Data Management from a User Perspective

Jan Just Keijser
Nikhef

Grid Tutorial, 13-14 November 2008

vl·e virtual laboratory for e·science

Outline

- Introduction
- Low Level Data Management
- LCG File Catalog (LFC)
- Data Management CLIs
- Command-line Examples
- Data Management APIs
- Programming Examples

Introduction (1)

Storage and Data Management can be (are) overwhelming at first

- **Storage systems:**
  - CASTOR
  - dCache
  - DPM
  - gridftp
  - LFC
  - SRB
  - SRM
- **Command sets:**
  - gridftp: globus-url-copy
  - LFC: lcg-* commands, lfc-* commands, edg-* commands
  - SRB: S* commands
  - SRM: srm* commands

Introduction (2)

Why so many systems, protocols and commands?

- **Historically grown**
- **Competing systems: there is no clear “winner” yet**
- **Different protocols serve different needs:**
  - management protocols vs transfer protocols
  - low-level vs. high-level access
  - some have metadata management (srb, srm)
  - some provide Hierarchical Storage Management (HSM), others disk-only
- **The transport protocol gsiftp:// is supported by nearly all products**
- **The management protocol SRM has a Web Services interface and is supported by CASTOR, DPM and dCache**

Protocol Mess, err, Mesh

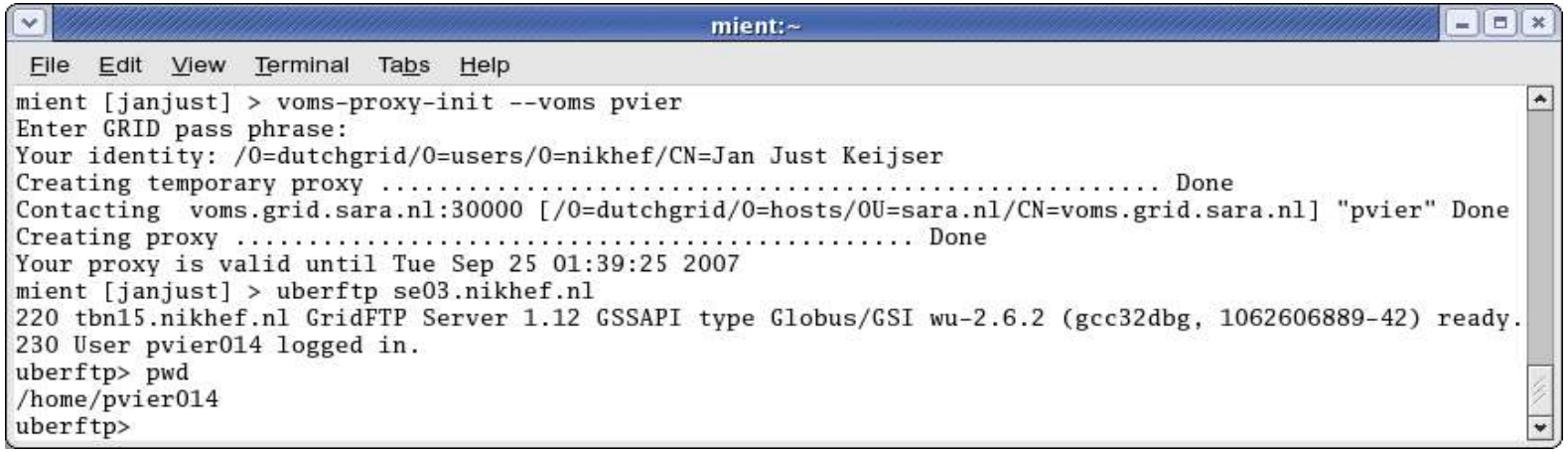
Product	*Management Protocol*	*Transport Protocol*
CASTOR	SRM v1	gsiftp, rfio
dCache	SRM v1, v2.1	gsiftp, dcap, gsi-dcap, xrootd
DPM	SRM v1, v2.2	gsiftp, gsi-rfio
SToRM	SRM v2	gsiftp, ?
GFAL (client)	SRM v1, v2	gsiftp, {rfio|gsi-rfio}, dcap, gsi-dcap

Notes

- SRM v1 and v2 are not compatible
- gsi-rfio and rfio cannot be used simultaneously
- gsiftp does not provide a full POSIX interface (seek, trunc), rfio and gsi-rfio do

