Enabling Grids for E-science in Europe

www.eu-egee.org

First Latinamerican Grid Workshop
Mérida, Venezuela, 15-20 November 2004

Storage Services

Patricia Méndez Lorenzo
patricia.mendez@cern.ch

LCG Experiment Integration and Support
CERN IT/GD-EIS

EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

Enabling Grids for E-science in Europe

- Introduction
- SE types in LCG-2
- SRM (Storage Resource Manager Interface)
- GFAL (Grid File Access Library)
- Hands-on session

Introduction

Enabling Grids for E-science in Europe

- In the past distributed computing was quite focused on the computational aspect
  - Supporting large distributed computational tasks
  - Managing the sharing of the network bandwidth
- Today data access is becoming the main bottleneck
  - Huge amount of data (~PB)
  - Distributed in many sites
  - Not all of them can be replicated
- What a Storage Element should provide
  - The SE resources have to provide a good access and services to storage spaces
  - Middleware has to be able to manage different storage systems uniformly and transparently for the user

SE Components in LCG-2

Enabling Grids for E-science in Europe

1. The Data can be accessed using the following protocols:

- ♠ File tranfer: GridFTP
- ♠ File I/O: RFIO, gsidcap, *file*

} Protocols

2. SE types supported by LCG-2:

- ♠ Classic SE (disk server): GridFTP + RFIO
- ♠ SRM-dCache disk pools: GridFTP + gsidcap
- ♠ SRM-Castor Mass Storage Systems: GridFTP + RFIO

} Protocols

3. User demands for reliable storage and storage management capabilities:

- ♠ Transparent access to files (migration to/from disk pool)
- ♠ File pinning
- ♠ Space reservation
- ♠ File status notification
- ♠ Security
- ♠ Life time management

Concepts

Enabling Grids for E-science in Europe

Storage Resource

¤ Every storage system able to provide storage to different clients

Disk pool managers: dCache and Castor

¤ Middleware modules able to manage distributed storage servers in a centralized way

- Disk pools
- Tapes system

¤ Physical disks or arrays are combined into a common file system

SRM (Storage Resource Manager)

¤ Middleware module designed to optimize the use of the storage resources

¤ It interacts with the operative systems and MSS (Mass Storage Systems) to make file archiving, staging and pinning transparent to users

Transparent for the user: He has not to know the names of the disk or tapes behind (Load Balancing System)

¤ Supervises and manages transfer using specific protocols (GridFTP)

¤ Designed to relocate and reuse dynamically the space

SRM in an example

Enabling Grids for E-science in Europe

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 Nikhef
in a classic SE

They are at Fermilab
In a SRM served disk

SRM in an example

Enabling Grids for E-science in Europe

This is dCache speaking…
Sure you can connect!, but
To use a special protocol t

We are a classic SE, well no
You need another paramete
to do with dCache

Castor, yes?... No sorry, nc
Do with classic CE ….

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

Not only protocols… SRM makes more

Enabling Grids for E-science in Europe

What she has to do:

1. To find a way to determine the file she needs, for example:
   a) Looking into file catalogues
   b) Data bases
2. She will obtain some LFNs
3. Then she has to find the physical locations PFN
   Files can be replicated in different sites

Once the files are found we have to access them

→ The middleware will determine the best file to be accessed by the application

1. Choose the site where to take the files from
2. Perform a planning based on a certain strategy
   a) Move the application close to the file
   b) Move the file(s) close to the computing resource
   c) Move application and data somewhere else

   (Execute file pinning)

   (Perform space allocation and Local pinning of the file at destination)
3. Execution of the plan

SRM Use Advantages

Enabling Grids for E-science in Europe

- Quota managements, freeing space if needed, space allocation, file pinning… done by SRM

- It makes the user's life easier: user has not to try many times before the storage system is free. SRM queues the request and informs the user on the status of the request

- It is resilient to network failures

- Isolate clients from errors in the storage systems

- It is able to work in "streaming mode" in the case the user needs a huge number of files

GFAL: Grid File Access Library

Enabling Grids for E-science in Europe

Interactions with SE require some components:

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

GFAL:

→ 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

Supported protocols:

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

Some remarks on RFIO

Enabling Grids for E-science in Europe

RFIO requires a specific format in SURLs and TURLs

For classic SEs: double slash after the hostname

```
sfn://lxb0710.cern.ch//flatfiles/SE00/dteam/my_file
rfio://lxb0710.cern.ch//flatfiles/SE00/dteam/my_file
```

For Castor backends: the hostname is included in the path

```
sfn:///castor/cern.ch/grid/dteam/my_file
rfio:///castor/cern.ch/grid/dteam/my_file
```

If the catalogs contain incorrect SURLs, programs using LFNs and GUIDs with GFAL and RFIO will fail ➔ solved with SRMs and LFC

Programs using the RFIO API (e.g., under GFAL)...

