# Practical using EGEE middleware

## *Data Management*

www.eu-egee.org

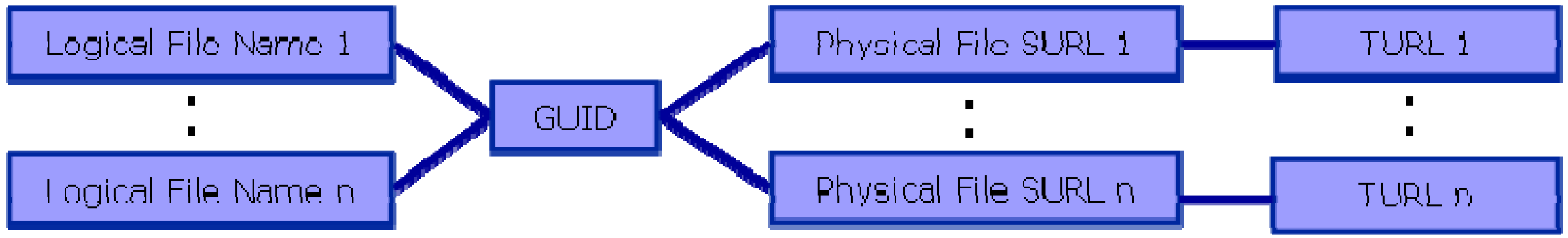
## Scope of data services

- **Files that are write-once, read-many**
  - If users edit files then
    - They manage the consequences!
    - Maybe just create a new filename!
  - No intention of providing a global file management system

- **3 service types for data**
  - Storage
  - Catalogs
  - Transfer

## Name conventions

- **Logical File Name (LFN)**
  - An alias created by a user to refer to some item of data, e.g. "lfn:cms/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://pcrd24.cern.ch/flatfiles/cms/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"



## Name conventions

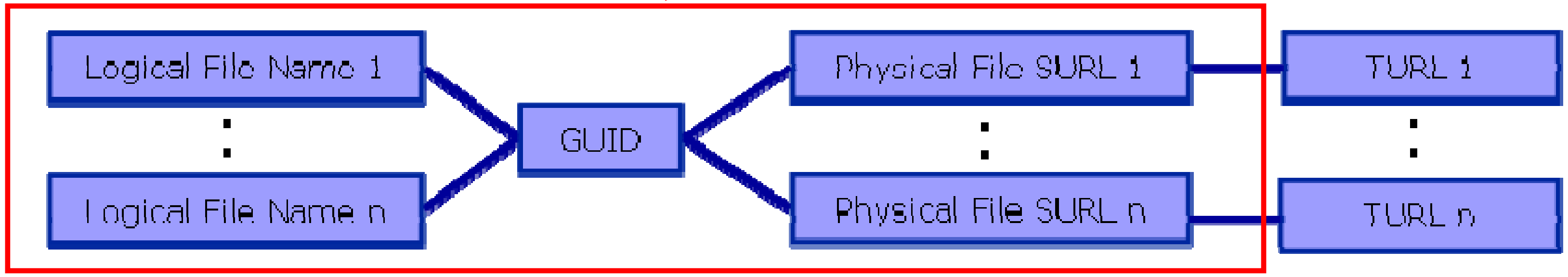
- **Users primarily access and manage files through "logical filenames"**

**LFC has a directory tree structure**

/grid/<VO_name>/ <you create it>

**LFC Namespace** — **Defined by the user**

- Mapping by the "LFC" catalogue server



## Two sets of commands

- **LFC = LCG File Catalogue**
    - LCG = LHC Compute Grid
    - LHC = Large Hadron Collider

  – Use LFC commands to interact with the catalogue only
    - To create catalogue directory
    - List files

  – Used by you and by lcg-utils

- **lcg-utils**

  – Couples catalogue operations with file management
    - Keeps SEs and catalogue in step!