Low Level Data Management

- **gsiftp/GridFTP (all SEs)**
  - globus-url-copy file:///home/janjust/file \
    gsiftp://srm.grid.sara.nl/pnfs/grid.sara.nl/data/dteam/file
  - Third party transfer
    - globus-url-copy gsiftp://hostA/pathA gsiftp://hostB/pathB
  - Also edg-gridftp-ls, edg-gridftp-rm, edg-gridftp-mkdir etc.
  - Uberftp
    - Interactive gridftp client
    - ftp commands
    - Gsi authentication



Low Level Data Management

- **dCache: gsi-dcap**
  - dccp -p 20000:25000 /tmp/file \
    gsidcap://srm.grid.sara.nl:22128/pnfs/grid.sara.nl/data/dteam/file
  - 20000:25000 is derived from GLOBUS_TCP_PORT_RANGE environment variable
- **Secure rfio**
  - rfcp /path/myfile \
    t2se01.physics.ox.ac.uk:/dpm/physics.ox.ac.uk/home/dteam/file
- **SRM: srmcp**
  - Srmcp file:////tmp/file \
    srm://srm.grid.sara.nl:8443//pnfs/grid.sara.nl/data/dteam/file
  - Count the slashes!

LCG File Catalog (LFC)

- **Provides:**
  - Command line tools with administrative functionality
  - Hierarchical unix-like namespace and namespace operations for LFNs
    - lfn:/grid/<vo name>/mydir/myfile
    - lfc-mkdir, lfc-chmod
  - Integrated GSI Authentication + Authorization
  - Access Control Lists (Unix Permissions and POSIX ACLs)
  - Checksums
  - Sessions (multiple operations inside a single transaction)
  - Bulk operations (inside transactions)
  - User exposed transaction C/C++ API (+ auto rollback on failure)
    - Python wrapper provided (python module lfc)

LFC Interfaces

- **Integration with GFAL and lcg_utils APIs**
  - ➔ lcg-utils/GFAL access the catalog in a transparent way
- **Integration with the WMS**
  - The RB can locate Grid files: allows for data based match-making
  - Jdl file:
    - InputData = "lfn:/grid/tutor/MyFile";

Data Management CLIs

These two packages provide (nearly) all the functionality needed by most grid users:

- **LFC_client:**
  - lfc-* commands
  - mostly for manipulating *directories*
- **lcg_utils:**
  - lcg-* commands
  - Transparent interaction with file catalogs and storage interfaces when needed
  - Abstraction from technology of specific implementations
  - mostly for manipulating *files*

CLI: *lfc-** Commands

Summary of LFC 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

CLI: *lcg-** Commands

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

lcg_utils commands: 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

Preparation for using lfc-* and lcg-* commands

- **Define the server hostname**
  - The LFC server must be published in the BDII ($LFC_GFAL_INFOSYS)
  - Use environment variable: $LFC_*HOST*=<*lfc_server_hostname*>
- **Use the 'lcg-infosites' command to find out the name of the current LFC and SE:**

  > ***lcg-infosites --vo tutor lfc***

  lfc.grid.sara.nl

  > ***lcg-infosites --vo tutor se***

  gb-se-ams.els.sara.nl se.grid.rug.nl srm.grid.sara.nl
- **Remember**
  - lfc-* commands are mostly for manipulating *directories*
  - lcg-* commands are mostly for manipulating *files*

Listing the entries of an 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>**

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

```
> lfc-ls /grid/tutor/me
> export LFC_HOME=/grid/tutor
> lfc-ls -l me
> lfc-ls -l -R /grid
```

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

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.
- Example:

  > ***lfc-mkdir /grid/tutor/me***

  You can check the directory with:

  > ***lfc-ls -l /grid/tutor/me***

  drwxr-xrwx 0 19122 1077 0 Jun 14 11:36 demo

lcg-cr: copy and register a file

lcg-cr [-d dest_file | dest_host] [-l lfn] [--vo vo] src_file

Where *lfn* is the Logical File Name that can include an LFC pathname created with *lfc-mkdir*

- Example:

  > lcg-cr --vo tutor -l me/test -d srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/tutor/test/some_file_name file:`pwd`/test

  guid:7b4efaef-bb0f-42a3-bb6f-bbe35080d105

- List our file by looking at the SFN of the replica's:

  > lcg-lr --vo tutor lfn:me/test

  sfn://srm.grid.sara.nl//*pnfs/grid.sara.nl/data/tutor/test/some_file_name*

- List the files in our directory in the LFC:

  > lfc-ls -l me

  -rw-rw-r-- 1 30010 2024 114 Sep 18 10:33 test

lfc-ln: creating a symbolic link

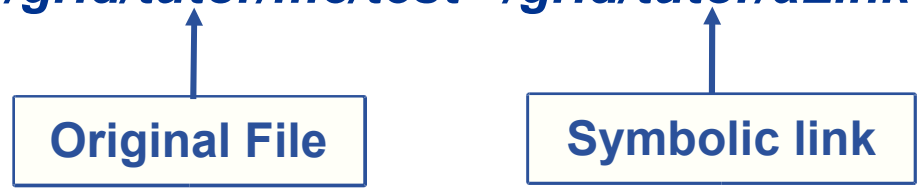
lfc-ln -s file linkname

lfc-ln -s directory linkname

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

- Example:

  > lfc-ln -s /grid/tutor/me/test /grid/tutor/aLink



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

  > lfc-ls -l

  lrwxrwxrwx 1 30010 2024 0 Sep 18 10:38 aLink ->
  /grid/tutor/me/test

lfc-*comment: adding/deleting metadata information

lfc-setcomment *path comment*

Add/replace a *comment* associated with a *path (*i.e. file or directory)

lfc-delcomment *path*

Delete a comment previously added

- This is the only metadata (one field) supported by the catalog

- Example:

  > ***lfc-setcomment me/test "nice file"***

- Let's see what happened:

  > ***lfc-ls --comment /grid/tutor/me/test***

  /grid/tutor/me/test nice file

lfc-rm, lcg-del: deleting the file

lfc-rm

Remove a file/link/directory **only** from the catalog

lcg-del

Remove a file from the SE(s) and the lfns/links from the catalog

Examples

- Delete all replicas:

  > ***lcg-del -a --vo tutor ***

  guid:8e413879-7cb3-4260-af9f-6964392da7e8

- Delete only one replica:

  > ***lcg-del -a --vo tutor -s srm.grid.sara.nl ***

  guid:8e413879-7cb3-4260-af9f-6964392da7e8

srmcp: an SRM Example

- **Note**: the srm* tools do not use the File Catalog. Hence they do not know of logical file names nor can they hide the storage hierarchy (SFN) from the user!

  srmcp source-url destination-url

  where the source-url and destination-url can be of the form

  – *file:////some-path/some-filename*

  – *srm://server:8443/pnfs/grid.sara.nl/data/<VO-name>/<filename>*

  Count the slashes ('/') ! This is because ***srmcp*** is a Java application.

Example

- Copy a file to the SRM server

  > ***srmcp*** *file:////home/janjust/file* ****
  srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/tutor/file

Data Management APIs

APIs for most client commands are available:

- **Grid File Access Library (GFAL): API**
  - Adds POSIX-like file I/O and explicit catalog interaction functionality
  - Useful/unavoidable when accessing files that are larger than the available scratch space on a grid worker node
  - Still provides the abstraction and transparency of lcg_utils
  - C/C++ interface
  - Python wrapper interface
- **LFC_client: lfc_* API calls**
  - Interaction with file catalogs
- **lcg_utils: lcg_* API calls**
  - Transparent interaction with file catalogs and storage interfaces when needed

LFC_client C API (low-level, 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

lcg-utils C API

lcg_cp
lcg_cr
lcg_del
lcg_rep
lcg_sd
lcg_aa
lcg_gt
lcg_lr
lcg_ra
lcg_rf
lcg_uf
lcg_la
lcg_lg

lcg-utils Python Example

```
#!/usr/bin/python
"""
# Copy a file to Storage element and register in the LFC.
"""
import sys
import lcg_util
src='file:/etc/hosts'
dest='srm.grid.sara.nl'
guid="212fa800-9d65-11da-a746-0800200c9b13"
lfn='/grid/tutor/me/testfile'
vo='tutor'
relativepath='me/testfile'
nstreams=1
config=''
insecure=0
verbose=1
actual_guid=""
for i in range(0,37):
      actual_guid=actual_guid + " "
output= lcg_util.lcg_cr(src,dest,guid,lfn,vo,relativepath,nstreams,'',insecure,verbose,actual_guid)
print "teststatus: ", output
print "actual_guid: ", actual_guid
```

Still Interested?

Questions?