...will not work from the UI ➔ Have to be executed in a WN

...will not work from a WN to access SE in a different site

The reason is that RFIO is not GSI-enabled and requires exact mapping from user's uids in the WN and the SEs.

gfal-gridftp commands

Enabling Grids for E-science in Europe

gfal_stat(…)	Checks if file/dir exists in the SE
gfal_opendir(…)	List a directory in the SE
gfal_mkdir(…)	Creates a directory in the SE
gfal_rename(…)	Renames a file in the SE
gfal_unlink(…)	Removes a file in the SE
gfal_rmdir(…)	Removes a directory in the SE

GFAL: Catalog API

Enabling Grids for E-science in Europe

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

Enabling Grids for E-science in Europe

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)

int **srm_get** (int nbfiles, char **surls, int nbprotocols, char **protocols, int *reqid, char **token, struct srm_filestatus **filestatuses)

int **srm_getstatus** (int nbfiles, char **surls, int reqid, char *token, struct srm_filestatus **filestatuses)

GFAL: File I/O API (I)

Enabling Grids for E-science in Europe

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);

GFAL: File I/O API (II)

Enabling Grids for E-science in Europe

int **gfal_closedir** (DIR *dirp);

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

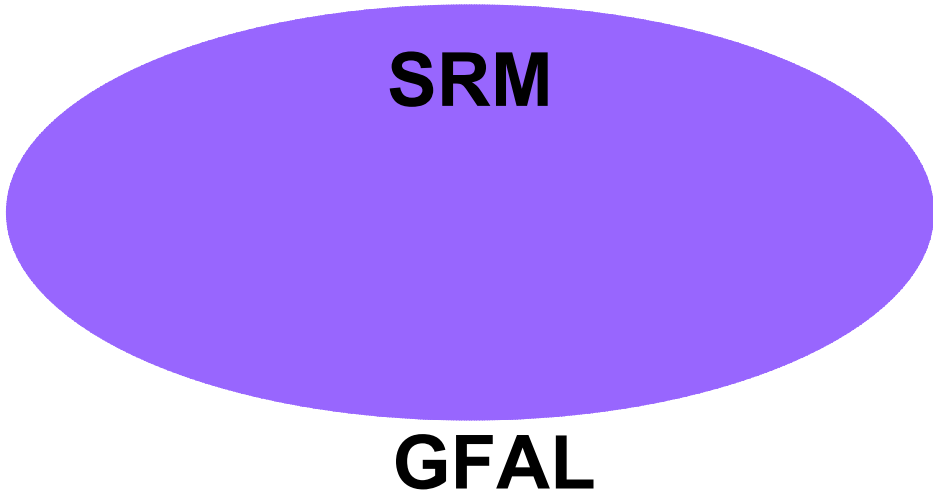
DIR ***gfal_opendir** (const char *dirname);

struct dirent ***gfal_readdir** (DIR *dirp);

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

GFAL, SRM and MSS

1. GFAL will be the highest level interface
2. It will take of SRM and Replica Managers and protocols (transparent for the user)
3. SRM will take care of the handling with MSS (not visible for the user)



Hands-on Session

Enabling Grids for E-science in Europe

Exercises:

1. Check the syntax of the GFAL API in the manpages and in $LCG_LOCATION/include/gfal_api.h

2. Submit a job that creates a file, writes the date in it every second for 10 seconds and registers the file

3. Submit a job that opens a file already registered in Grid and reads only the first 20 lines. Retrieve them with the std. output of the job

General Remarks:

- Use skeletons and solutions included in GFAL_exercises folder
- Remember to use double slash in the SURLs to be used with RFIO under GFAL
- The programs will not work directly from the UI. You have to send them in a job to be executed in a WN. They will not be able to access SEs in different sites either

Enabling Grids for E-science in Europe

Remarks (I):

- You need to include the “gfal.h” library as an “extern C”
- You should open the file and read characters until the 20th line is reached
- Hint: read one char a time and compare it to the newline character

Enabling Grids for E-science in Europe

Remarks (II):

- First, create an application that writes the file. Use the skeleton of “gfal_time_write.cpp” for the timing tasks (need to include “time.h” and “unistd.h”). As destination filename, a SURL has to be used.

- Second, create an application that registers the created file (providing its SURL), using register_pfn and create_alias (since register_alias is used for extra LFNs). Use functions from “uuis/uuid.h” for the manul GUID generation, as shown in the skeleton of “gfal_time_reg.cpp”.

  Hint: You may use gfal_stat, to check the file existence and its size.

- Now, create an application that performs this registration using lcg_utils. See that this other way is simpler. “gfal_time_reg_lcg.cpp

- Integrate both creation/writing and registration in a single program. See “gfal_time_complete.cpp”.