Enabling Grids for E-science in Europe

www.eu-egee.org

Workload Management System

Simone Campana
CERN/INFN-CNAF

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

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

EGEE/LCG Workload Management System

Enabling Grids for E-science in Europe

- The Workload Management System (WMS)
  - is a layer of the grid middleware
  - It allows the user to interact with the Grid resources
- What does it allow Grid users to do?
  - To submit their jobs
  - To execute them on the “best resources”
  - To get information about their status
  - To retrieve their output

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

Job Preparation

Enabling Grids for E-science in Europe

- Information to be specified when a job has to be submitted:
  - Job characteristics
  - Job requirements and preferences on the computing resources
    - Also including software dependencies
  - Job data requirements
- Information specified using a Job Description Language (JDL)
  - Based upon Condor's *CLASSified ADvertisement language (ClassAd)*
    - Fully extensible language
    - A ClassAd
      - Constructed with the classad construction operator []
      - It is a sequence of attributes separated by semi-colon (;).
- The JDL allows definition of a set of attribute,
  - the WMS takes them into account when making its scheduling decision

Job Preparation

- An attribute is a pair (key, value), where value can be a Boolean, an Integer, a list of strings, ...
  - <attribute> = <value>;
- In case of literal string for values:
  - if a string itself contains double quotes, they must be escaped with a backslash
    - `Arguments = " \"Hello\" 10";`
  - the character "'" cannot be specified in the JDL
  - special characters such as &, |, >, < are only allowed
    - if specified inside a quoted string
    - if preceded by triple \
      - `Arguments = "-f file1\\\&file2";`
- Comments must be preceded by a sharp character (#) or have to follow the C++ syntax
- The JDL is sensitive to blank characters and tabs
  - they should not follow the semicolon (;) at the end of a line

Job Description Language

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

JDL: some relevant attributes

Enabling Grids for E-science in Europe

- JobType
  - *Normal* (simple, sequential job), *Interactive*, *MPICH*, *Checkpointable*
  - Or combination of them
- Executable (mandatory)
  - The command name
- Arguments (optional)
  - Job command line arguments
- StdInput, StdOutput, StdError (optional)
  - Standard input/output/error of the job
- Environment (optional)
  - List of environment settings
- InputSandbox (optional)
  - List of files on the UI local disk needed by the job for running
  - The listed files will automatically staged to the remote resource
- OutputSandbox (optional)
  - List of files, generated by the job, which have to be retrieved
- VirtualOrganisation (optional)
  - A different way to specify the VO of the user

Requirements

- Job requirements on the resources
- Specified using GLUE attributes of resources published in the Information Service
- Its value is a boolean expression
- Only one requirements can be specified
  - if there are more than one, only the last one is taken into account
- If not specified, default value defined in UI configuration file is considered
  - Default: *other.GlueCEStateStatus == "Production"* (the resource has to be able to accept jobs and dispatch them on WNs)

Relevant Glue Attributes

Enabling Grids for E-science in Europe

- State (objectclass GlueCEState)
  - GlueCEStateRunningJobs:
    - number of running jobs
  - GlueCEStateWaitingJobs:
    - number of jobs not running
  - GlueCEStateTotalJobs:
    - total number of jobs (running + waiting)
  - GlueCEStateStatus:
    - queue status: queueing (jobs are accepted but not run), production (jobs are accepted and run), closed (jobs are neither accepted nor run), draining (jobs are not accepted but those in the queue are run)
  - GlueCEStateWorstResponseTime:
    - worst possible time between the submission of a job and the start of its execution
  - GlueCEStateEstimatedResponseTime:
    - estimated time between the submission of a job and the start of its execution
  - GlueCEStateFreeCPUs:
    - number of CPUs available to the scheduler

Relevant Glue Attributes

Enabling Grids for E-science in Europe

- Policy (objectclass GlueCEPolicy)
  - GlueCEPolicyMaxWallClockTime:
    - maximum wall clock time available to jobs submitted to the CE, in seconds (previously it was in minutes)
  - GlueCEPolicyMaxCPUTime:
    - maximum CPU time available to jobs submitted to the CE, in seconds (previously it was in minutes)
  - GlueCEPolicyMaxTotalJobs:
    - maximum allowed total number of jobs in the queue
  - GlueCEPolicyMaxRunningJobs:
    - maximum allowed number of running jobs in the queue
  - GlueCEPolicyPriority:
    - information about the service priority

Relevant Glue Attributes

Enabling Grids for E-science in Europe

- Architecture (objectclass GlueHostArchitecture)
  - GlueHostArchitecturePlatformType:
    - platform description
  - GlueHostArchitectureSMPSize:
    - number of CPUs
- Processor (objectclass GlueHostProcessor)
  - GlueHostProcessorVendor:
    - name of the CPU vendor
  - GlueHostProcessorModel:
    - name of the CPU model
  - GlueHostProcessorVersion:
    - version of the CPU
  - GlueHostProcessorOtherProcessorDescription:
    - other description for the CPU
  - […]

Relevant Glue Attributes

Enabling Grids for E-science in Europe

- Application software (objectclass GlueHostApplicationSoftware)
  - GlueHostApplicationSoftwareRunTimeEnvironment:
    - list of software installed on this host
- Main memory (objectclass GlueHostMainMemory)
  - GlueHostMainMemoryRAMSize:
    - physical RAM
  - GlueHostMainMemoryVirtualSize:
    - size of the configured virtual memory
- Benchmark (objectclass GlueHostBenchmark)
  - GlueHostBenchmarkSI00:
    - SpecInt2000 benchmark
  - GlueHostBenchmarkSF00:
    - SpecFloat2000 benchmark
- Network adapter (objectclass GlueHostNetworkAdapter)
  - […]
  - GlueHostNetworkAdapterOutboundIP:
    - permission for outbound connectivity
  - GlueHostNetworkAdapterInboundIP:
    - permission for inbound connectivity

JDL Requirements (again …)

Enabling Grids for E-science in Europe

- Possible requirements values are below reported (from DC experience):
  - *other.GlueCEInfoLRMSType == "PBS" && other.GlueCEInfoTotalCPUs > 1* (the resource has to use PBS as the LRMS and whose WNs have at least two CPUs)
  - *Member("CMSIM-133", other.GlueHostApplicationSoftwareRunTimeEnvironment)* (a particular experiment software has to run on the resource and this information is published on the resource environment)
    - The *Member* operator tests if its first argument is a member of its second argument. Used in case of multi attribute.
  - *RegExp("cern.ch", other.GlueCEUniqueId)* (the job has to run on the CEs in the domain cern.ch)
    - Matches the regular expression
  - *(other.GlueHostNetworkAdapterOutboundIP == true) && Member("VO-alice-Alien", other.GlueHostApplicationSoftwareRunTimeEnvironment) && Member("VO-alice-Alien-v4-01-Rev-01", other.GlueHostApplicationSoftwareRunTimeEnvironment) && (other.GlueCEPolicyMaxWallClockTime > 86000)* (the resource must have some packages installed VO-alice-Alien and VO-alice-Alien-v4-01-Rev-01 and the job has to run for more than 86000 WallClock time units)

Rank

- Expresses preference (how to rank resources that have already met the Requirements expression)
- It is expressed as a floating-point number
- The CE with the highest rank is the one selected (see Matchmaking later on)
- If not specified, default value defined in the UI configuration file is considered
- Possible rank values are below reported:
  - *-other.GlueCEStateEstimatedResponseTime* (the lowest estimated traversal time)
    - Usually the default
  - *other.GlueCEStateFreeCPUs* (the highest number of free CPUs)
    - Bad idea: number of free CPU published per QUEUE, not per VO
  - *(other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -other.GlueCEStateWaitingJobs)* (the number of waiting jobs is used if this number is not null and the rank decreases as the number of waiting jobs gets higher; if there are not waiting jobs, the number of free CPUs is used)

Essential JDL

Enabling Grids for E-science in Europe

- At least one has to specify the following attributes:
  - the name of the executable
  - the files where to write the standard output and standard error of the job
  - the arguments to the executable, if needed
  - the files that must be transferred from UI to WN and viceversa

```
[
Executable = "ls -al";
StdError = "stderr.log";
StdOutput = "stdout.log";
OutputSandbox = {"stderr.log", "stdout.log"};
]
```

Example of JDL file

Enabling Grids for E-science in Europe

```
[
JobType = “Normal”;
Executable = "$(CMS)/exe/sum.exe";
InputSandbox = {"/home/user/WP1testC","/home/file*”,
"/home/user/DATA/*"};
OutputSandbox = {“sim.err”, “test.out”, “sim.log"};
Requirements  = (other. GlueHostOperatingSystemName
== “linux") && (other.GlueCEPolicyMaxWallClockTime >
10000);
Rank = other.GlueCEStateFreeCPUs;
]
```