## LFC basics

**LFC has a directory tree structure**

/grid/<VO_name>/ <you create it>

**LFC Namespace** **Defined by the user**

- **All members of a given VO have read-write permissions in their directory**

- **Commands look like UNIX with “lfc-” in front (often)**

## lcg-utils

- **File management functions**
  - copy files to/from/between SEs
  - Files can be replicated to be
    - “Close” to compute elements for efficiency
    - Resilient to SE failure (or upgrade)
- **Uses LFC to maintain coherence of catalogue**

## Storage Element

- **Provides**
  - Storage for files
  - Transfer protocol (gsiFTP) ~ GSI based FTP server
  - POSIX-like file access
    - Grid File Access Layer (**GFAL**)
      - *API interface*
      - *To read parts of files too big to copy*
- **Two types**
  - “Classic” SE
    - Massive storage system - disk or tape based
  - “SRM” SE
    - SE’s are virtualised by common interface: “SRMv1”
    - SRM = Storage Resource Manager
    - work in progress to migrate to SRMv2

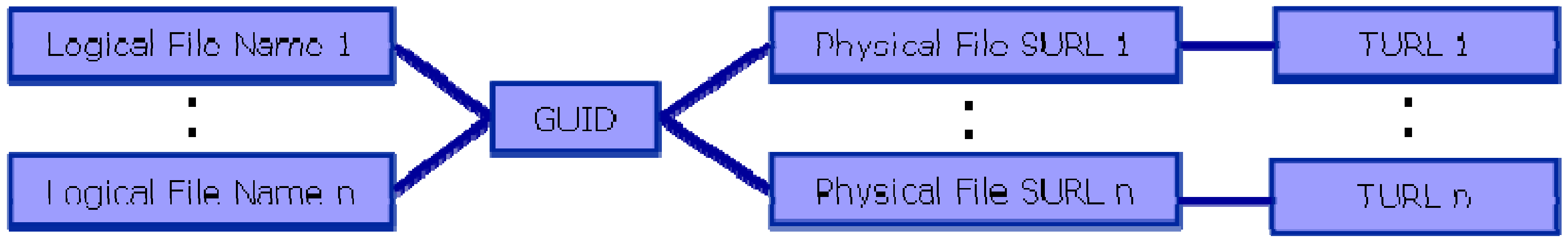
## We are about to…

- List directory
- Upload a file to an SE and register a logical name (lfn) in the catalog
- Create a duplicate in another SE
- List the replicas

- Create a second logical file name for a file
- Download a file from an SE to the UI

- And later we will: Use the lfn so that a job runs on a CE “close” to one of the SEs that holds a file

- Please go to the web  page for this practical



- Next slide for after practical

## LFC server

If a site acts as a central catalog for several VOs, it can either have:

- One LFC server, with one DB account containing the entries of all the supported VOs. You should then create one directory per VO.
- Several LFC servers, having each a DB account containing the entries for a given VO.

Both scenarios have consequences on the handling of database backups

- Minimum requirements (First scenario)
  - 2Ghz processor with 1GB of memory (not a hard requirement)
  - Dual power supply
  - Mirrored system disk

# LFC Catalog 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 |

## Summary of 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 |

## What we have not used in this tutorial….

- **GFAL functions to read blocks from files on SE's… can't always copy files to a worker node!)**

- **File transfer service**
  - Next slides:
    - Why have this?
    - What is it?

