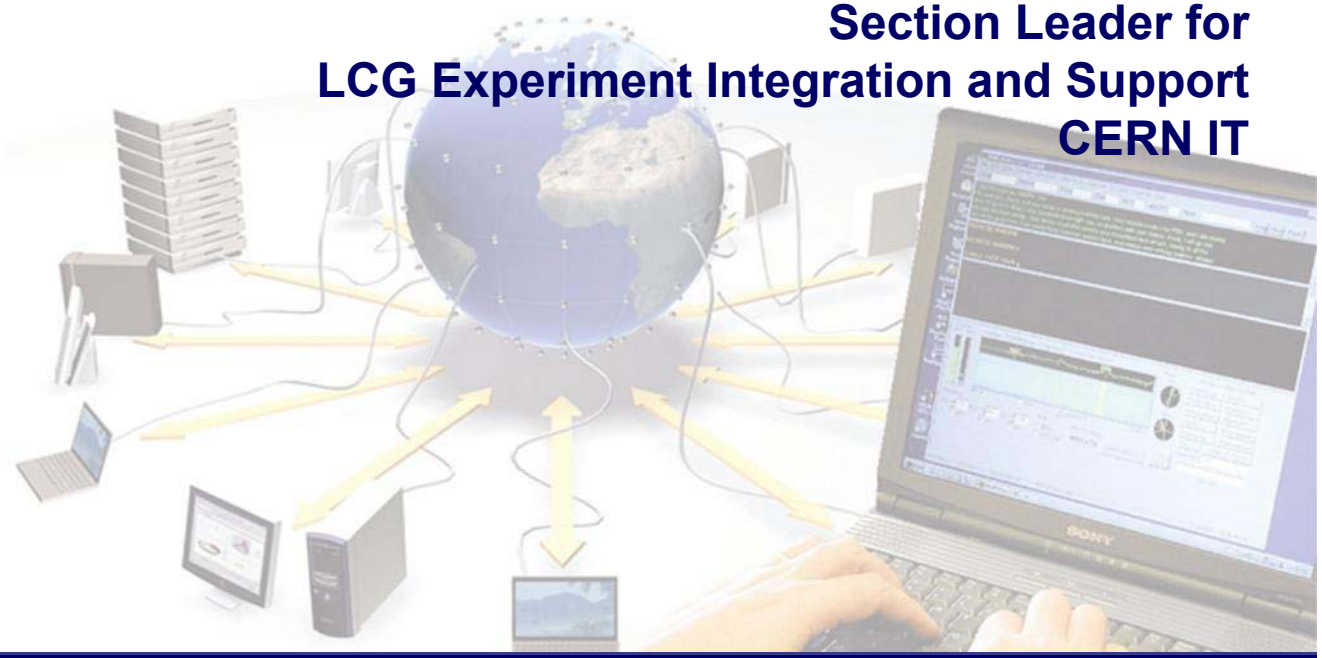
Enabling Grids for E-science in Europe

www.eu-egee.org

Biomed Application Developer's Course
6th October 2004

Grid Data Management

Flavia Donno
Section Leader for
LCG Experiment Integration and Support
CERN IT



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

Contents

Enabling Grids for E-science in Europe

- Problem Statement
- Data Management tools overview
- Intro to Basic DM tools
- Walkthrough of several Grid DM scenarios
- Available APIs
- lcg_util APIs usage

Problem Statement: How to connect User/Programs/Data?

Enabling Grids for E-science in Europe

- User
  - logged in to a Grid “User Interface” machine, or
  - Logged in to a “desktop” machine
- Programs
  - On desktop
  - On UI
  - On Grid machines
- Data
  - May need to supply (Grid or non-Grid) data to applications
  - Applications may generate data, need to put it somewhere safe
  - How do you retrieve it from somewhere safe?

