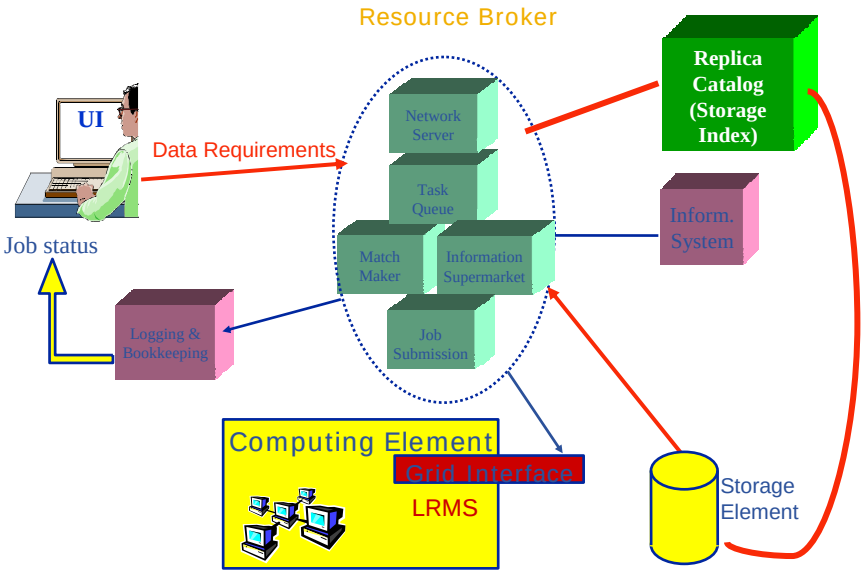
Enabling Grids for E-sciencE

- **List the contents of a directory**
  - `glite-catalog-ls <directory>`
    - `e.g. glite-catalog-ls /tmp`
  - Usual UNIX ls options: -v -l
- **Get details of a file**
  - `glite-catalog-stat <file or directory>`
    - `e.g. glite-catalog-stat /tmp/myfile`
- **Create a directory**
  - `glite-catalog-mkdir <dir>`
- **Locate a file**
  - `glite-catalog-find <pattern> <dir>`

Main FiReMan Commands II

- **Create an LFN**
  - `glite-catalog-create <lfn>`
- **Create a replica**
  - `glite-catalog-setreplica -a <surl> <lfn>`
  - First replica needs to be a *master*:
  - `glite-catalog-setreplica –m <surl> <lfn>`
- **List replicas**
  - `glite-catalog-getreplica <lfn>`
- **Remove a replica**
  - `glite-catalog-setreplica -d <surl> <lfn>`
- **Remove an LFN**
  - `glite-catalog-rm <lfn>`

Using Data Location for Job Scheduling



Using Data Location for Job Scheduling

```
[
Executable = "helloCSC.sh";
StdOutput = "Message.txt";
StdError = "stderr.log";
StorageIndex = "http://lxb2028.cern.ch:8080/EGEE/glite-data-
catalog-service-fr/services/SEIndex";
InputData = "lfn:///tmp/testCSC";
DataAccessProtocol = "gridftp,gliteio";
InputSandbox = {"helloGet.sh"};
OutputSandbox = {"Message.txt","stderr.log", "testfile.txt"};
]
```

Endpoint of the Catalog (StorageIndex interface)