A “real world” JDL file

Enabling Grids for E-science in Europe

```
[
JobType = "normal";
Executable = "lexor_wrap.sh";
StdOutput = "dc2.003020.digit.A8_QCD._01730.job.log.3";
StdError = {"dc2.003020.digit.A8_QCD._01730.job.log.3";}
OutputSandbox {"metadata.xml", "lexor_wrap.log","dq_337704_stagein.log","dq_337704_stageout.log",\
"dc2.003020.digit.A8_QCD._01730.job.log.3" };
RetryCount = 0;
Arguments = "dc2.003020.simul.A8_QCD._01730.pool.root,\
                dc2.003020.digit.A8_QCD._01730.pool.root.3 100 0";
Environment = {
    "LEXOR_WRAPPER_LOG=lexor_wrap.log","LEXOR_STAGEOUT_MAXATTEMPT=5","LEXOR_STAGE
    OUT_INTERVAL=60","LEXOR_LCG_GFAL_INFOSYS=atlas-
    bdii.cern.ch:2170","LEXOR_T_RELEASE=8.0.7","LEXOR_T_PACKAGE=8.0.7.5/JobTransforms","LEXO
    R_T_BASEDIR=JobTransforms-08-00-07-
    05","LEXOR_TRANSFORMATION=share/dc2.g4digit.trf","LEXOR_STAGEIN_LOG=dq_337704_stagein.l
    og","LEXOR_STAGEIN_SCRIPT=dq_337704_stagein.sh","LEXOR_STAGEOUT_LOG=dq_337704_stag
    eout.log","LEXOR_STAGEOUT_SCRIPT=dq_337704_stageout.sh" };
MyProxyServer = "lxb0727.cern.ch";
VirtualOrganisation = "atlas";
 rank = -other.GlueCEStateEstimatedResponseTime
```

A “real world” JDL file (…continues)

Enabling Grids for E-science in Europe

```
requirements = (
Member("VO-atlas-lcg-release-0.0.2",
    other.GlueHostApplicationSoftwareRunTimeEnvironment) &&
(other.GlueCEStateStatus == "Production“) &&
!Member("VO-atlas-has-m1",
    other.GlueHostApplicationSoftwareRunTimeEnvironment)) &&
    (other.GlueCEInfoHostName != "lcgce02.gridpp.rl.ac.uk" ) &&
    (other.GlueCEInfoHostName != "lcg-ce.lps.umontreal.ca" ) &&
    (other.GlueCEInfoHostName != "lcgce02.triumf.ca" ) &&
    (other.GlueCEInfoHostName != "ce-a.ccc.ucl.ac.uk" ) &&
Member("VO-atlas-release-8.0.7",
    other.GlueHostApplicationSoftwareRunTimeEnvironment)) &&
( other.GlueCEPolicyMaxCPUTime >= (Member("LCG-
    2_1_0",other.GlueHostApplicationSoftwareRunTimeEnvironment) ?
( 36000000 / 60 ) : 36000000 ) / other.GlueHostBenchmarkSI00 ) ) &&
( other.GlueHostNetworkAdapterOutboundIP == true )  &&
(other.GlueHostMainMemoryRAMSize >= 512 ) );
]
```

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

Job Submission

Enabling Grids for E-science in Europe

edg-job-submit [-r *<res_id>*] [-c *<config file>*] [-vo *<VO>*] [-o *<output file>*] *<job.jdl>*

-r the job is submitted directly to the computing element identified by *<res_id>*

-c the configuration file *<config file>* is pointed by the UI instead of the standard configuration file

-vo the Virtual Organisation (if user is not happy with the one specified in the UI configuration file)

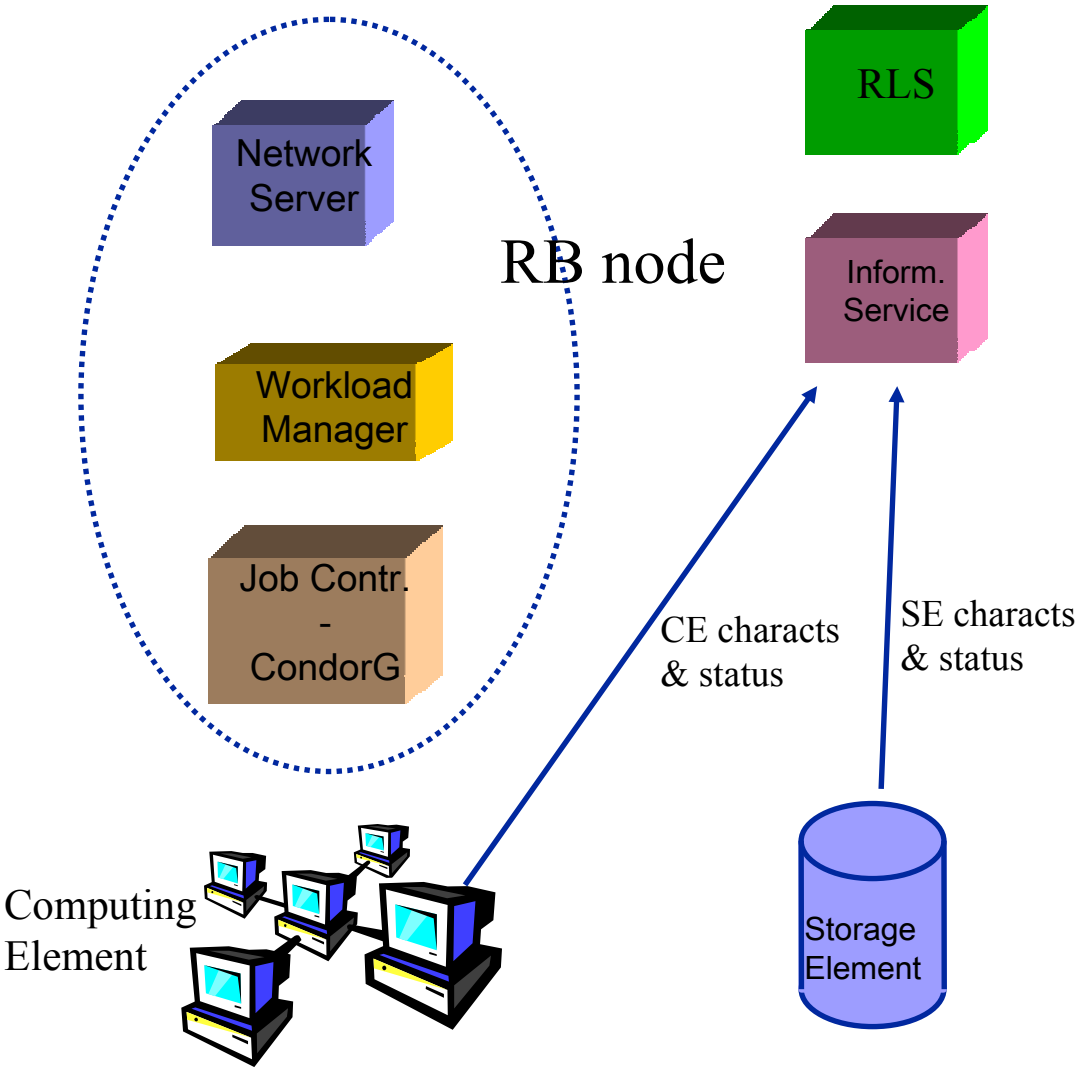
-o the generated edg_jobId is written in the *<output file>*

Useful for other commands, e.g.:

edg-job-status -i *<input file>* (or edg_jobId)

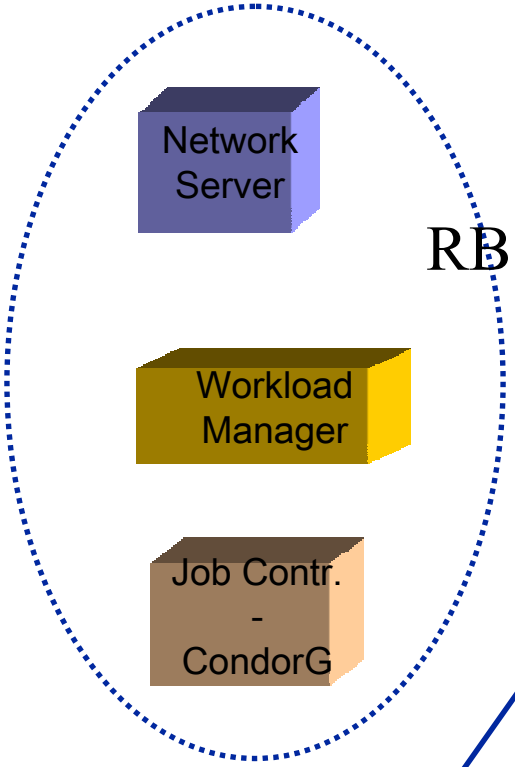
-i the status information about edg_jobId contained in the *<input file>* are displayed

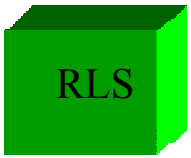
Job Submission



Job Submission

UI: allows users to access the functionalities of the WMS
(via command line, GUI, C++ and Java APIs)



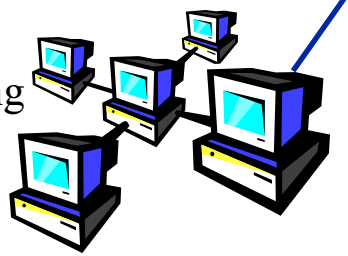


Job Status

submitted

CE characts & status

SE characts & status



Job Submission

Network Server

RB node

Workload

RLS

Inform. Service

Job Status

submitted

edg-job-submit myjob.jdl

Myjob.jdl

```
JobType = "Normal";
Executable     = "$(CMS)/exe/sum.exe";
InputSandbox = {"/home/user/WP1testC","/home/file*", "/home/user/DATA/*"};
OutputSandbox = {"sim.err", "test.out", "sim.log"};
Requirements   = other. GlueHostOperatingSystemName == "linux" &&
other.GlueCEPolicyMaxWallClockTime > 10000;
Rank  = other.GlueCEStateFreeCPUs;
```

