Submitting Jobs to the Grid from UCL

Ben West

- Overview
- The Logging and Bookkeeping Service
- The User Interface
  - Submitting a job
- The Resource Broker
  - RB Master
  - RB Agent
- The Job Submission Service
  - The Job Description Language
- Security
- UI Installation

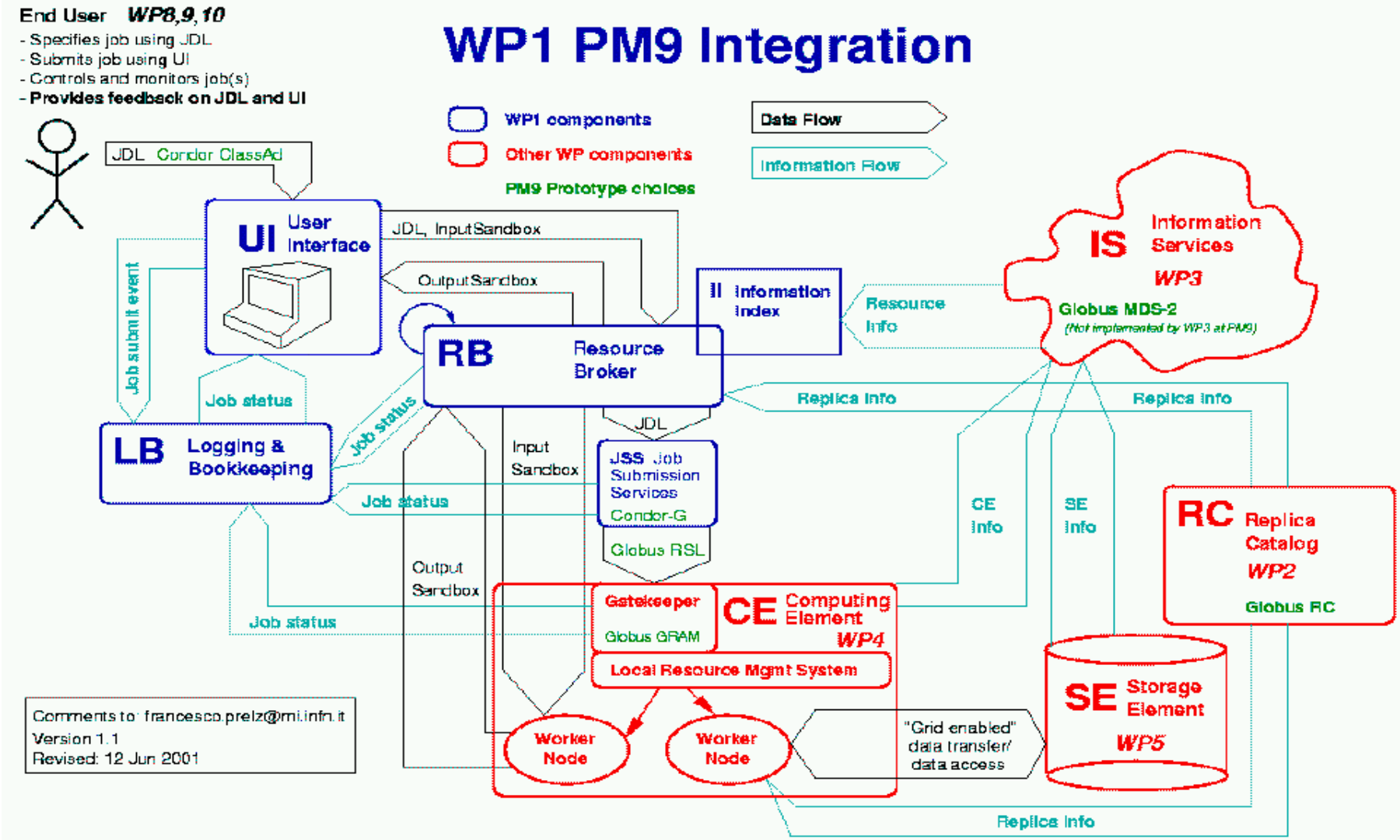
Acronyms

- UI – User Interface
- RB – Resource Broker
- II – Information Index
- JSS – Job Submission Service
- JDL – Job Description Language
- LB – Logging & Bookkeeping
- RC – Replica Catalogue
- SE – Storage Element
- LFN – Logical File Name
- PFN – Physical File Name
- CE – Computing Element
- LRMS – Local Resource Management System (e.g. pbs)
- WN – Worker Node
- WMS – Workload Management System (i.e. WP1)
- IS – Information Services
- VO – Virtual Organisation