LFN of the file needed

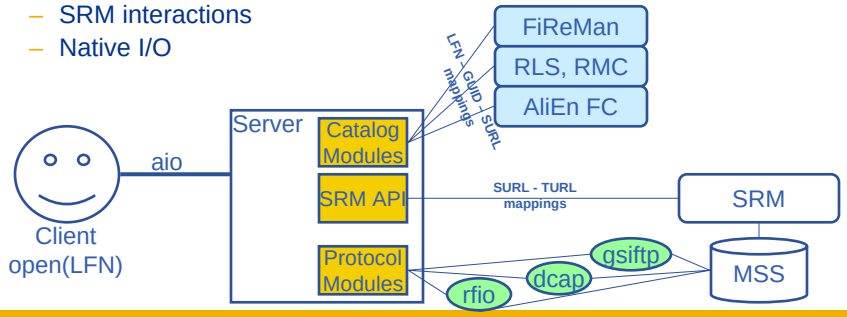
Access protocol used

Remote File Access

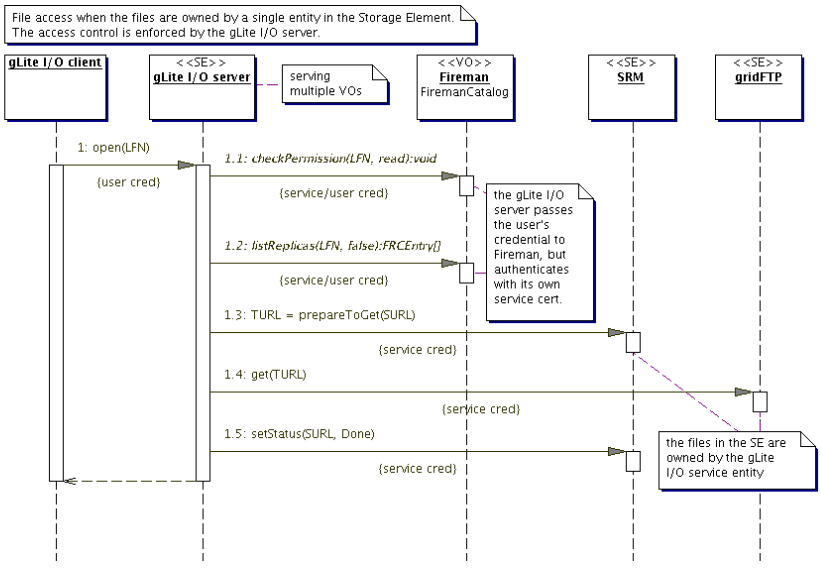
- **How can we access files stored on an SRM?**
- **The Catalogs allow to find the SURL of a file**
- **The SRM will translate the SURL into a TURL**
  - Not all SRMs support the same protocols for direct file access
  - E.g.: Castor – rfio, dCache – dcap
- **Need a common abstraction that hides these differences and also interacts with the catalogs**

gLite-I/O

- **Client only sees a simple API library and a Command Line Interface**
  - GUID or LFN can be used, i.e. open("/grid/myFile")
- **GSI Delegation to gLite I/O Server**
- **Server performs all operations on User's behalf**
  - Resolve LFN/GUID into SURL and TURL
- **Operations are pluggable**
  - Catalog interactions
  - SRM interactions
  - Native I/O



File Open



Basic gLite-I/O commands

Copy a local file to Storage Element

- `glite-put <local-file> <lfn:///lfn-name>`

Copy a file from Storage element

- `glite-get <lfn:///lfn-name> <local-file>`

Remove a file from Storage element

- `glite-rm <lfn:///lfn-name>`

  if the lfn is the last replica, file entry is removed from the catalog

C API provides POSIX-like interactions:

- `glite-open(…), glite-read(…), glite-write(…), glite-close(…)`

File Movement

- **The need for file movement**
  - Data is produced at one place but needs to be analyzed at many places
    - E.g. LHC experiments
  - Data is produced at many places – needs to be combined for analysis
    - E.g. Astronomy, weather forecast, …
  - Not all computation can take place where data is originally stored
    - Better exploit available computational and storage resources
  - Having multiple copies of a file increases the availability of data and reduces the risk of data loss
    - In case of unavailability of one storage resource others may hold the data as well

Replication vs. Copy

Enabling Grids for E-sciencE

- Replication is well known in distributed systems and important for Grids
- Replicas are identical and can be identified and located
- Replica location information can be regarded meta-data management
- Potentially, millions of files need to be registered and located
  - This is done in Replica Catalogs, e.g. the Globus RLS system, the LCG LFC, the gLite FiReMan (as discussed before)
- Replicas are managed copies of data

Data Transfer Requirements

Enabling Grids for E-sciencE

- Here we consider only file level granularity
  - No object streaming etc.
- Secure and efficient point-to-point file transfer over Wide Area Network links
- Needs to interact with existing Grid Security Infrastructure (GSI)
- Utilize network bandwidth
  - “Optimal” file transfer in close connection with network optimization