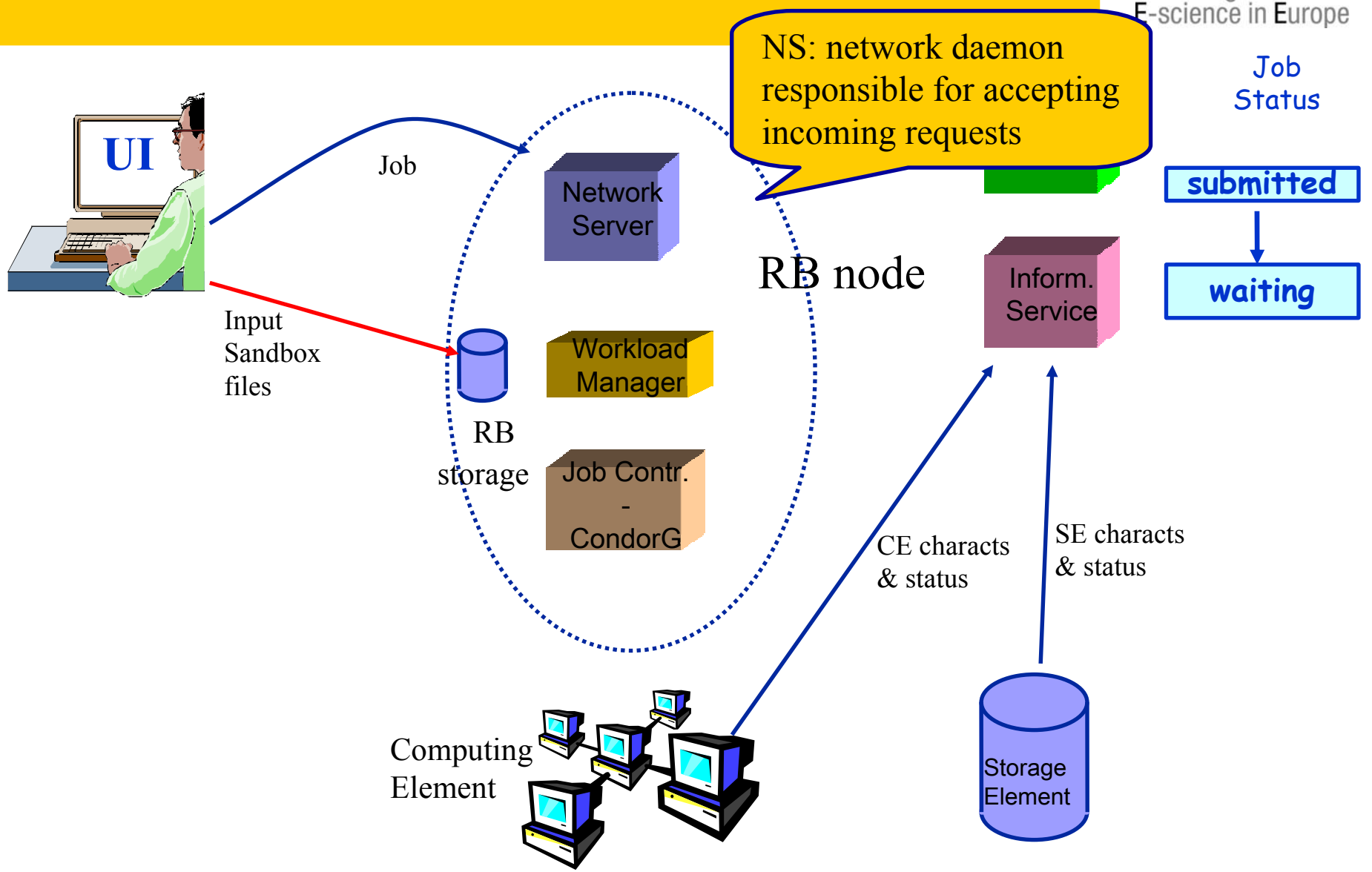
Job Description Language (JDL) to specify job characteristics and requirements

& status

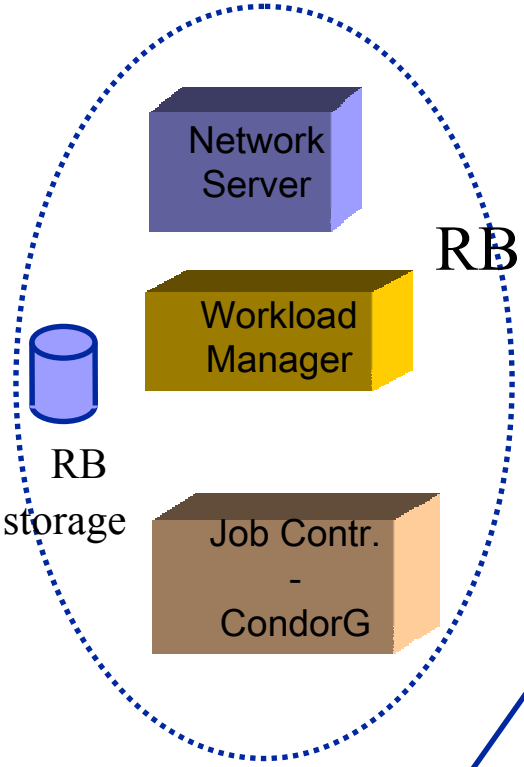
Computing Element

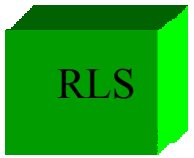
Storage Element

Job Submission

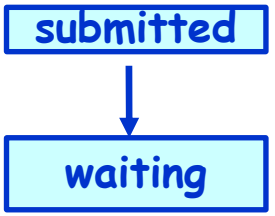


Job Submission





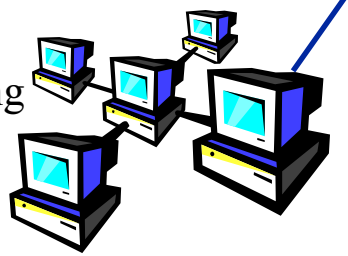
Job Status



WM: responsible to take the appropriate actions to satisfy the request

CE characts & status

SE characts & status



Job Submission

RB node

Network Server

Match-Maker/ Broker

Workload Manager

RB storage

Job Contr. - CondorG

Where must this job be executed ?

RLS

Inform. Service

Job Status

submitted

waiting

CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission

RB node

Network Server

Match-Maker/ Broker

Workload Man

RB storage

Job Con - CondorG

RLS

Inform. Service

Matchmaker: responsible to find the "best" CE where to submit a job

CE characts & status

SE characts & status

Computing Element

Storage Element

Job Status

submitted

waiting

Job Submission

RB node

Network Server

Match-Maker/ Broker

Workload Manager

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

What is the status of the Grid ?

CE characts & status

SE characts & status

Where are (which SEs) the needed data ?

Storage Element

Computing Element

Job Submission

RB node

Network Server

Match-Maker/ Broker

Workload Manager

CE choice

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

CE characts & status

SE characts & status

Computing Element

Storage Element

Job Submission

RB node

Network Server

Workload Manager

Job Adapter

RB storage

Job Contr. -

RLS

Inform. Service

Job Status

submitted

waiting

CE characts

SE characts & status

JA: responsible for the final "touches" to the job before performing submission (e.g. creation of wrapper script, etc.)

Computing Element

Storage Element

Job Submission

RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

JC: responsible for the actual job management operations (done via CondorG)