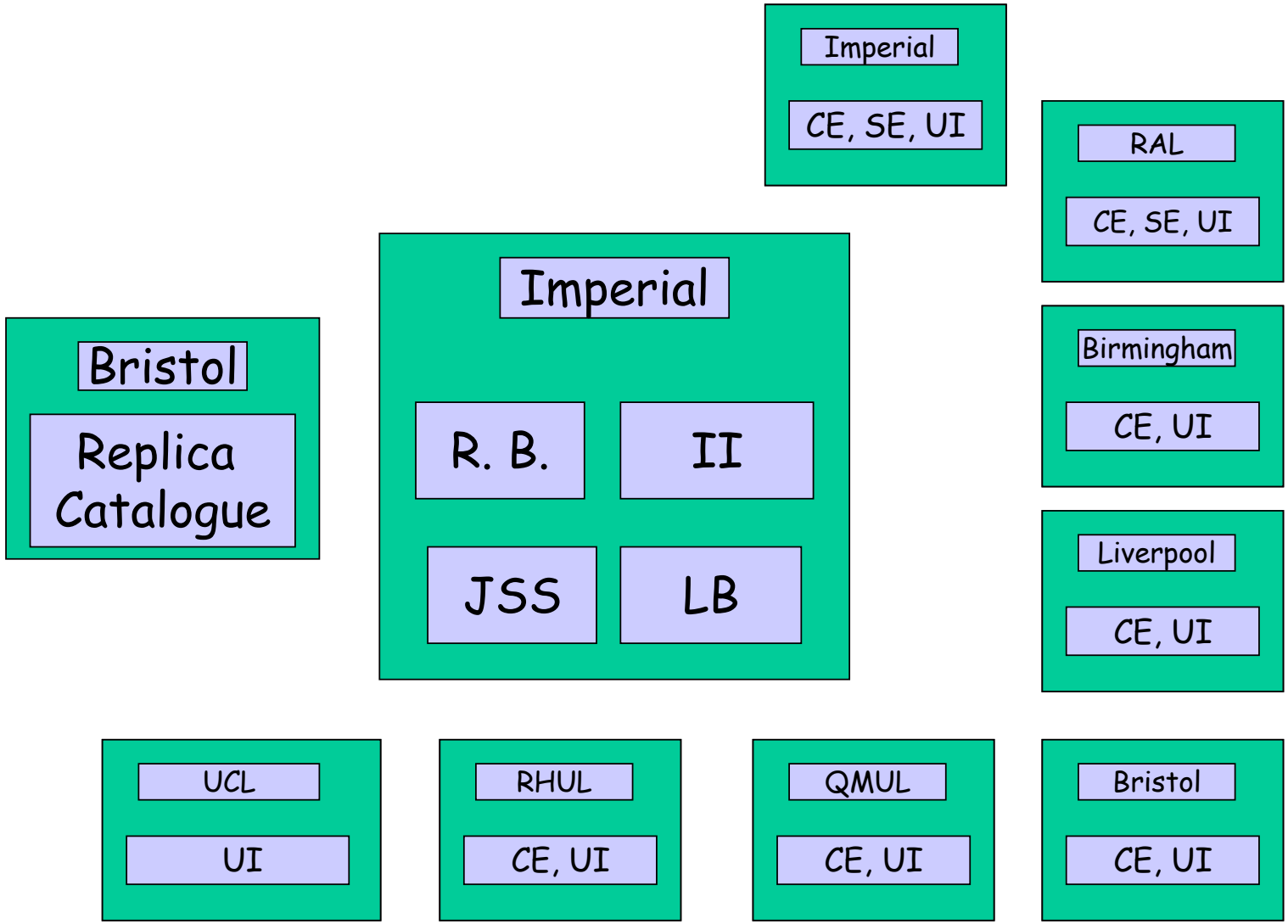
Overview…

- Running a job on the Grid…
  1. Create a JDL file describing your job
  2. Submit the JDL to an RB using a convenient UI.
  3. The RB matches the request to a suitable CE by querying:
     - Replica Catalogue (LFN → list of PFNs on SEs)
     - Information Index (CE & SE information c.f. Information Services)
  4. The RB sends the JDL to the JSS.
  5. The JSS translates the request (into Globus RSL) and passes it to the assigned CE's Gatekeeper
  6. The CE's Gatekeeper passes the job onto the Local Resource Management System (e.g. pbs)
  7. The LRMS passes the job onto a worker node
  8. The worker node gets the input sandbox from the RB and any necessary data from SEs
  9. The completed job sends its output data to an SE and its output sandbox to the RB
  10. The use can now collect their output sandbox from the RB, and access their data on the SE

Overview…


End User WP8,9,10
- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI
WP1 PM9 Integration
WP1 components
Other WP components
PM9 Prototype choices
Data Flow
Information Flow
JDL Condor ClassAd
UI User Interface
JDL, InputSandbox
OutputSandbox
Job submit event
Job status
II Information Index
Resource Info
IS Information Services
WP3
Globus MDS-2
(Not implemented by WP3 at PM9)
RB Resource Broker
Replica Info
Replica Info
JDL
LB Logging & Bookkeeping
Job status
Job status
Input Sandbox
JSS Job Submission Services
Condor-G
Globus RSL
Job status
CE Info
SE Info
RC Replica Catalog
WP2
Globus RC
Output Sandbox
Gatekeeper
Globus GRAM
CE Computing Element
WP4
Job status
Local Resource Mgmt System
SE Storage Element
WP5
Worker Node
Worker Node
"Grid enabled" data transfer/ data access
Replica Info
Comments to: francesco.prelz@mi.infn.it
Version 1.1
Revised: 12 Jun 2001