Grid Data Management Tools Overview

Enabling Grids for E-science in Europe

- Replica Location Service (RLS) keeps track of where various copies of “grid datasets” (files) are located
- LCG Replica Management Tools (RM) are the primary user tools
  - Data Transfer mostly uses gridftp behind the scenes
    - Like good old FTP except uses grid auth(oriza)(entica)tion
    - No passwords!
    - Can also use multiple streams for faster transfer
  - RM handles interaction with gridftp & RLS to ease instantiation, registration, and replication of grid datasets
- Resource Broker
  - can send (small amounts of) data to/from jobs
  - can use RLS to find your data, and send your job to it, if your data is in the RLS and you tell RB about it

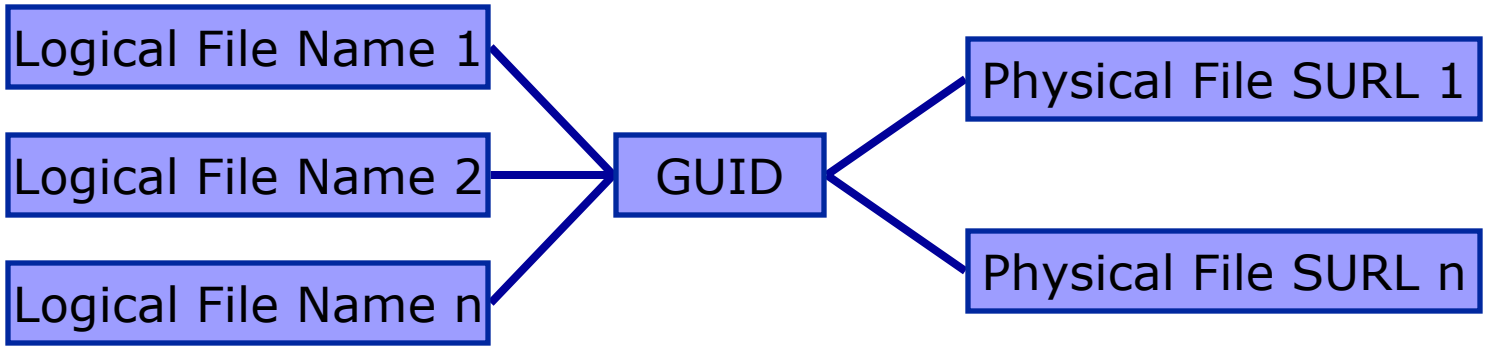
Grid Data Management Tools

Enabling Grids for E-science in Europe

- Tools for
  - Locating data
  - Copying data
  - Managing and replicating data
  - Meta Data management
- On GILDA/LCG-2 you have
  - EDG Replica Location Service (RLS)
    - Local Replica Catalog
    - Replica MetaData Catalog
  - globus-url-copy (GridFTP)
  - LCG Replica Manager
  - Grid File Access Library
  - Various client protocol libraries rfio, dcap, etc.