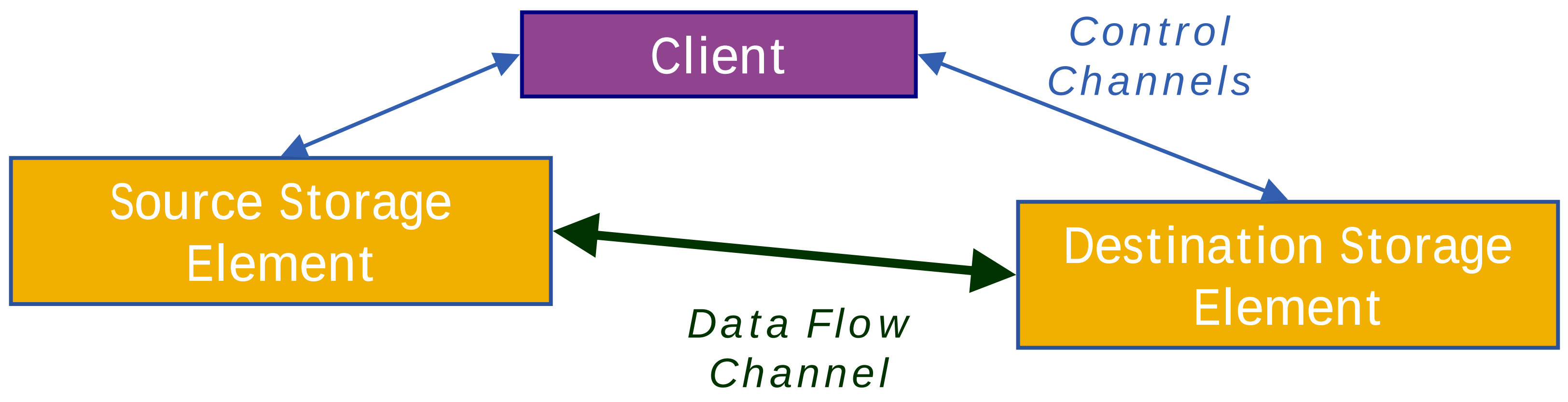
## Acknowledgement

- FTS slides taken from EUChinagrid presentation given by *Yaodong Cheng*
- *IHEP, Chinese Academy of Sciences*

- *EUChinaGRID tutorial*
- *Beijing, 15-16 June 2006*

- http://agenda.euchinagrid.org/fullAgenda.php?ida=a0621

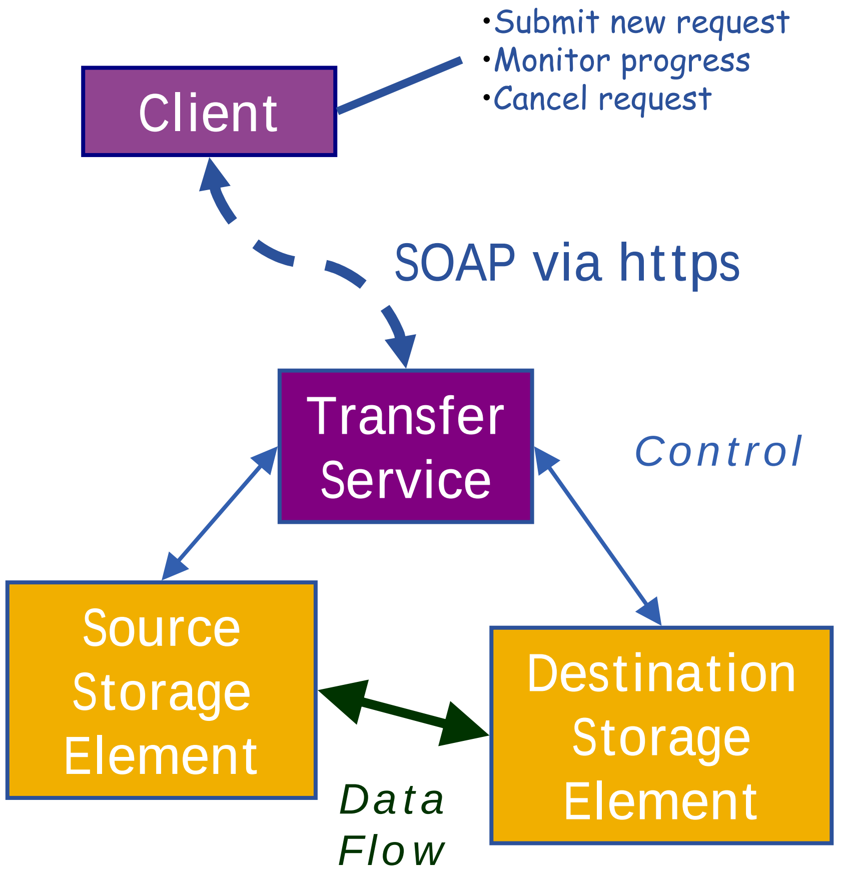
## Direct Client Controlled Data Movement