The UK Testbed



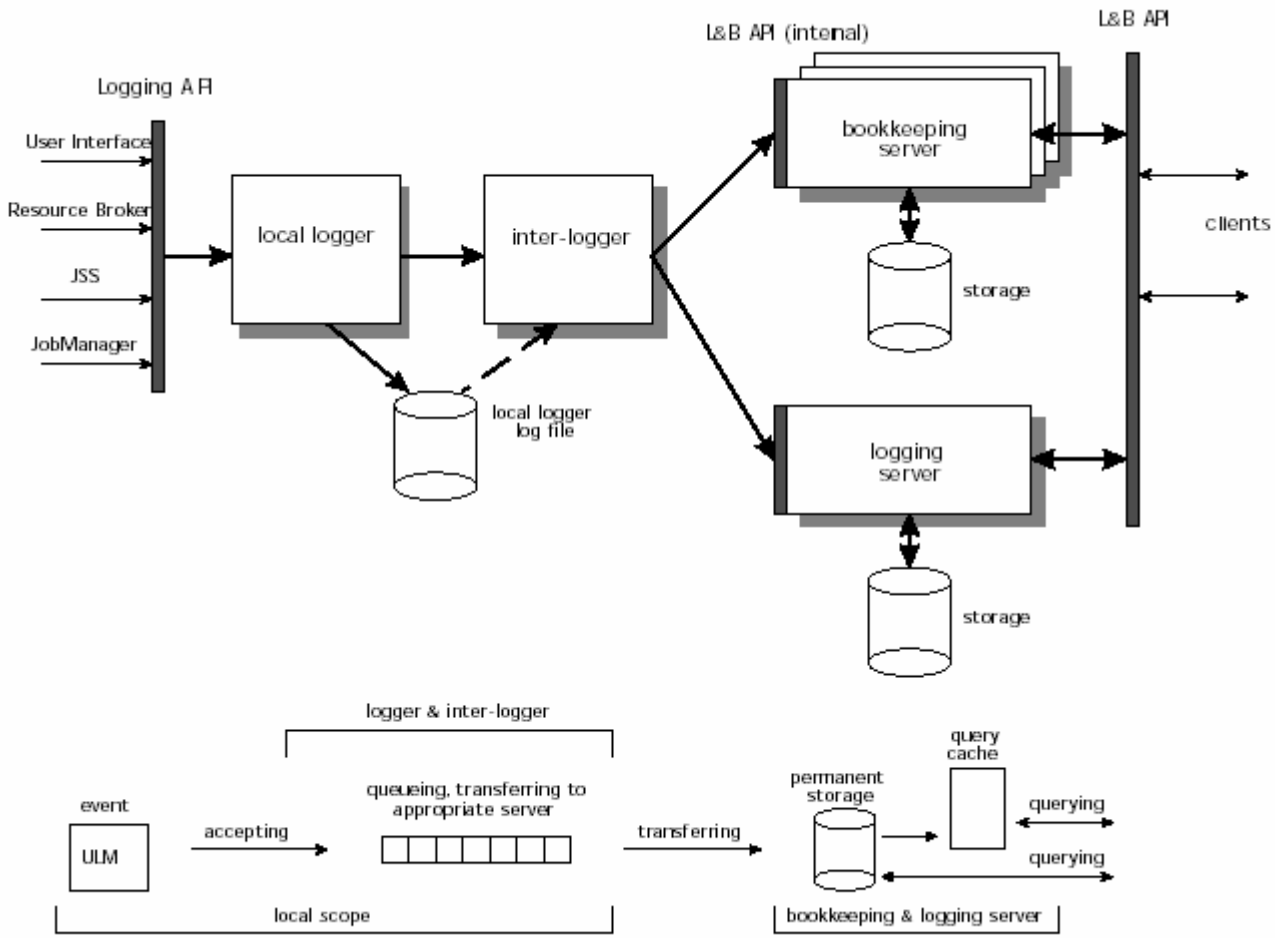
Logging and Bookkeeping

- Every stage of the process is recorded by the logging and bookkeeping service.
  - The bookkeeping service provides users with information about their current jobs storing short-term data.
  - The logging service stores long-term data about jobs and the scheduling system itself and is intended for scheduler debugging and post mortem analysis of jobs.
  - Some data are stored by both the bookkeeping and logging services
- The level of verbosity can be set by the user (though only to a point)
- All jobs are identified by a unique *dg_jobid* which is generated by the User Interface upon job submission. This *dg_jobid* is a valid URL and a https get on it should (eventually) return XML based job status.

Logging and Bookkeeping

- The main components of the Logging and Bookkeeping Service are:
  - Logging API – API used by event sources to pass messages to the LB
  - Logging and bookkeeping server API – API used for querying the L&B system
  - Local logger – process which accepts messages from their sources (via logging API) and passes the to the L&B subsystem. Persistence is implemented by a simple transaction log file.
  - Inter-logger – Process responsible for transferring data to bookkeeping or logging server. Maintains message queues and allows for possible communication problems.
  - Bookkeeping server – accepts messages from the inter-logger and manages primary data storage. Also handles user queries.
  - Logging server – like bookkeeping server but deals with persistent type log messages (info is not analysed or managed in anyway, except to provide query access)

Logging and Bookkeeping



The User Interface

- The UI is designed to be portable, configurable, secure and lightweight means of communicating with both the Resource Broker (RB):
  - Job submission
  - Job cancellation
  - Job output collection

  and Logging & Bookkeeping services
  - Job status
  - Job logging info
- UI based on two elements:
  - A language to describe characteristics, requirements & preferences of a job (also used to describe the resources)
  - A set of commands to manage jobs on the Grid
- Job management currently implemented via command line invoked Python scripts and sandbox transfers are executed by the UI using gsiftp (so the UI only needs gsiftp client software)

The User Interface – still to come

- Job management via:
  - Application APIs – edg v1.3
  - GUI – early version already exists

