Enabling Grids for E-science in Europe

www.eu-egee.org

First Latinamerican Grid Workshop
18 November 2004

Job Services With Data Requirements

Elisabetta Ronchieri
INFN CNAF

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

Contents

Enabling Grids for E-science in Europe

- Job Description Language
- Workload Management System (WMS) Matchmaking
- The BrokerInfo
- Data Location Interface (DLI)

Job Description Language

Enabling Grids for E-science in Europe

- The supported attributes are grouped in two categories:
  - **Job Attributes**
    - Define the job itself
  - **Resources**
    - Taken into account by the RB for carrying out the matchmaking algorithm (to choose the “best” resource where to submit the job)
    - *Computing Resource (see talk Job Services)*
      - Used to build expressions of Requirements and/or Rank attributes by the user
      - Have to be prefixed with “other.”
    - *Data and Storage resources*
      - Input data to process, SE where to store output data, protocols spoken by application when accessing SEs

JDL: Data attributes

Enabling Grids for E-science in Europe

- **InputData** (first version)
  - Refers to data used as input by the job: these data are published in the Replica Location Service (RLS) and stored in the SEs)
  - Contains Logical File Names (LFNs) and/or Global Unique Identifiers (GUIDs)
- **DataAccessProtocol** (mandatory if InputData has been specified)
  - Specifies the protocol or the list of protocols which the application is able to speak with for accessing *InputData* on a given SE
- **OutputSE**
  - Represents the Uniform Resource Identifier of the output SE
  - RB uses it to choose a CE that is compatible with the job and is close to SE

JDL: Data attributes

Enabling Grids for E-science in Europe

- Example **InputData**, **DataAccessProtocol**, and **OutputSE** attributes in the JDL

```
InputData = {"lfn:green", "guid:ec3ee4d2-a653-11d7-
849e-ea0706438314"};
DataAccessProtocol = "gsiftp";
OutputSE = "se012.cnaf.infn.it";
```

Example of JDL file with InputData

Enabling Grids for E-science in Europe

```
[
JobType = "Normal";
Executable = "gridTest";
StdError = "sim.err";
StdOutput = "sim.out";
InputData = {"lfn:green", "guid:ec3ee4d2-a653-11d7-
849e-ea0706438314"};
DataAccessProtocol = "gsiftp";
InputSandbox = "/home/joda/test/gridTest";
OutputSandbox = "sim.err", "sim.out"};
Requirements  = (other.GlueHostOperatingSystemName
== "linux") && (other. GlueCEStateFreeCPUs > 4);
Rank = other.GlueCEPolicyMaxCPUTime;
]
```

JDL: Data attributes

Enabling Grids for E-science in Europe

- **OutputData**
  - Allows the user for the automatic upload and registration in Grid of files produced by the job on the WN
  - Specifies several output files
  - Each **OutputData** consists of three elements:
    - **OutputFile** (mandatory)
      - Specifies the name of the generated file to be uploaded to the Grid
    - **StorageElement** (optional)
      - Indicates the SE where the file should be stored
      - If unspecified, the default SE is chosen using ($VO_<VO>_DEFAULT_SE)
    - **LogicalFileName** (optional)
      - Represents a LFN the user wants to be associated to the output file in Grid
      - If unspecified, it is set a GUID

JDL: Data attributes

Enabling Grids for E-science in Europe

- Example **OutputData** attribute in the JDL

```
OutputData = {
   [
      OutputFile = "toto.out" ;
      StorageElement = "adc0021.cern.ch";
      LogicalFileName = "lfn:theBestTotoEver"
   ],
   [
      OutputFile = "toto2.out" ;
      StorageElement = "adc0021.cern.ch" ;
   ]
};
```

Automatic upload and Registration of output file

Enabling Grids for E-science in Europe

- The automatic upload of data from a job is done on the WN
- On the UI, the user has to specify in the JDL
  - **OutputData** attribute: it allows the user to automatically upload and register files generated by the job into the Grid

```
OutputFile = "dummy.dat";
StorageElement = "gilda-se-01.pd.infn.it";
LogicalFileName = "lfn:AntonioDelgado_outputData";
```

  - **OutputSE** attribute: it represents the SE where the job wants to save the files generated during its execution

```
OutputSE = "gilda-se-01.pd.infn.it";
```

- The automatically upload of data from a job with a default SE will work properly if that CE has at least one SE defined as close in the Information System
- Otherwise, the **StorageElement** Attribute is used