Computing Element

RLS

Inform. Service

CE characts & status

SE characts & status

Storage Element

Job Status

submitted

waiting

ready

Job Submission

RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

Input Sandbox files

Job

Computing Element

RLS

Inform. Service

CE characts & status

SE characts & status

Storage Element

Job Status

submitted

waiting

ready

scheduled

Job Submission

RB node

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

RLS

Inform. Service

Job Status

submitted

waiting

ready

scheduled

running

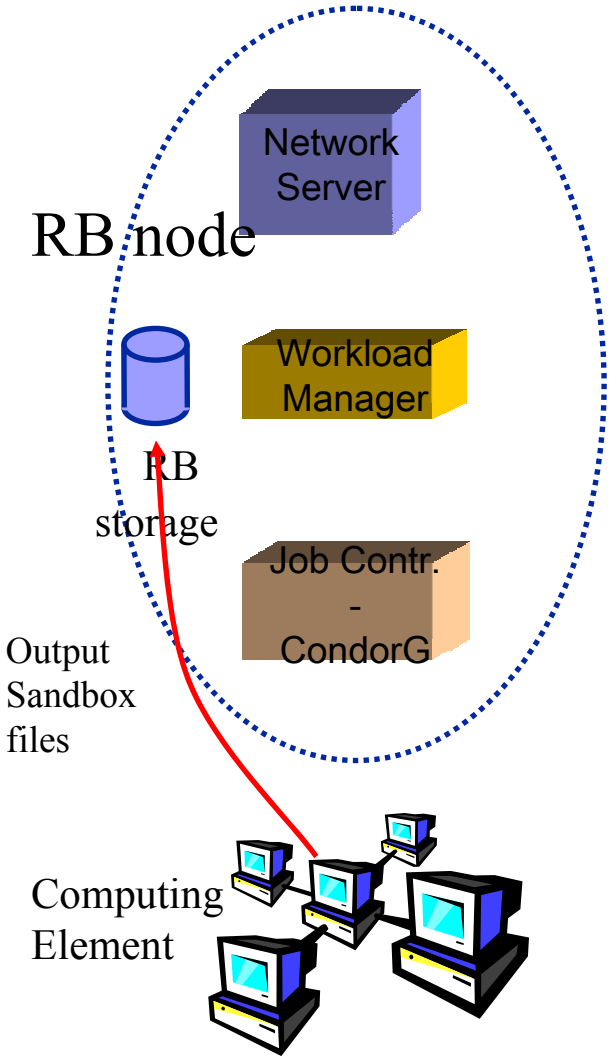
Computing Element

Job

"Grid enabled" data transfers/ accesses

Storage Element

Job Submission



RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

ready

scheduled

running

done

Job Submission

RB node

Network Server

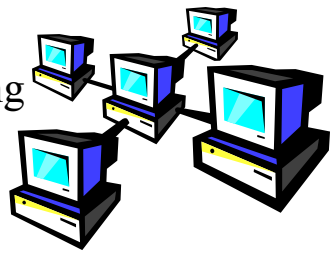
Workload Manager

RB storage

Job Contr. - CondorG

edg-job-get-output <dg-job-id>

Computing Element



RLS

Inform. Service

Storage Element

Job Status

submitted

waiting

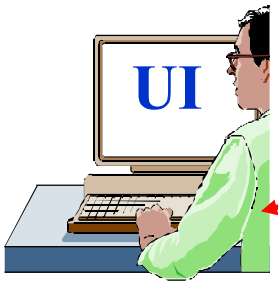
ready

scheduled

running

done

Job Submission



Output Sandbox files

Network Server

Workload Manager

RB storage

Job Contr. - CondorG

RB node

RLS

Inform. Service

Computing Element

Storage Element

Job Status

submitted

waiting

ready

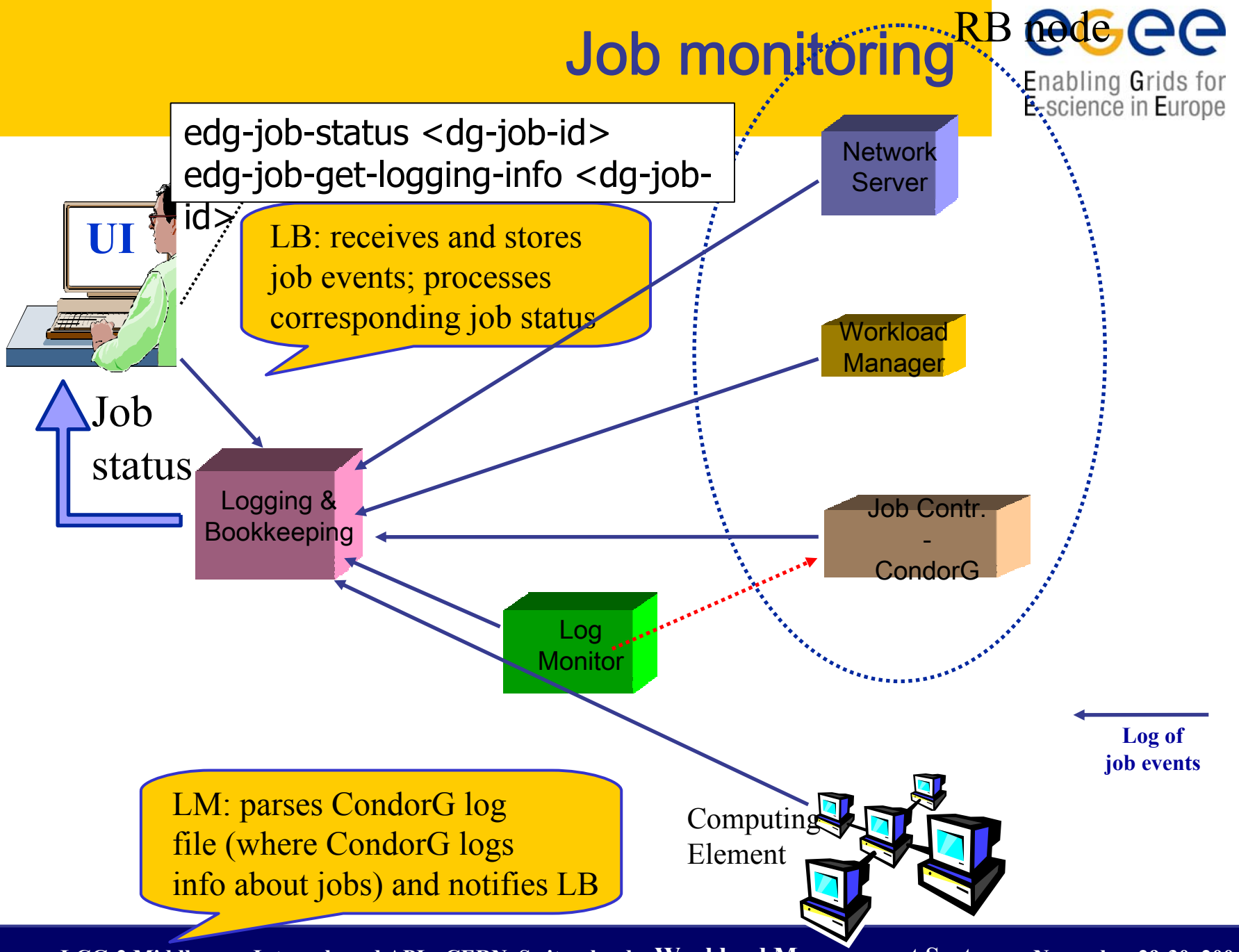
scheduled

running

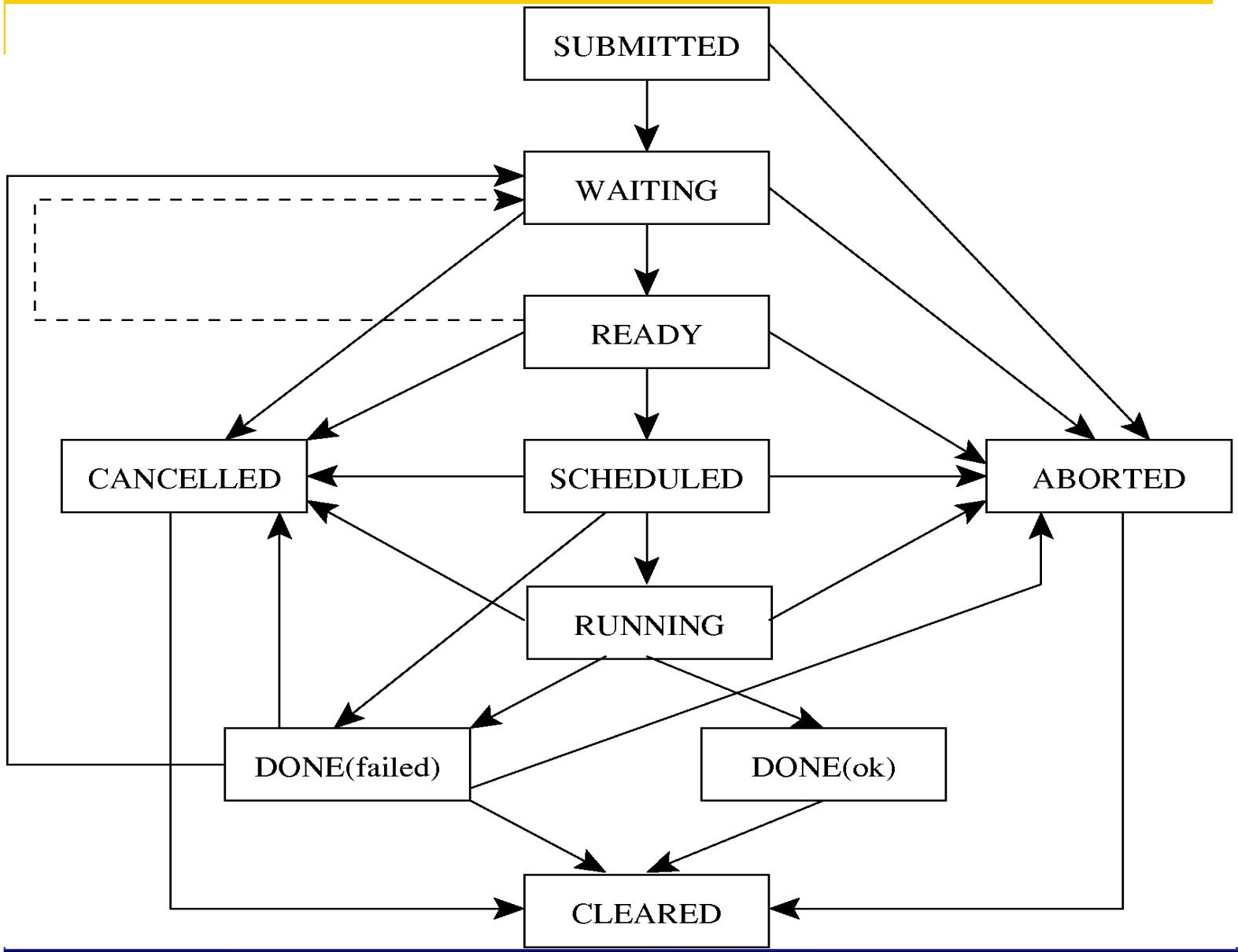
done

cleared

Job monitoring



Possible job states



Job resubmission

- If something goes wrong, the WMS tries to reschedule and resubmit the job (possibly on a different resource satisfying all the requirements)
- Maximum number of resubmissions: min(RetryCount, MaxRetryCount)
  - RetryCount: JDL attribute
  - MaxRetryCount: attribute in the “RB” configuration file
- e.g., to disable job resubmission for a particular job: *RetryCount=0;* in the JDL file
  - Choosen by several LCG experiments during data challenges
  - Experiments built a fault tollarant production system on top of WMS