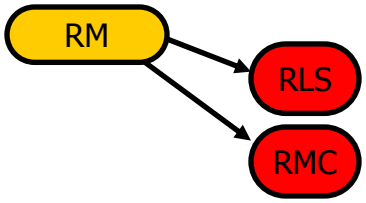
Naming Conventions

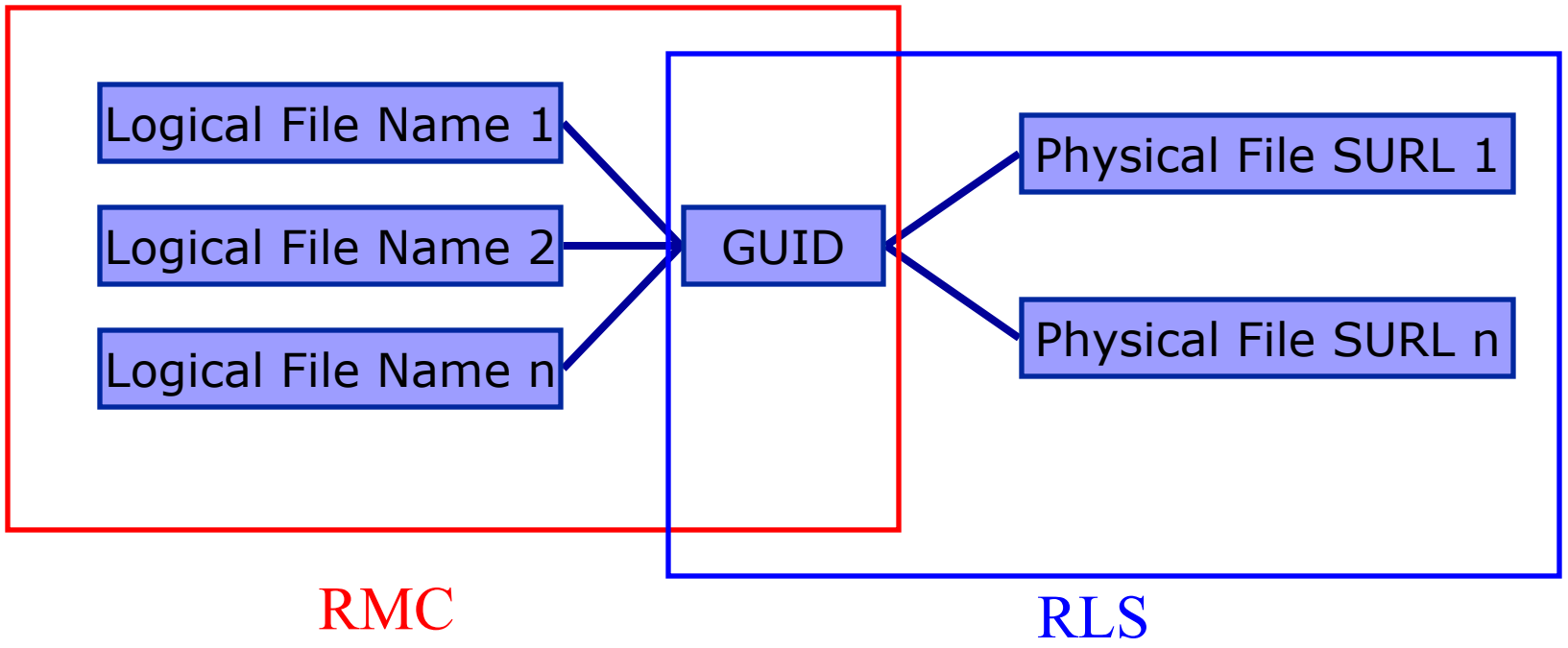
- Logical File Name (LFN)
  - An alias created by a user to refer to some item of data e.g. “lfn:cms/20030203/run2/track1”
- Site URL (SURL) (or Physical File Name (PFN))
  - The location of an actual piece of data on a storage system e.g. “srm://pcrd24.cern.ch/flatfiles/cms/output10_1” “sfn://lxshare0209.cern.ch/data/alice/ntuples.dat
- Globally Unique Identifier (GUID)
  - A non-human readable unique identifier for an item of data e.g. “guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6”



RLS: LRC and RMC

- RMC:
  - Stores LFN-GUID mappings
- RLS:
  - Stores GUID-SURL mappings





Basic RM Commands (I)

Enabling Grids for E-science in Europe

- Putting data on the Grid
  - Put the file `/home/flavia/fd.awk` (on the local computer) onto the *storage element* `(find an SE on GILDA!)` and register it with the *logical file name* `flavia.example`
  - `lcg-cr --vo gilda -d <did you find GILDA SE?> -l lfn:flavia.example file:/home/flavia/fd.awk`

    `> guid:6ac491ea-684c-11d8-8f12-9c97cebf582a`
- Storage Element – grid-aware computer with support for data storage
- Logical File Name – symbolic file name with which you can refer to a grid file without specifying actual location
- Above command returned a “guid”:
  - `guid:6ac491ea-684c-11d8-8f12-9c97cebf582a`
- Guids are unique, LFNs are not!!