Automatic upload and Registration of output file

Enabling Grids for E-science in Europe

- On the WN, it is generated an output file called DSUpload_<unique_jobid_string>.out, where the JobWrapper save the result of the upload operations
  - It is automatically put in the files specified in the **OutputSandbox** attribute
- Once the user run the edg-job-get-output <jobid> command, he/she can check the result of the uploading simply doing
  - `cat DSUpload_<jobid>.out`

```
# <outputfile> <lfn|guid|Error>
dummy.dat    lfn:AntonioDelgado_outputData
```

- The body of the DSUpload_<unique_jobid_string>.out, file, meaning that dummy.dat was upload successfully and registered as lfn:AntonioDelgado_outputData

Example of JDL file with OutputData

Enabling Grids for E-science in Europe

```
[
JobType = "Normal";
Executable = "gridTest";
StdError = "sim.err";
StdOutput = "sim.out";
OutputData = [
 OutputFile = "dummy.dat";
 StorageElement = "gilda-se-01.pd.infn.it";
 LogicalFileName =  "lfn:AntonioDelgado_outputData";
]
OutputSE = "se012.cnaf.infn.it";
]
```

The Matchmaking algorithm

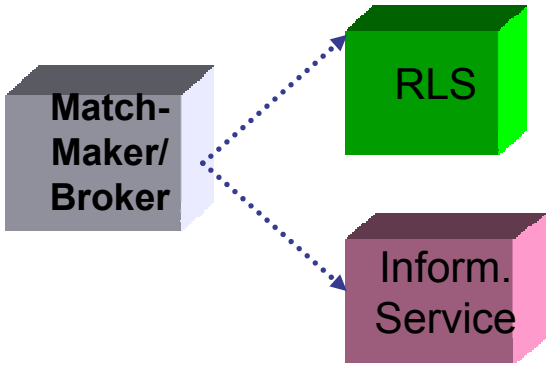
Enabling Grids for E-science in Europe

- The matchmaker has the goal to find the best suitable CE where to execute the job
- There are three different scenarios to be dealt with separately:
  - Direct job submission (*see talk Job Services*)
  - Job submission without data-access requirements (*see talk Job Services*)
  - Job submission with data-access requirements

The Matchmaking algorithm: job submission with data access requirements

Enabling Grids for E-science in Europe

- The Matchmaker interacts with Replica Location Service (RLS) and Information Service
- The RLS is used to resolve the location of data
- The Matchmaker has to find the most suitable CEs taking into account the SEs where both input data are physically stored and output data should be staged
- The user has to specify in the JDL the following attributes
  - **OutputSE** represents the SE where the output file should be staged
  - **InputData** represents the input files (LFNs, GUIDs)
  - **DataAccessProtocol** represents the protocol spoken by the application to access the files



The Matchmaking algorithm: job submission with data access requirements

Enabling Grids for E-science in Europe

- What does it change respect to the job submission without data access requirements?
- The matchmaking algorithm is always characterized by two steps:
  - Requirements checks
  - Ranking checks
- In addition, the Broker
  - Performs a pre-match processing
  - Classifies those CEs satisfying both data access and user authorization requirements