Other (most relevant) UI commands

Enabling Grids for E-science in Europe

- **`edg-job-list-match`**
  - Lists resources matching a job description
  - The - - rank option prints the ranking of each resource
  - Performs the matchmaking without submitting the job
  - See matchmaking section later
- **`edg-job-cancel`**
  - Cancels a given job
- **`edg-job-status`**
  - Displays the status of the job
- **`edg-job-get-output`**
  - Returns the job-output (the OutputSandbox files) to the user

Other (most relevant) UI commands

Enabling Grids for E-science in Europe

- **`edg-job-get-logging-info`**
  - Displays logging information about submitted jobs (all the events “pushed” by the various components of the WMS)
  - Different levels of verbosity (-v option) :
    - Verbosity 1 is the most suitable for debugging
    - Verbosity 2 is just too much info
- About debugging a failed job
  - Understanding a job failure is not an easy task
  - Output of edg-job-get-logging-info not always straightforward to interpret
    - Short failure description
    - Difficult to distinguish a “grid” failure from a “user job” problem
    - Same error could be due to different causes (“in-famous” globus 155 …)
  - More useful info can be found in the logs of the RB
    - Not easily accessible by the end user
    - In principle can fetch them using gridftp but … come on …
  - User should try to log as much info as possible in the standard error file.
  - User should try to monitor the job/application
    - RGMA,GridIce for jobs status
    - RGMA for applications

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

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
- To accomplish this task, the WMS interacts with the other EGEE/LCG components (File Catalog, and Information Service)
- There are three different scenarios to be dealt with separately:
  - Direct job submission
  - Job submission without data-access requirements
  - Job submission with data-access requirements

The Matchmaking algorithm: direct job submission

Enabling Grids for E-science in Europe

- The user JDL contains a link to the resource to submit the job
- The WMS does not perform any matchmaking algorithm at all
- The job is simply submitted to the specified CE

IMPORTANT:

- If the CEId is specified then the WMS
  - Does NOT check whether the user is authorised to access the CE
  - Does NOT interacts with the File Catalog for the resolution of files requirements, if any
  - Only checks the JDL syntax, while converting the JDL into a ClassAd
- The user run the edg-job-submit --resource <ce_id> <nome.jdl> command

  ce_id = hostaname:port/jobmanager-lsf-grid01

The Matchmaking algorithm: job submission without data access requirements

Enabling Grids for E-science in Europe

- The user JDL contains some requirements
- Once the JDL has been received by the WMS and converted in ClassAd, the WMS invokes the matchmaker
- The matchmaker has to find if the characteristics and status of Grid resources match the job requirements
- There are two phases:
  - Requirements check:
    - The Matchmaker contacts the GOUT/II in order to create a set of suitable CEs compliant with user requirements and where the user is authorized to submit jobs
    - The Matchmaker creates the set of suitable CEs
  - Ranking phase:
    - The Matchmaker contacts directly the LDAP (GRIS) server of the involved CEs to obtain the values of those attributes that are in the rank JDL expression

The Matchmaking algorithm: job submission without data access requirements

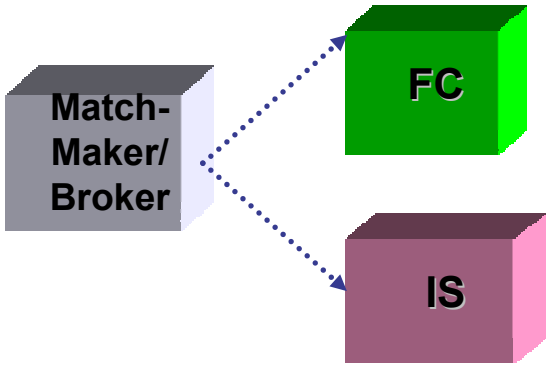
Enabling Grids for E-science in Europe

- There are two phases:
  - Requirements check:
    - The Matchmaker contacts the BDII in order to create a set of suitable CEs compliant with user requirements and where the user is authorized to submit jobs
    - The Matchmaker creates the set of suitable CEs
  - Ranking phase:
    - The Matchmaker contacts the BDII again to obtain the values of those attributes that are in the rank expression (used to contact GRISes)
- The CE with maximum rank value is selected
  - If 2 or more CE have same rank, Matchmakes selects random one
- Can adopt a stochastic selection (enabling fuzzyness)
  - The user has to set the JDL FuzzyRank attribute to true
  - The rank value = probability to select the CE
  - The higher the rank value is, the higher the probability is.

The Matchmaking algorithm: job submission with data access requirements

Enabling Grids for E-science in Europe

- The user can specify in the JDL the following attributes
  - InputData represents the input files
    - InputData = {“lfn:my-file-001"}
    - lfn=logical file name, see Data Management
  - OutputSE represents the SE where the output file should be staged
    - OutputSE = "gilda-se-01.pd.infn.it";
  - OutputData represents the output files
    - OutputFile = "dummy.dat";
    - StorageElement = "gilda-se-01.pd.infn.it";
    - LogicalFileName = "lfn:iome_outputData";
  - DataAccessProtocol represents the protocol spoken by the application to access the file
    - DataAccessProtocol = "gsiftp";



The Matchmaking algorithm: job submission with data access requirements

Enabling Grids for E-science in Europe

- The Matchmaker finds the most suitable CEs taking into account
  - the SEs where input data are physically stored
  - the SE where output data should be staged
- Previous to requirements and ranking checks, the broker
  - Performs a pre-match processing
    - interacts with File Catalog
  - Filters 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

- The Matchmaker interacts with a File Catalog and the Information Service
  - The FC is used to resolve the location of data
  - see Data Management talk for more details
- The Matchmaker finds most sutable CEs considering
  - SEs where both input data are physically stored
  - SEs where output data should be staged
- Previous to requirements and ranking checks, the broker
  - Performs a pre-match processing (access the FC)
  - Filters CEs satisfying both data access and user authorization requirements

The Matchmaking algorithm: job submission without data access requirements

Enabling Grids for E-science in Europe

- In general, the CE with maximum rank value is selected
- The matchmaker can select a CE randomly, if
  - there are two or more CEs that meet all the requirements
  - those CEs have the same rank
- The matchmaker can adopt a stochastic selection while searching for the best matching CE
  - enable fuzzyness in the matchmaking algorithm
  - The user has to set the JDL FuzzyRank attribute to true
  - The rank value represents the probability that each CE has to be selected as the best matching one
  - The higher the probability is, the higher the rank value is
  - The FuzzyRank algorithm has been recently improved
    - Not optimal behaviour observed during LHC Data Callenges

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 File Catalog in order to
  - resolve logical file names (aliases to real file names … wait for Data Management lecture)
  - 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