- **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

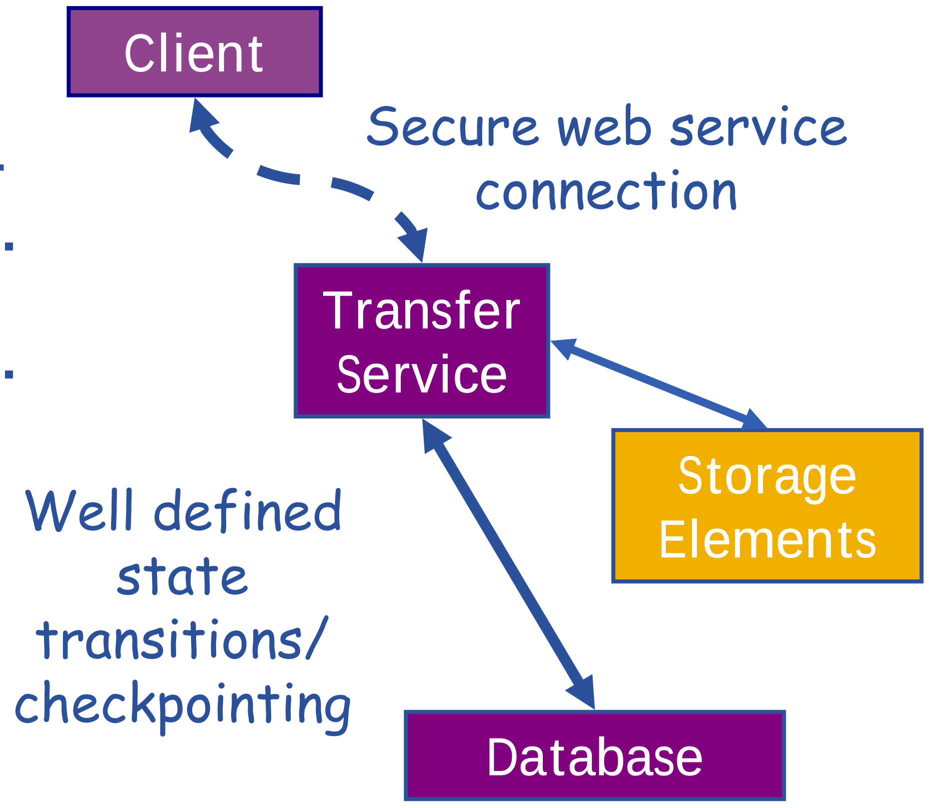
## Transfer Service

- **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



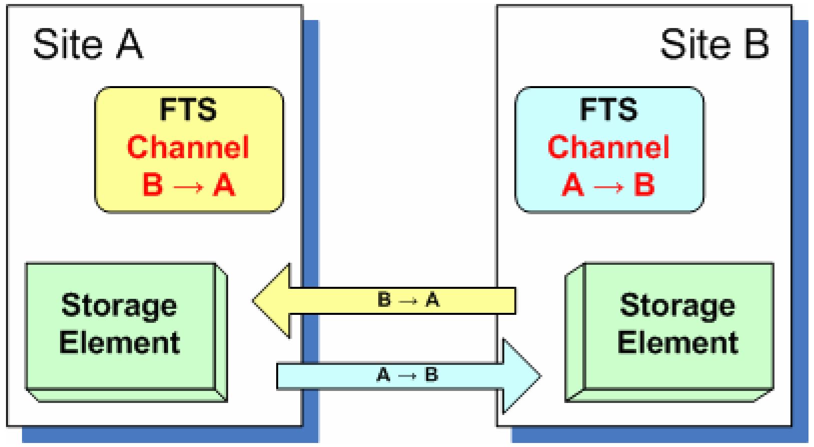
# Transfer Service Architecture

- **Clients submit jobs via SOAP over https.**
- **Jobs are lists of URLs in** srm:// **format. Some transfer parameters can be specified (streams, buffer sizes).**
- **Clients cannot subscribe for status changes, but can poll.**
- **C command line clients. C, Java and Perl APIs available.**
- **Backend databases supported: MySQL and Oracle.**
- **Web service runs in Tomcat5 container, agents runs as normal daemons.**


Client
Secure web service connection
Transfer Service
Storage Elements
Well defined state transitions/ checkpointing
Database

# 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



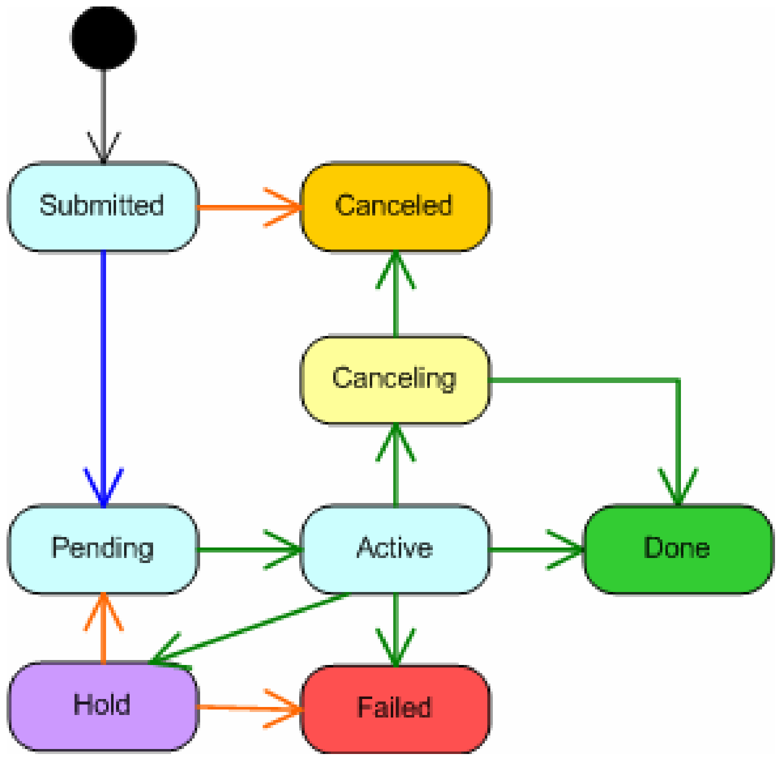
- Channels control certain transfer properties: transfer concurrency, gridftp streams.
- Channels can be controlled independently: started, stopped, drained.

## gLite FTS: Agents

- VO Agents
- Any job submitted to FTS is first handled by the VO agent
- VO agent authorises job and changes its state to "Pending"
- VO agents can perform other tasks – naturally these can be VO specific:
  - Scheduling
  - File catalog interaction

**Channel Agents**

- Transfers on channel are managed by the channel agent
- Channel agents can perform inter-VO scheduling



## Conclusion

- **FTS offer an important and useful service on the grid – a significant advance on client managed file transfers.**
- **FTS channel architecture offers very useful features to control transfers between sites or into a single site, though it may become overly complex in a grid without clear data flow patterns.**
  - The ability to control VO shares and transfer parameters on a channel is important for sites.
- **FTS agent architecture allows VOs to connect the transfer service closely with their own data management stacks, a useful feature for HEP experiments.**