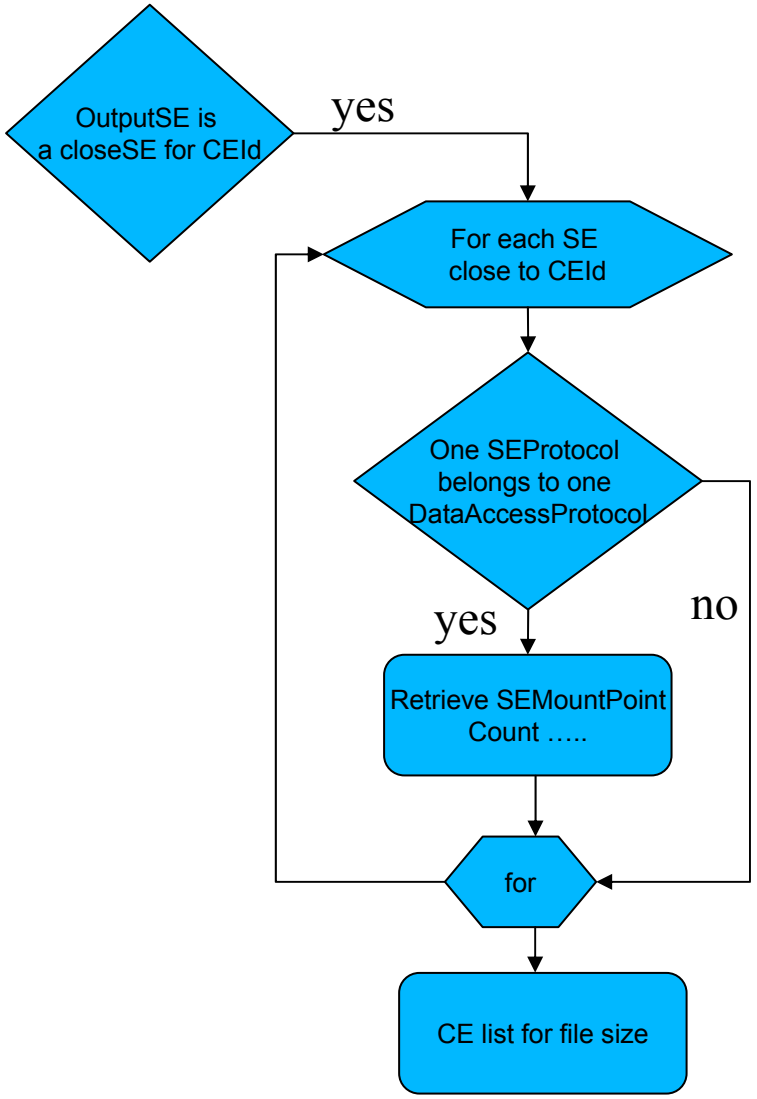
The Matchmaking algorithm: job submission with data access requirements

Enabling Grids for E-science in Europe

- During the pre-match processing, the Broker contacts the RLS in order to
  - resolve logical file names
  - collect all the information about SEs
- This information will be used to write down the BrokerInfo file (*see later*)
- The BrokerInfo file is added to the list files of the **InputSandbox** attribute and sent to the WN

The Matchmaking algorithm: job submission with data access requirements

- During the CEs classification, the Broker
  - finds the CEs satisfying both the authorization requirements and having the OutputSE close to them
    - **OutputSE** attribute contains the SE where data has to be stored
  - for each SE close to the CE, checks if at lest one SEProtocol belongs to protocols specified in the **DataAccessProtocol** attribute
    - The SEMountPoint is retrieved
    - The number of distinguished Input Data files supplied for such SE is counted
  - classifies the CEs depending on the number of input files stored in SEs



The BrokerInfo

- The BrokerInfo file allows the user job to access at run time certain information related to the job, e.g. the name of CE, the file specified in the InputData attribute, etc
- It is always created in the job working directory on the WN
- It is called .BrokerInfo
- Its syntax is based on Condor Classads
- It can be parsed using a CLI:
  - directly from the job on the WN
  - from the user on the UI
    - IMPORTANT: the .BrokerInfo file must be retrieved with the rest of the generated output files when the job finishes

      OutputSandbox = {".BrokerInfo", ….};

The BrokerInfo command

- `edg-brokerinfo [-v] [-f <filename>] function [parameter] [parameter] …`
  - [-v] produces more verbose output
  - [-f <filename>] specifies the BrokerInfo file that has to be parsed
    - If unspecified, the BrokerInfo file is taken by the environment variable $EDG_WL_RB_BROKERINFO

The BrokerInfo command

Enabling Grids for E-science in Europe

- function is one of the following:
  - getCE returns the name of the CE the job is running on
  - getVirtulOrganization returns the name of the VO specified in the VirtualOrganization JDL attribute
  - getDataAccessProtocol returns the protocol list specified in the DataAccessProtocol JDL attribute
  - getInputData returns the files specified in the InputData JDL attribute
  - getSEs returns the list of the SE with contain a copy of at least one file among those specified in the InputData JDL attribute
  - getCloseSEs returns a list of SEs close to the CE
  - getSEMountPoint <SE> returns the access point for the specified <SE>, if it is in the list of close SEs of the WN
  - getSEPort <SE> <Protocol> returns the port number used by <SE> for the data transfer protocol <Protocol>
  - getLFN2SFN <LFN> returns the storage file name of the file specified by <LFN>
    - <LFN> is a logical file name of a GUID specified in the InputData attribute