GridFTP

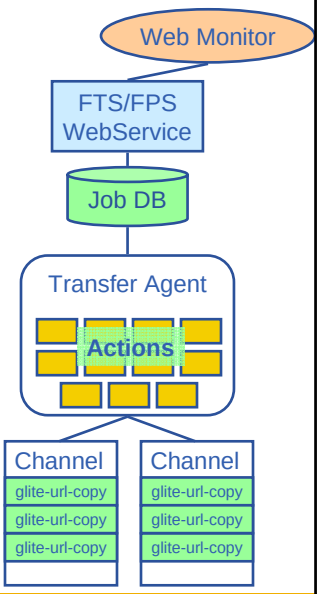
- **Data transfer and access protocol for** secure and efficient **data movement**
- **Standardized in the Global Grid Forum**
- extends **the standard** FTP **protocol**
  - Public-key-based Grid Security Infrastructure (GSI) or Kerberos support (both accessible via GSS-API
  - Third-party control of data transfer
  - Parallel data transfer
  - Striped data transfer Partial file transfer
  - Automatic negotiation of TCP buffer/window sizes
  - Support for reliable and restartable data transfer
  - Integrated instrumentation, for monitoring ongoing transfer performance

Reliable File Transfer

- **GridFTP is the basis of most transfer systems**
- **Retry functionality is limited**
  - Only retries in case of network problems; no possibility to recover from GridFTP a server crash
- **GridFTP handles one transfer at a time**
  - No possibility to do bulk optimization
  - No possibility to schedule parallel transfers
- **Need a layer on top of GridFTP that provides reliable scheduled file transfer**
  - E.g. Globus RFT, SRMCopy, gLite File Transfer & File Placement Services (FTS/FPS) and Data Scheduler (not yet available)

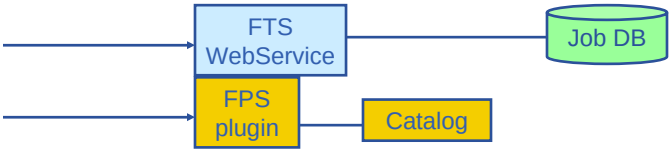
gLite Transfer services

- **File Transfer/Placement Service (FTS,FPS)**
  - Transfer Job Database
  - Exposes the Transfer Web Service Interface to which user clients talk (submit, cancel, status capability)
  - Has a Web Interface
  - Manages Catalog updates if necessary
- **Transfer Agent**
  - Basic Actions
    - Get transfer jobs from Transfer Job Database
    - Manages transfer over many channels
    - Monitors transfer status and updates Transfer Job Database
  - Extensible with user-defined custom actions
    - Retry Policy
- **Transfer Service (glite-url-copy)**
  - Actually performs transfer: SRM – SRM, gsiftp – SRM, gsiftp – gsiftp
  - Monitor capability, including gsiftp performance markers

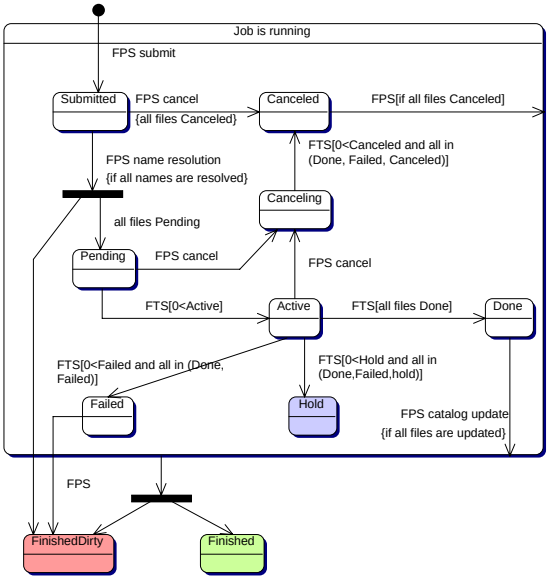


FTS vs. FPS

- **File Transfer Service (FTS)**
  - Acts only on SRM SURLs
  - `submit(source-SURL, destination-SURL)`
- **File Placement Service (FPS)**
  - A plug-in into the File Transfer that allows to act on logical file names (LFNs)
  - Interacts with replica catalogs (similar to gLite-I/O)
  - Registers replicas in the catalog
  - `submit(transferJobs) (transferJob = sourceLFN, destinationSE)`