- Job partitioning (edg v2.0)
  - Ability to send a single large job that will split and run on many CE's
- Job dependencies (Specification by edg v1.4)
  - Ability to submit jobs which depend on the output of other jobs
- Advanced reservation (edg v2.0)
  - Ability to pre-book time on CEs without submitting a job

Submitting a Job

dg-job-submit <job_description_file>
[-input input_file] [-resource CE_id]
[-notify e_mail_address] [-config path_name]
[-output output_file] [-hours H] [-nomsg] [-noint]
[-debug]

```
[bjw@pc18 bjw]$ dg-job-submit test.jdl

Connecting to host gm04.hep.ph.ic.ac.uk, port 7771
Logging to host gm04.hep.ph.ic.ac.uk, port 15830

**********************************************************************************
                        JOB SUBMIT OUTCOME
 The job has been successfully submitted to the Resource Broker.
 Use dg-job-status command to check job current status. Your job identifier (dg_jobId) is:

 - https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/103725216672970?gm04.hep.ph.ic.ac.uk:7771

**********************************************************************************
```

Submitting a Job – A simple JDL file

```
Executable      = "test.exe";
Arguments       = "100";
StdOutput       = "test.out";
StdError        = "test.err";
InputSandbox    = {"test.exe"};
OutputSandbox = {"test.out","test.err", "test.exe"};
```

- Files transferred to and from the job using the Sandboxes
  - Only intended for small files (e.g. config, logs, errors)
  - Anything more should use Grid services…
- JDL can also be used to describe requirements and job submission preferences, more later…

Submitting a Job – Choosing a CE

- Unless you specify a Compute Element, the Resource Broker will assign the job to a CE based on user preferences expressed in the JDL file (more later…)
- To find out what resources are available matching your requirements (again expressed in the JDL) the command dg-job-list-match can be used

```
[bjw@pc18 bjw]$ dg-job-list-match test.jdl

Connecting to host gm04.hep.ph.ic.ac.uk, port 7771

**********************************************************************
                COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:

- gw39.hep.ph.ic.ac.uk:2119/jobmanager-pbs-workq
- gppce05.gridpp.rl.ac.uk:2119/jobmanager-pbs-S
- gppce05.gridpp.rl.ac.uk:2119/jobmanager-pbs-M
- gppce05.gridpp.rl.ac.uk:2119/jobmanager-pbs-L

**********************************************************************
[bjw@pc18]$ dg-job-submit test.jdl -resource gppce05.gridpp.rl.ac.uk:2119/jobmanager-pbs-S
```

Submitting a Job - What is dg_jobid?

- A job can be referred to using several identifiers depending on context (local_jobid, jss_jobid, dg_jobid) but only dg_jobid is uniquely identifiable across the whole Grid.
- dg-job-id-info <dg_JobId1...dg_JobIdN | -input input_file>

```
[bjw@pc18 bjw]$ dg-job-id-info
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/103725216672970?gm04.hep.ph.ic.ac.uk:7771

**********************************************************************************
JOB ID INFO
Printing info for the Job ID :
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/103725216672970?gm04.hep.ph.ic.ac.uk:7771

Logging and Bookkeeping Server Address  =    https://gm04.hep.ph.ic.ac.uk
Logging and Bookkeeping Server Port     =    7846
Resource Broker Server Address          =    gm04.hep.ph.ic.ac.uk
Resource Broker Server Port             =    7771
Submission Time (hh:mm:ss)              =    10:37:25 (UTC)
User Interface Machine IP Address       =    128.40.4.108
User Interface Process Identifier       =    21667
Randomly Generated Number (0000-9999)   =    2970
**********************************************************************************
```

Submitting a Job – Job status

dg-job-status <dg_JobId1 ... dg_JobIdN | -all |-input input_file> [-full] [-config path_name] [-output output_file] [-noint] [-debug]

```
[bjw@pc18 bjw]$  dg-job-status
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/103725216672970?gm04.hep.ph.ic.ac.uk:7771

**********************************************************
BOOKKEEPING INFORMATION:

Printing status info for the Job :
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/110418218359397?gm04.hep.ph.ic.ac.uk:7771

dg_JobId                  =
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/110418218359397?gm04.hep.ph.ic.ac.uk:7771
Status                    =   OutputReady
Last Update Time          =   Thu Apr 18 11:12:30 2002
Status Reason             =   terminated
Job Owner                 =   /O=Grid/O=UKHEP/OU=hep.ucl.ac.uk/CN=Ben West
Status Enter Time         =   Thu Apr 18 11:12:30 2002
**********************************************************
```