The BrokerInfo file

```
[
   ComputingElement =
   [
           CloseStorageElements =
            {
                    [
                            GlueSAStateAvailableSpace = 399645448;
                            GlueCESEBindCEAccesspoint = "/flatfiles/SE00";
                            mount = GlueCESEBindCEAccessPoint;
                            name = "grid009.to.infn.it";
                            freespace = GlueSAStateAvailableSpace
                     ]
            };
            name = "grid008.to.infn.it:2119/jobmanager-lcgpbs-long"
    ];
    InputFNs =
    {
    };
    StorageElements =
    {
    };
    VirtualOrganisation = "dteam"
]
```

The BrokerInfo file

```
 DataAccessProtocol =  {
             "gridftp" ,
             "file"
};
InputFNs =  {
  [
    name = "lfn:wp1-test-file-01-lfn";
    SFNs =  {
        "sfn://testbed013.cern.ch/wp1/wp1-test-file-01",
        "sfn://testbed113.cern.ch/wp1/wp1-test-file-01"
    }
  ],
  [
    name = "lfn:wp1-test-file-02-lfn";
    SFNs = {
        "sfn://testbed001.cern.ch/wp1/wp1-test-file-02",
        "sfn://testbed005.cern.ch/wp1/wp1-test-file-02"
    }
  ],
  [
    name = "lfn:wp1-test-file-03-lfn";
    SFNs =  {
        "sfn://testbed002.cern.ch/wp1/wp1-test-file-03"
    }
  ]
}
```

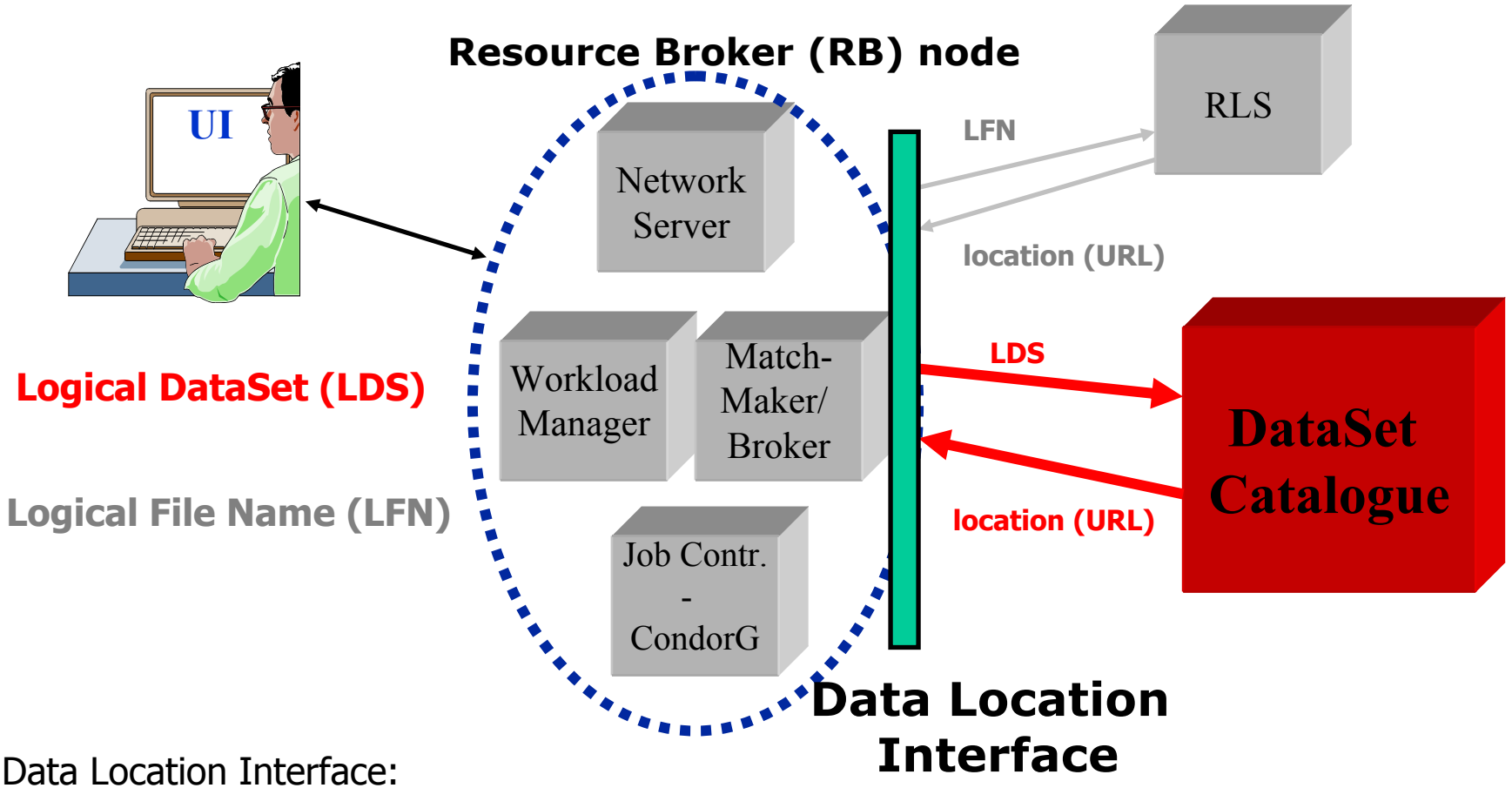
DataLocationInterface (DLI) and Data Sets