Basic RM Commands (II)

Enabling Grids for E-science in Europe

- Finding your data: the listReplicas (lr) method
  - `lcg-rep --vo gilda lfn:flavia.example  # via LFN`

```
>sfn://lxb0707.cern.ch/flatfiles/SE00/dteam/generated/2004
  -07-09/file79aee616-6cd7-4b75- 8848-f09110ade178 >
  sfn://lxb0710.cern.ch/flatfiles/SE00/dteam/generated/200
  4-07-08/file0dcabb46-2214-4db8- 9ee8-2930de1a6bef
```

  - `lcg-rep --vo gilda \  # via GUID`
    `guid:76373236-b4c7-11d8-bb5e-eba42b5000d0`
  - `sfn://gridkap02.fzk.de/grid/fzk.de/mounts/nfs/data/lcg1/SE00/lhcb/generated/2004-06-02/file7115df45-b4c7-11d8-bb5e-eba42b5000d0`
- "replicas" because someone (or some application) may make a copy on a different storage element (SE) – the LFN and GUID refer to *all* copies

Basic RM Commands (III)

- Finding information about RLS or “DMS”
  - How did we know that `<SE on GILDA>` was a storage element?
  - `lcg-infosites --vo gilda all nousedspace`

```
LRC endpoint for dteam: http://rlsdteam.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog
/services/edg-local-replica-catalog
RMC endpoint for dteam: http://rlsdteam.cern.ch:7777/dteam/v2.2/edg-replica-metadata-cata
log/services/edg-replica-metadata-catalog
****************************************************************
These are the related data for gilda: (in terms of CPUs)
****************************************************************
#CPU Free Total Jobs Running Waiting ComputingElement
------------------------------------------------------------
6      6      0     0     0 ce01.lip.pt:2119/jobmanager-lcgpbs-dteam
9      9      0     0     0 lcg-ce.ecm.ub.es:2119/jobmanager-pbs-long
****************************************************************
These are the related data for dteam: (in terms of SE)
****************************************************************
Avail Space(Kb) SEs
----------------------------------------------------------
72982236 se01.lip.pt
7549980 se00.inta.es
```

  - Lots more information printed
  - Locations of RLS components
  - Locations of all computing resources

Common Grid Data Management Tasks

Enabling Grids for E-science in Europe

- Dealing with Data Your Job Generates
  - Getting the data back to your desktop
  - Putting the data “on the Grid”
- Getting Data to your Job
  - Submitting data along with your job
  - Putting your data onto the Grid (from outside)
  - Sending your Grid job to your Grid data
- Moving Data on the Grid
- How to find your data if you don’t remember where you put it
- Examples and documentation can be found:
  https://edms.cern.ch/file/454439//LCG-2-UserGuide.html

Hand-on time!

Enabling Grids for E-science in Europe

- Check lcg-utils commands described in LCG-2 User Guide

- What is a Storage Element ?

- What are the available SEs on GILDA ?
- Bring your on private file on GILDA
- Replicate it
- Interrogate the RLS on its location
- Remove all physical instances of the file from GILDA GRID

Grid Application -> Data on your desktop

Enabling Grids for E-science in Europe

- You can set up your job for “data pickup”
  - Job generates data in current working directory on WN
  - At job end, the data files are placed in temp storage at RB
  - You get them back via “edg-job-get-output”
- Key items:
  - You need to know names of files you want to get back
  - OutputSandbox = {“higgs.root",“graviton.HDF"};
  - not intended for large files (> hundred MB) – storage limitation on Resource Broker machine
- Example: output-sandbox.{jdl,sh}

Grid Application -> data “on Grid”

Enabling Grids for E-science in Europe

- Your program generates data to some local file
- Program has to know (or be able to figure out) what the local file name is
- Program uses the lcg-utils commands to
  - Put the data onto Grid storage
  - Register the data as a Grid dataset
- A couple optional, but useful, extras:
  - On which SE should the data be stored (or even in which directory on which SE!). Default: “local” SE
  - A logical file name. Default: no LFN!