More on data access requirements: Data Sets

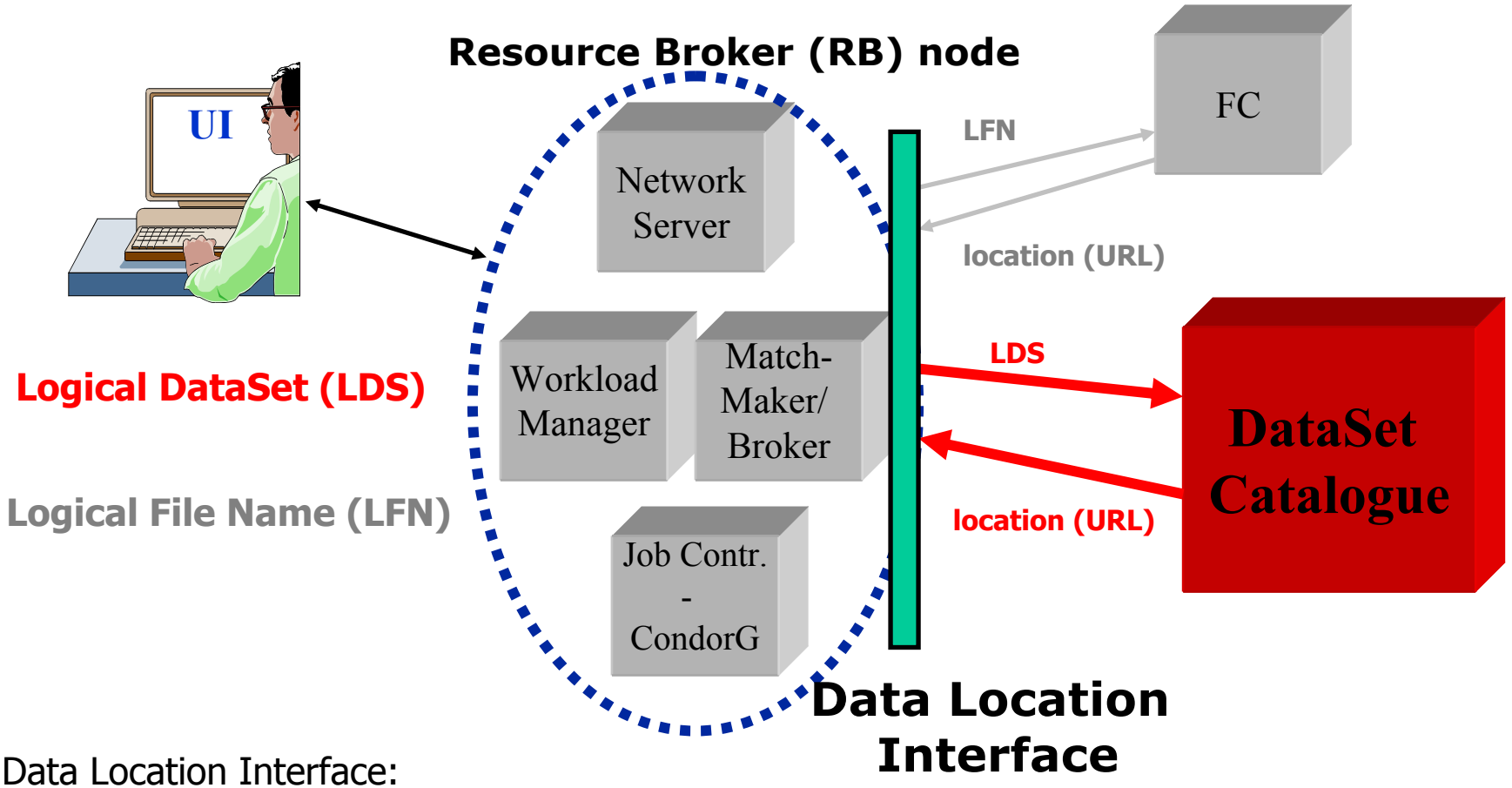
Enabling Grids for E-science in Europe

- A user of the WMS may prefer 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).
- JDL expression:

  InputData = {"lds:mu03_tt_4mu/mu_Hit245_2_g133"}

  ReplicaCatalog={“http://hostname.cern.ch:8085/“}
- A different catalog can be used based on experiment (VO) needs

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

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

Interactive Job

Enabling Grids for E-science in Europe

- The Interactive job is a job whose standard streams are forwarded to the submitting client
- The user has to set the JDL JobType attribute to interactive
- When an interactive job is submitted, the edg-job-submit command
  - starts a Grid console shadow process in the background that listens on a port assigned by the Operating System
    - The port can be forced through the ListenerPort attribute in the JDL
  - opens a new window where the incoming job streams are forwarded
- The DISPLAY environment variable has to be set correctly, because an X window is open
- The user can specify --nogui option, which makes the command provide a simple standard non-graphical interaction with the running job
- It is not necessary to specify the OutputSandbox attribute in the JDL because the output will be sent to the interactive window

Logical Checkpointing Job

Enabling Grids for E-science in Europe

- The Checkpointing job is a job that can be decomposed in several steps
- In every step the job state can be saved in the LB and retrieved later in case of failures
- The job state is a set of pairs <key, value> defined by the user
- The job can start running from a previously saved state and not from the beginning again
- The user has to set the JDL JobType attribute to checkpointable

Logical Checkpointing Job

- When a checkpointable job is submitted and starts from the beginning, the user run simply the edg-job-submit command
  - the number of steps, that represents the job phases, can be specified by the JobSteps attribute
    - e.g. JobSteps = 2;
  - the list of labels, that represents the job phases, can be specified by the JobSteps attribute
    - e.g. JobSteps = {“genuary”, “february”};
- The latest job state can be obtained by using the edg-job-get-chkpt <jobid> command
- A specific job state can be obtained by using the edg-job-get-chkpt –cs <state_num> <jobid> command
- When a checkpointable job has to start from an intermediate job state, the user run the edg-job-submit command using the –chkpt <state_jdl> option where <state_jdl> is a valid job state file, where the state of a previously submitted job was saved

Other (most relevant) UI commands

- `edg-job-attach`
  - Starts an interactive session for previously submitted interactive jobs
  - Srarts a listener process on the UI machine
- `edg-job-get-chkpt`
  - Allows the user to retrieve one or more checkpoint states by a previously submitted job

MPI Job

Enabling Grids for E-science in Europe

- There are a lot of libraries supporting parallel jobs, but we decided to support MPICH.
- The MPI job is run in parallel on several processors
- The user has to set the JDL JobType attribute to MPICH and specify the NodeNumber attribute that's the required number of CPUs
- When a MPI job is submitted, the UI adds
  - in the Requirements attribute

    `Member("MpiCH", other.GlueHostApplicationSoftwareRunTimeEnvironment)` (the MPICH runtime environment must be installed on the CE)

    `other.GlueCEInfoTotalCPUs >= NodeNumber` (a number of CPUs must be at least be equal to the required number of nodes)
  - In the Rank attribute

    `other.GlueCEStateFreeCPUs` (it is chosen the CE with the largest number of free CPUs)

MPI Job

```
[
JobType = "MPICH";
NodeNumber = 2;
Executable = "MPItest.sh";
Argument = "cpi 2";
InputSandbox = {"MPItest.sh", "cpi"};
OutputSandbox = "executable.out";
Requirements = other.GlueCEInfoLRMSType == “PBS” ||
other.GlueCEInfoLRMSType == “LSF”;
]
```

- The NodeNumber entry is the number of threads of MPI job
- The MPItest.sh script only works if PBS or LSF is the local job manager

MPI jobs

Enabling Grids for E-science in Europe

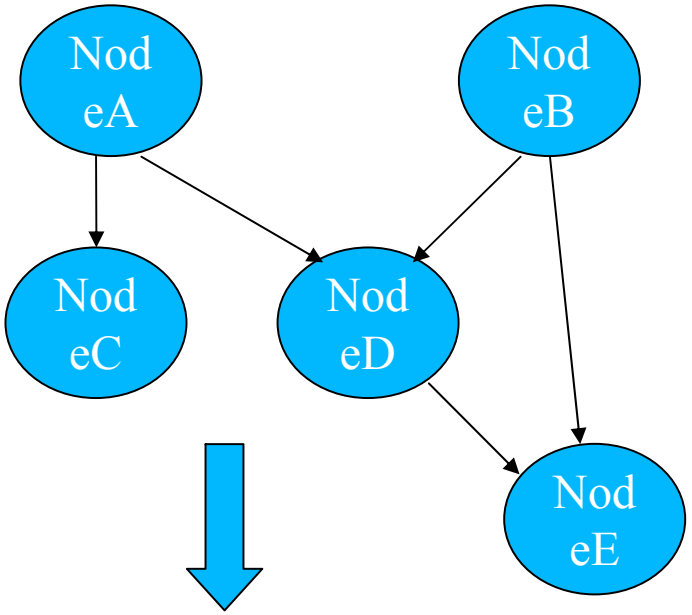
- Snapshot of MPItest.sh:

```
# $HOST_NODEFILE contains names of hosts allocated for MPI job
for i in `cat $HOST_NODEFILE` ; do
        echo "Mirroring via SSH to $i"
# creates the working directories on all the nodes allocated for parallel execution
        ssh $i mkdir -p `pwd`
# copies the needed files on all the nodes allocated for parallel execution
        /usr/bin/scp -rp ./* $i:`pwd`
# sets the permissions of the files
        ssh $i chmod 755 `pwd`/$EXE
        ssh $i ls -alR `pwd`
done
#  execute the parallel job with mpirun
mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE `pwd`/$EXE > executable.out
```

- Important: you need shared keys between worker nodes
  - Avoids sharing of home directories
  - Enforced in GILDA
  - NOT enforced in LCG2 … The VO needs to negotiate on a site by site basis

What is a DAG

- DAG means Directed Acyclic Graph
- Each node represents a job
- Each edge represents a temporal dependency between two nodes
  - e.g. NodeC starts only after NodeA has finished
- A dependency represents a constraint on the time a node can be executed
- Dedendencies are represented as “expression lists” in the ClassAd language




```
dependencies = {
{NodeA, {NodeC, NodeD}},
{NodeB, NodeD},
{NodeB, NodeD}, NodeE}
}
```