Enabling Grids for E-science in Europe

- WMS works with single files using LFNs and/or GUIDs
- Main idea is that a user of the WMS "prefers" to work with data sets rather than single files
  - Enhance file based interface with dataset based interface
- A data set is a collection of files that are treated as a set
- Logical Data Set (LDS): to identify a given set
- Physical DataSet is identified by "one" location (URL)
  - Preferably, all files of a physical dataset are located in one place.
  - However, datasets can also be distributed over several locations or not all files might be available (partially available).

From Heinz Stockinger's "Resource Broker and Data Catalogues (DataLocationInterface)" talk at INFN GRID Workshop

DLI: WMS – DataSet Catalogue



Data Location Interface:

URL-List = listReplicas("lfn","lfn:my-lfn")

URL-List = listReplicas("lds","lds:mu03_tt_4mu/mu_Hit245_2_g133)

From Heinz Stockinger's "Resource Broker and Data Catalogues (DataLocationInterface)" talk at INFN GRID Workshop

DLI: Change in JDL

Enabling Grids for E-science in Europe

- Introduce a new data type: Logical DataSet (LDS)
  - Prefix "lds:" to distinguish between LFNs and Logical DataSets
  - Syntax: lds:anyString
  - Examples:
    - lds:dataset/owner
    - lds:mu03_tt_4mu/mu_Hit245_2_g133
    - lds:myDataSetName
- A different catalog can be used based on experiment (VO) needs

DLI: JDL – Data Attributes

- **InputData** (second version)
  - Refers to data used as input by the job: these data are published in a dataset catalogue that is registered in MDS
  - Contains Logical Data Sets (LDSs) (*see later*)
- **ReplicaCatalog** (always optional: *it could be specified if InputData is used in the second version*)
  - Specifies the dataset catalogue explicitly
- **Example 1**

```
InputData = "lds:mu03_tt_4mu/mu_Hit245_2_g133";
ReplicaCatalog = "http://hostname.cern.ch:8085/";
```

- **Example 2**

```
InputData = "lds:test-file-dli";
```

DLI: Data Location Interface

Enabling Grids for E-science in Europe

- The Data Location Interface has been discussed with several partners in CMS, LCG and EGEE as well as partly with other HEP experiments
  - Document available at:
    - http://cmsdoc.cern.ch/cms/grid/docs/DataLocationInterface.pdf
  - Uses a SOAP interface. WSDL file at:
    - http://cmsdoc.cern.ch/cms/grid/docs/DataLocationInterface.wsdl
- Proposed as a common interface to data location services – for Matchmaker only
  - i.e. We can interface to "any" new catalogue that implements this interface
- DLI is currently not available on the GILDA testbed but in a feature release

From Heinz Stockinger's "Resource Broker and Data Catalogues (DataLocationInterface)" talk at INFN GRID Workshop

WMS API

Enabling Grids for E-science in Europe

- BrokerInfo CLI and APIs are described

  http://server11.infn.it/workload-grid/docs/edg-brokerinfo-user-guide-v2_2.pdf

Hands-on time!

Enabling Grids for E-science in Europe

1. Create an executable that, using the ***BrokerInfo APIs***, lists all close SEs.
   - Submit this job via a JDL file
   - Retrieve the output of the job
   - Check that the output contains the information you want
2. **Try to submit** a MPI job with data-access requirements