Reminders

Enabling Grids for E-science in Europe

- Reminders:
  - If you want a specific SE, find it using the

    lcg-infosites --vo gilda closeSE --is lxn1178.cern.ch

    command.
  - Put the file on grid storage (in RLS, on SE) using the

    lcg-cr –vo <yourvo>

    command.

Alternate Method: Let WMS do it

- OutputData JDL attribute specifies where files should go
  - If no LFN specified WP2 selects one
  - If no SE is specified, the close SE is chosen
- At the end of the job the files are moved from WN and registered
- File with result of this operation is created and added to the sandbox : DSUpload_<unique jobstring>.out
-
```
OutputData = { [
    OutputFile =  "toto.out" ;
    StorageElement =  "adc0021.cern.ch" ;
    LogicalFileName =  "lfn:theBestTotoEver" ; ],
  [
    OutputFile =  "toto2.out" ;
    StorageElement =  "adc0021.cern.ch" ;
    LogicalFileName =  "lfn:theBestTotoEver2" ; ]
};
```

Submitting Data Along With Your Job

Enabling Grids for E-science in Europe

- This is fairly easy: use the Input Sandbox
- Careful – not a sandbox in the javascript sense
- Careful 2 – not meant for large (multi-megabyte) transfers
- InputSandbox = {“input-ntuple.root"};
- Example files: inp-sbox.{jdl,sh}

Moving Data Onto Grid from Outside

Enabling Grids for E-science in Europe

- Putting data on the Grid
  - Put the file `/home/flavia/fd.awk` (on the local computer) onto the *storage element* `gridkap02.fzk.de` and register it with the *logical file name* `flavia.example`
  - ```
    lcg-cr --vo gilda file:/home/flaiva/fd.awk \
          -l lfn:flavia.example -d gridkap02.fzk.de
    ```
- Above command returned a “guid”:
  - `guid:76373236-b4c7-11d8-bb5e-eba42b5000d0`
- Guids are unique, LFNs are not!!
- Try it with different SEs or no SE, or even with no LFN
    ```

Having Grid Send Job to Your Data

Enabling Grids for E-science in Europe

- Need to have data “on the Grid” == listed in RLS
- Tell your job (JDL) about the grid data:
  - InputData = “lfn:myfile.dat”
- Resource Broker puts info about data matching in “brokerinfo” file on remote execution node
- In your job execution script, use the “edg-brokerinfo” command & edg-rm commands to get job-local copy
- Example files: find-data.{jdl,sh}

Moving Data Around

Enabling Grids for E-science in Europe

- `edg-rep --vo gilda lfn:lfntest.data -d \`
  `lcgse01.gridpp.rl.ac.uk`
- Try the previous test (w/ edg-job-list-match) – should find a new site willing to accept your job

Finding Your Data

Enabling Grids for E-science in Europe

Reminder: the listReplicas (lr) method

- `lcg-lr --vo gild lfn:flavia.example  # via LFN`
- `sfn://gridkap02.fzk.de/grid/fzk.de/mounts/nfs/data/lcg1/SE00/lhcb/generated/2004-06-02/file7115df45-b4c7-11d8-bb5e-eba42b5000d0`
- `lcg-lr --vo lhcb lr \    # via GUID`
  `guid:76373236-b4c7-11d8-bb5e-eba42b5000d0`
- `sfn://gridkap02.fzk.de/grid/fzk.de/mounts/nfs/data/lcg1/SE00/lhcb/generated/2004-06-02/file7115df45-b4c7-11d8-bb5e-eba42b5000d0`

Advanced commands

Enabling Grids for E-science in Europe

- Low level tools for distributed data copying & info
  - globus-url-copy
  - edg-gridftp-ls and friends
- Interaction with RLS components
  - edg-lrc (local replica catalog)
  - edg-rmc (replica metadata catalog, search on metadata)
- Google is your friend

Available APIs

- LCG-2 C APIs available for all lcg-utils functions
  - Man pages are available (try % man lcg_lr)
  - Documentation available soon
- LCG-2 C++ and Java APIs available for all catalog operations

http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-lrc-devguide.pdf

http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-rmc-devguide.pdf

Examples can be found:

http://isscvs.cern.ch:8180/cgi-bin/cvsweb.cgi/edg-rls-client/test/?cvsroot=lcgware

http://isscvs.cern.ch:8180/cgi-bin/cvsweb.cgi/edg-metadata-catalog client/test/?cvsroot=lcgware

http://grid-deployment.web.cern.ch/grid-deployment/eis/tutorial/RLS_API.tar.gz

Available APIs

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <stdio.h>
#include <errno.h>

// lcg_util is a C library. Since we write C++ code here, we need to
// use extern C
//
extern "C"
{
#include <lcg_util.h>
}
using namespace std;
/*****************************************************************************/
/* The folling example code shows you how you can use the lcg_util API for */
/* replica management. We expect that you modify parts of this code in      */
/* to make it work in your environment. This is particularly indicated      */
/* by ACTION, i.e. your action is required.                                 */
/*****************************************************************************/
int main ()
{
cout << "Data Management API Example " << endl;
char *vo = "cms"; // ACTION: fill in your correct VO here: gilda !
cout << "-----------------------------------------------" << endl;
```