DAG Job

Enabling Grids for E-science in Europe

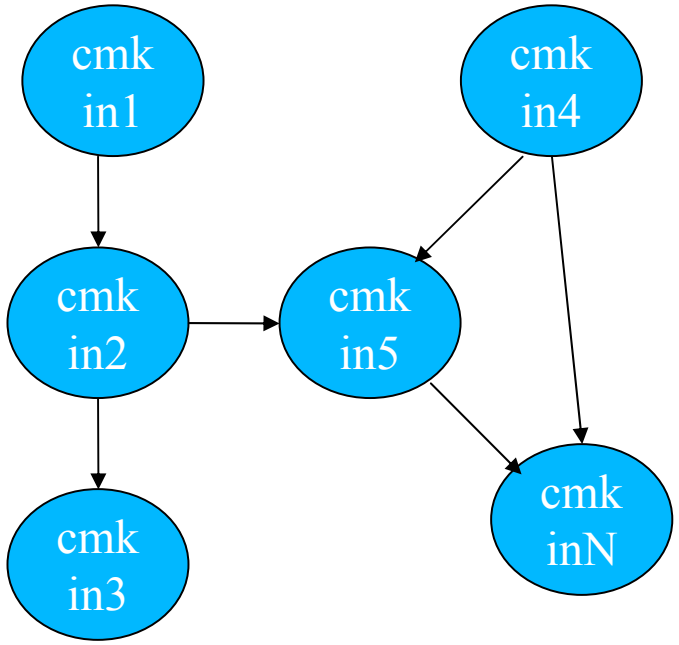
- The DAG job is a Directed Acyclic Graph Job
- The sub-jobs are scheduled only when the corresponding DAG node is ready
- The user has to set the JDL JobType attribute to dag, nodes attributes that contains the description of the nodes, and dependencies attributes

NOTE:

- A plug-in has been implemented to map an EGEE DAG submission to a Condor DAG submission
- Some improvements have been applied to the ClassAd API to better address WMS need

DAG Job

```
nodes = {
   cmkin1 = [
      file = "bckg_01.jdl" ;
   ],
   cmkin2 = [
      file = "bckg_02.jdl" ;
   ],
   ……
   cmkinN = [
      file = "bckg_0N.jdl" ;
   ]
};
dependencies = {
   {cmkin1, cmkin2},
   {cmkin2, cmkin3},
   {cmkin2, cmkin5},
   {{cmkin4, cmkin5}, cmkinN}
}
```



Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

WMS APIs

Enabling Grids for E-science in Europe

- The WMS makes C++ and Java APIs available for UI, LB consumer and client.
  - Python libraries also available
    - Created from C++ APIs using Swig
- In the following document:

  http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0118-1_2.pdf

  details about the rpms containing the APIs are given.
- Correspondent doxygen documentation can be found in share/doc area. Ex.:
  - $EDG_LOCATION/share/doc/edg-wl-ui-api-cpp-lcg2.1.49/html
- Look also at:
  - http://grid-deployment.web.cern.ch/grid-deployment/eis/tutorial/edg-wl-ui-api/index.html
  - http://grid-deployment.web.cern.ch/grid-deployment/eis/tutorial/edg-wl-common-api/index.html

"JobAd" Class

Enabling Grids for E-science in Europe

Representation of the job description in the JDL language

String and Stream Constructor/Destructor

- void **fromStream** (istream &jdl_in)
  - *Update the JobAd object with the given input stream.*
- string **toString** ()
  - *Convert the JobAd Instance into a single line string representation.*
- string **toString** (const string &attr_name)
  - *Retrieve the string representation of an attribute.*
- string **toSubmissionString** ()
  - *Convert the JobAd Instance into a single line string representation ready for submission.*
- void **toFile** (const string &file_path)
  - *Put the JobAd Instance as a string into a file.*

Insertion Methods

- void **setAttribute** (const string &attr_name, const string &attr_value)
  - *Set an attribute key-value pair*
- void **addAttribute** (const string &attr_name,const, string &attr_value)
  - *Add an value to an attribute*
- void **setAttributeExpr** (const string &attr_name, const string &attr_value)
  - *Add The specified Expression Attribute to the jdl istance.*

"JobAd" Class

Enabling Grids for E-science in Europe

Retrieval Methods

- string **getString** (const string &attr_name)
  - *Retreive the value of the specified attribute.*
- int **getInt** (const string &attr_name)
  - *Retreive the value of the specified attribute.*
- double **getDouble** (const string &attr_name)
  - *Retreive the value of the specified attribute.*
- bool **getBool** (const string &attr_name)
  - *Retreive the value of the specified attribute.*

Miscellaneous Methods

- void **checkSyntax** (const string &attr_name, classad::ExprTree *attr_value)
  - *Check if the couple attribute/value is admitted.*
- void **checkMultiAttribute** (const vector< string > &multi)
  - *Check if the Member/isMember expression is properly used in rank and requirements attributes expressions.*
- classad::ExprTree * **delAttribute** (const string &attr_name)
  - *Delete an Attribute.*
- void **check** ()
  - *Check the JobAd instance for both syntax and semanthic errors.*

- Namespace: **edg::workload::common::requestad**

“JDL” class (attributes)

Enabling Grids for E-science in Europe

- edg/workload/common/requestad/JDLAttributes.h
- Namespace: edg::workload::common::requestad

```
class JDL {
public:
  static const string REQUIREMENTS;
  static const string FUZZY_RANK;
  static const string EXITCODE;
  static const string NODENUMB;
  static const string SHPORT;
  static const string RETRYCOUNT;
  static const string CE_MATCH;
  static const string CHKPT_STEPS;
  static const string CHKPT_CURRENTSTEP;
  static const string RANK;
  static const string NOTIFYTYPE;
  static const string JOBSTATUS;
[…]
}
```

"Job" Class

Enabling Grids for E-science in Europe

Namespace: **edg::workload::userinterface**

- JobId * getJobId()
  - *Get the JobId instance*
- JobAd * getJobAd()
  - *Get the JobAd instance.*
- void setCredPath(const string cp)
  - *Set a different Proxy certificate from the default one.*
- void unsetCredPath()
  - *Set the Proxy certificate as default.*
- void setLoggerLevel(int level)
  - *Set the verbosity level for NS debug default value*
- void setJobAd(const JobAd &ad)
  - *Set the JobAd instance.*
- void setJobId(const JobId &id)
  - *Set the JobId instance.*

"Job" Class

- Status **getStatus** (bool ad=true)
  - *Retrieve the status of the job.*
- Event **getLogInfo** ()
  - *Retrieve the bookkeeping information of the job.*
- void **submit** (const string &nSHost, int nsPort, const string &lbHost, int lbPort,const string &ce_id="")
  - *Submit the job to the Network Server.*
- vector< pair< string,double > >**listMatchingCE** (const string &host, int port)
  - *Look for matching Computing Element available resources.*
- ResultCode **cancel** ()
  - *Cancel the job from the Network Server.*
- void **getOutput** (const string &dir_path)
  - *Retrieve output files of a submitted job (Success Done status has to be reached).*

"JobId" Class

Enabling Grids for E-science in Europe

- void **fromString** (const string &dg_JobId)
  - *sets the JobId instance from the JobId in string format given as input.*
- string **toString** () const
  - *Converts the jobId into a string.*
- void **clear** ()
  - *Unsets the JobId instance.*
- bool **isSet** ()
  - *Check wheater the jobId has been already created (true) or not (false).*
- void **setJobId** (const string &lb_server, int port=0, const string &unique="")
  - *Set the JobId instance according to the LB and RB server addresses and the unique string passed as input parameters.*

- Namespace: **edg::workload::common::jobid**

"JobStatus" class

Enabling Grids for E-science in Europe

- Namespace: **edg::workload::logging::client**

Public Member Values:

- enum **Code** {UNDEF = 0, SUBMITTED, WAITING, READY, SCHEDULED, RUNNING, DONE, CLEARED, ABORTED, CANCELLED, UNKNOWN, PURGED, CODE_MAX };
- enum **Attr** {[…]}
  - *Job attributes, like worker node where the job is executed (CE_NODE), return code (DONE_CODE), type of job (JOBTYPE) etc …*
- enum **AttrType** { INT_T,STRING_T,TIMEVAL_T,BOOL_T,JOBID_T, …};
  - *Type of attributes*
- Code **status**;
  - *Numeric status code*

Public Methods:

- const string & **name** (void) const;
  - *String representation of the status code*
- const vector<pair<Attr,AttrType> >& **getAttrs** (void) const;
  - *List of attributes and types valid for this instance*
- const std::string& **getAttrName** (Attr) const;
  - *Attribute name*

"JobStatus" class & "Event" class

Enabling Grids for E-science in Europe

Several retrieve methods, depending of the AttrType:

- int getValInt (Attr) const;
- string getValString (Attr) const;
- struct timeval getValTime (Attr) const;
- const JobId getValJobId (Attr) const;
- bool getValBool (Attr) const;
- const vector<string> getValStringList (Attr) const;
- const vector<pair<string,string> > getValTagList (Attr) const;
- const vector<JobStatus> getValJobStatusList (Attr) const;

- The class "Event" has structure similar to "JobStatus"
  - It is used to retrieve the Logging Info

"Exception" Class

Enabling Grids for E-science in Europe

- Namespace: edg::workload::common::utilities

  - Pure virtual
  - It inherits std::exceptions

Public Methods:

  - virtual string dbgMessage ();
    - *Return a string debug message containing information about Exception thrown*
  - virtual int getCode ();
    - *Return the Code number*
  - virtual const char* what () const throw ();
    - *return the Error Message associated to the Exception*
  - virtual void log (const string& logfile = "");
    - *Print Exception error information into a log file*
  - virtual string getExceptionName ();
    - *Return the name of the Exception raised*
  - virtual string printStackTrace () ;
    - *Return the list of methods that caused the Exception*

WMS APIs exampke

Enabling Grids for E-science in Europe

```
% ./workload Hello.jdl lxb0704.cern.ch 7772 lxb0704.cern.ch 9000
```


```
#include <iostream>
#include <string>

#include "edg/workload/logging/client/JobStatus.h"
#include "edg/workload/common/utilities/Exceptions.h"
#include "edg/workload/common/requestad/JobAd.h"
#include "edg/workload/userinterface/client/Job.h"


using namespace std ;
using namespace edg::workload::common::utilities ;
using namespace edg::workload::logging::client ;
/* ************************************************************************
 *  Example based on edg-wl-job-submit.cpp, edg-wl-job-status.cpp
 *  for further examples see also:

 http://isscvs.cern.ch:8180/cgi-bin/cvsweb.cgi/workload/userinterface/test/?cvsroot=lcgware

 *
 *  author: Heinz.Stockinger@cern.ch
 *
 *  Example usage on GILDA:
 *  ./workload Hello.jdl grid004.ct.infn.it 7772 grid004.ct.infn.it 9000
 *
 */
```

More examples in CVS

WMS APIs example

```
int main (int argc,char *argv[])
{

 cout << "Workload Management API Example " << endl;

 try{
  if (argc < 6 ||  strcmp(argv[1],"--help") == 0)  {
    cout << "Usage : " << argv[0]
              << "  <JDL file>  <ns host> <ns port> <lbHost> <lbPort> [<ce_id>]"
              << endl;
    return -1;
  }

   edg::workload::common::requestad::JobAd jab;

   jab.fromFile ( argv[1] ) ;
   edg::workload::userinterface::Job   job(jab);
   job.setLoggerLevel (6) ;

   cout << "Submit job to " << argv[2] << ":" <<  argv[3]  << endl;
   cout << "LB address: "<< argv[4] << ":" <<  argv[5]  << endl;
   cout << "Please wait..." << endl;

   // We now submit the job. If a CE is given (argv[6]), we send it directly
   // to the specified CE
   //
   if (argc ==6)
     job.submit (argv[2], atoi(argv[3]), argv[4], atoi(argv[5]),  "") ;
   else
     job.submit (argv[2], atoi(argv[3]), argv[4], atoi(argv[5]), argv[6] ) ;

   cout << "Job Submission OK; JobID= "
             << job.getJobId()->toString()  << endl << flush ;
```

Job Definition

Job Submission

- The JobAd class provides users with management operations on JDL files
- We instantiate a Job object that corresponds to our JDL file and handles our job

WMS APIs example

Enabling Grids for E-science in Europe

```
// Retrieve the status of the job //

JobStatus status = job.getStatus();

[…]
 // Print some detailed error information in case the job did not
    // succeed.
    //
    if ((status.status == 8) || (status.status == 9)) {
      printStatus(status);
      exit(-1);
    }

[…]
    cout << "\nThe job finished successfully“ << endl;

    return 0;

  } catch (Exception &exc){
    cerr << "\nWMS Error\n";
    cerr <<  exc.printStackTrace();
  }
  return -1;
}
```

Get info about state of the job

Handle Exceptions

The job finished successfully. We can retrieve the output.

WMS APIs example

Enabling Grids for E-science in Europe

```
CC          = gcc-3.2.2
GLOBUS_FLAVOR  = gcc32


ARES_LIBS =  -lares
BOOST_LIBS = -L/opt/boost/gcc-3.2.2/lib/release -lboost_fs \
                -lboost_thread -lpthread -lboost_regex
CLASSAD_LIBS = -L/opt/classads/gcc-3.2.2/lib -lclassad
EXPAT_LIBS =  -lexpat
GLOBUS_THR_LIBS =  -L/opt/globus/lib -lglobus_gass_copy_gcc32dbgpthr \
    -lglobus_ftp_client_gcc32dbgpthr -lglobus_gass_transfer_gcc32dbgpthr \
    -lglobus_ftp_control_gcc32dbgpthr -lglobus_io_gcc32dbgpthr \
    -lglobus_gss_assist_gcc32dbgpthr -lglobus_gssapi_gsi_gcc32dbgpthr \
    -lglobus_gsi_proxy_core_gcc32dbgpthr \
    -lglobus_gsi_credential_gcc32dbgpthr \
    -lglobus_gsi_callback_gcc32dbgpthr -lglobus_oldgaa_gcc32dbgpthr \
   -lglobus_gsi_sysconfig_gcc32dbgpthr \
   -lglobus_gsi_cert_utils_gcc32dbgpthr \
   -lglobus_openssl_gcc32dbgpthr -lglobus_proxy_ssl_gcc32dbgpthr \
   -lglobus_openssl_error_gcc32dbgpthr -lssl_gcc32dbgpthr \
   -lcrypto_gcc32dbgpthr -lglobus_common_gcc32dbgpthr

GLOBUS_COMMON_THR_LIBS =  -L/opt/globus/lib -L/opt/globus/lib \
                                -lglobus_common_gcc32dbgpthr
GLOBUS_SSL_THR_LIBS =  -L/opt/globus/lib -L/opt/globus/lib \
                            -lssl_gcc32dbgpthr -lcrypto_gcc32dbgpthr
VOMS_CPP_LIBS = -L/opt/edg/lib -lvomsapi_gcc32dbgpthr

all: workload
```

Makefile

```
workload: workload.o
        $(CC) -o workload \
        -L${EDG_LOCATION}/lib -ledg_wl_common_requestad \
                –lpthread \
                -ledg_wl_userinterface_client \
        -ledg_wl_exceptions -ledg_wl_logging \
                -ledg_wl_loggingpp \
        -ledg_wl_globus_ftp_util -ledg_wl_util \
                -ledg_wl_common_requestad \
        -ledg_wl_jobid -ledg_wl_logger -ledg_wl_gsisocket_pp \
        -ledg_wl_checkpointing -ledg_wl_ssl_helpers \
                -ledg_wl_ssl_pthr_helpers \
        $(VOMS_CPP_LIBS) \
        $(CLASSAD_LIBS) $(EXPAT_LIBS) $(ARES_LIBS) \
        $(BOOST_LIBS) \
        $(GLOBUS_THR_LIBS) \
        $(GLOBUS_COMMON_THR_LIBS) \
        $(GLOBUS_SSL_THR_LIBS) \
        workload.o

workload.o: workload.cpp
        $(CC) -I ${EDG_LOCATION}/include \
        -I/opt/classads/gcc-3.2.2/include -c workload.cpp

clean:
        rm -rf workload workload.o
```

Contents

Enabling Grids for E-science in Europe

- The Workload Management System
- Job Preparation
  - Job Description Language
- Job submission and job status monitoring
- WMS Matchmaking
- Different job types
  - Interactive jobs
  - MPI jobs
  - DAG jobs
- APIs Overview
- edg-brokerinfo

edg-brokerinfo

Enabling Grids for E-science in Europe

- BrokerInfo file is created by the RB:
  - Contains various information concerning a job
    - chosen CE, location of the considered data …
  - Shipped to the WN with the job
    - can be used by the applications
  - Wrote relying on the Condor ClassAds
    - not be so easy to read and interpret it
- edg-brokerinfo:
  - Tool allowing users to parse this file
  - Command line tool and a C++ API
- To use edg-brokerinfo the *EDG_WL_RB_BROKERINFO* environmental variable must be defined
  - refers to the pathname of the .BrokerInfo file.
  - In the worker node is “automatically” defined

The BrokerInfo command

- **Command line:**
  - **edg-brokerinfo [-v] [-f <filename>] function [parameter] [parameter] …**
    - [-v] produces more verbose output
    - [-f <filename>] specifies the BrokerInfo file to be parsed

- **C++ API:** can look at the documentation:
  - http://server11.infn.it/workload-grid/docs/edg-brokerinfo-user-guide-v2_2.pdf

The BrokerInfo command

- 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

Enabling Grids for E-science in Europe

```
[
   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"
]
```

Summary

Enabling Grids for E-science in Europe

- We explained the main functionality of the Workload Management System
- The JDL file describes a user job
- A set of commands allow the user to get status information and retrieve relevant data
- APIs are available in C++ and Java for UI, and LB.
- edg-brokerinfo allows to parse the Brokerinfo file

Bibliography

Enabling Grids for E-science in Europe

- LCG-2 User Guide (html,ps,pdf)
  - https://edms.cern.ch/file/454439//LCG-2-UserGuide.pdf
- LCG-2 Middleware (html,ps,pdf)
  - https://edms.cern.ch/file/498079/0.1/LCG-mw.pdf
- Brokerinfo:
  - http://server11.infn.it/workload-grid/docs/edg-brokerinfo-user-guide-v2_2.pdf
- LCG-2 User Scenario (html,ps,pdf)
  - https://edms.cern.ch/file/498081/1.0/UserScenario2.pdf