Transfer Job States



Main File Transfer Service Commands

- **Submit a transfer job**
  - `glite-transfer-submit <source-surl> <dest-surl>`
    - This returns a `<job-id>`
- **Cancel a transfer job**
  - `glite-transfer-cancel <job-id>`
- **Status of a transfer job**
  - `glite-transfer-status <job-id>`
- **List transfer jobs**
  - `glite-transfer-list`

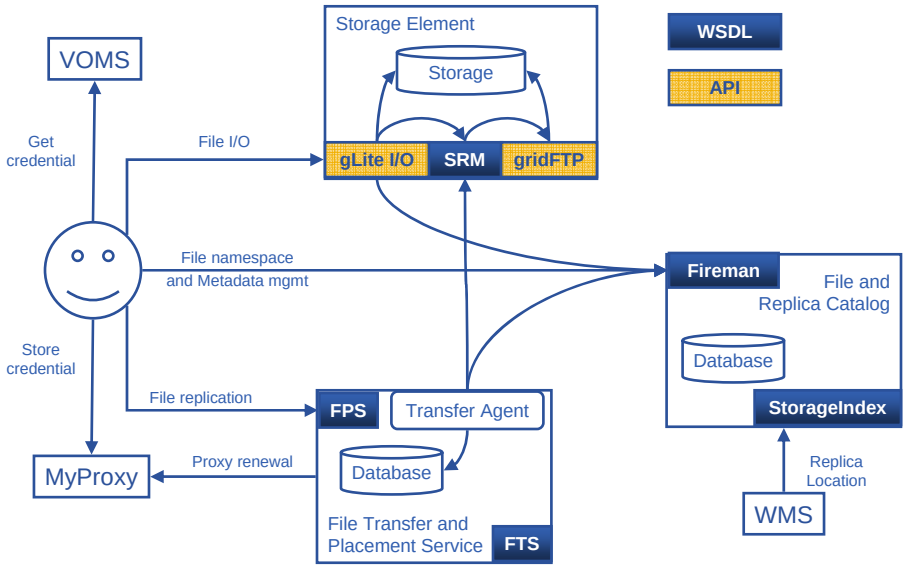
How to Replicate?

- **Using the File Transfer Service (FTS)**
  - Lookup source SURL in replica catalog
  - Initiate and monitor transfer
  - After successful transfer register new replica in the catalog
- **Using the File Placement Service (FPS)**
  - Initiate and monitor transfer
  - Plugin takes care of catalog interactions
- **FTS and FPS offer the same interface**
  - Difference only in input parameters
    - SURLs vs. LFNs
  - Different configuration
    - FPS requires catalog endpoint

Main File Placement Service commands

- **Basically the same as for the File Transfer Service:**
- **Submit a placement job**
  - `glite-placement-submit <dest_SE> <sourceLFN>`
    - This returns a `<job-id>`
- **Cancel a placement job**
  - `glite-placement-cancel <job-id>`
- **Status of a placement job**
  - `glite-placement-status <job-id>`
- **List placement jobs**
  - `glite-placement-list`

DM Summary



End-User Interactions (User Interface)

- **File Access**
  - glite-get, glite-put, glite-rm - on LFN and GUID
  - glite-IO API - C
- **Logical Namespace Management**
  - glite-catalog-* commands (like ls, create, rename, ..)
  - Fireman API - C, C++, Java, Perl
  - POOL File Catalog API (GliteCatalog implementation) – not exercised
- **Transfer and Replication**
  - glite-transfer-* commands (submit, status, cancel, ..)
  - FPS API - C, C++, Java, Perl

More Information

- **gLite homepage**
  - http://www.glite.org
- **DM subsystem documentation**
  - http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/doc.htm
- **FiReMan catalog user guide**
  - https://edms.cern.ch/file/570780/1/EGEE-TECH-570780-v1.0.pdf
- **gLite-I/O user guide**
  - https://edms.cern.ch/file/570771/1.1/EGEE-TECH-570771-v1.1.pdf
- **FTS/FPS user guide**
  - https://edms.cern.ch/file/591792/1/EGEE-TECH-591792-Transfer-CLI-v1.0.pdf