C APIs

Available APIs

Enabling Grids for E-science in Europe

```
// Copy a local file to the Storage Element and register it in RLS
//
char *localFile = "file:/tmp/test-file"; // ACTION: create a testfile
char *destSE = "lxb0707.cern.ch"; // ACTION: fill in a specific SE char

*actualGuid = (char*) malloc(50);
int verbose = 2; // we use verbosity level 2
int nbstreams = 8; // we use 8 parallel streams to transfer a file

lcg_cr(localFile, destSE, NULL,
        NULL, vo, NULL, nbstreams,
       NULL, 0, verbose, actualGuid);

if (errno)
{
  perror("Error in copyAndRegister:");
 return -1;
} else {
   cout << "We registered the file with GUID: " << actualGuid << endl;
}
cout << "-----------------------------------------------" << endl;
```

Copy and Register

Available APIs

```
// Call the listReplicas (lcg_lr) method and print the returned URLs
//
// The actualGuid does not contain the prefix "guid:". We add it here and
// then use the new guid as a parameter to list replicas
//
std::string guid = "guid:";
guid.insert(5,actualGuid);
char ***pfns = (char***) malloc(200);

lcg_lr((char*) guid.c_str(), vo, NULL, 0, pfns);

if(errno)
{
   perror("Error in listReplicas:");
  free(pfns);
  return -1;
} else {
  cout << "PFN = " << **pfns << endl;
}

free(pfns);

cout << "----------------------------------------------" << endl;
```

List Replicas

Available APIs

```
// Delete the replica again
//
int rc = lcg_del((char*) guid.c_str(), 1, destSE, vo, NULL, 0, verbose);

if(rc != 0)
{
   perror("Error in delete:");
  return -1;
} else {
  cout << "Delete OK" << endl;
}

return 0;

}
```

Delete Replica

Available APIs

```
CC = g++
GLOBUS_FLAVOR = gcc32

all: data-management

data-management: data-management.o
                $(CC) -o data-management \
                -L${GLOBUS_LOCATION}/lib -lglobus_gass_copy_${GLOBUS_FLAVOR} \
                -L${LCG_LOCATION}/lib -llcg_util -lgfal \
                data-management.o

data-management.o: data-management.cpp
                $(CC) -I ${LCG_LOCATION}/include -c data-management.cpp

clean:
                rm -rf data-management data-management.o
```

Makefile used

Summary

Enabling Grids for E-science in Europe

- We gave an introduction to the LCG-2 Data Management Middleware Components and Tools
- We described how to use the available CLIs
- Use-case scenarios of Data Movement on Grid
- We presented the available APIs
- An example usage of lcg_util library is shown