- Ready→Scheduled→Running→Done→OutputReady→Cleared
- Lost dg_jobid? Use –all to get a list of all your jobs

Submitting a Job – Job collection

dg-job-get-output <dg_JobId1...dg_JobIdN | -input input_file>
[-dir directory_path] [-config path_name]
[-noint] [-debug]

```
[bjw@pc18 bjw]$  dg-job-get-output
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/103725216672970?gm04.hep.ph.ic.ac.uk:7771

**********************************************************
JOB GET OUTPUT OUTCOME

 Output sandbox files for the job:
 - https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/111046274447062?gm04.hep.ph.ic.ac.uk:7771
 have been successfully retrieved and stored in the directory:
 /tmp/111046274447062

**********************************************************************************************
```

- Sandboxes are deleted on completion of this command so put them somewhere safe!
  - Or book the log file into the same SE as the rest of your output

Submitting a Job – and finally

dg-job-cancel <dg_JobId1 ... dg_JobIdN | -all | -input input_file>
[-notify e_mail_address] [-config path_name]
[-output output_file] [-noint] [-debug]

dg-job-get-logging-info <dg_JobId1...dg_JobIdN | -all | -input
input_file> [-from T1] [-to T2] [-full] [-level]
[-config path_name] [-output file_name] [-noint] [-debug]

```
[bjw@pc18 bjw]$ dg-job-cancel
https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/14422740448442?gm04.hep.ph.ic.ac.uk:7771

Are you sure you want to remove specified job(s)? [y/n]n:y

Cancel request submitted to RB "gm04.hep.ph.ic.ac.uk". Waiting for job(s) cancellation results....

**********************************************************************************
                    JOBS CANCEL OUTCOME

 Cancel  SUCCESS  for job:
 - https://gm04.hep.ph.ic.ac.uk:7846/128.40.4.108/14422740448442?gm04.hep.ph.ic.ac.uk:7771
 The job has been successfully marked for removal

**********************************************************************************
```

The Resource Broker

- The **Resource Broker** is the core component of the workload management system
  - Provides User Interface with **job submission** and **collection** services
  - Performs **resource allocation** (finding the CE that best matches the requirements and preferences of the a submitted job taking into account the current load distribution on the Grid)
  - Communicates with **Job Submission Service** about running jobs
  - Stores all jobs in a persistent (PostgreSQL) database **Jobs Registry**
  - Updates the Logging Broker
  - Clears out completed jobs
- Based on a traditional network connected (**TCP/IP**) **client/server** model with two separate servers:
  - RB Master
  - RB Agent

Resource Broker Master

- Performs the job management aspects of the Resource Broker
- Spawns two main server threads UI-Listener and JSS-Listener
- UI-Listener
  - Listens on a well-known port for connection requests from UI machines holding them in a Pending Queue
  - Once connection is established and authenticated an RB Agent thread is executed to service the connection
- JSS-Listener
  - Listens on (another) well-known port for any call-backs from the JSS, updating the Jobs Registry when necessary
  - In the event of a job failure which can't be solved by the JSS (i.e. one where the job must be submitted to a different CE) will execute a new RB Agent thread to find a new CE for the job

Resource Broker Master

- Also spawns Logging and Pruning threads
- Logging thread
  - Once fresh information about pending jobs has been retrieved (i.e. job complete, job aborted etc.) the logging thread will send the appropriate logging event to the Logging and Bookkeeping Service
  - Note that the JSS will also log directly with the Logging Broker so there is some redundancy
- Pruning thread
  - The Output Sandbox of all completed jobs are stored on the RB
  - To avoid storage problems the pruning thread will remove the job from the Job Registry and delete its Output Sandbox

Resource Broker Agent

- Responsible for receiving and servicing client requests and may run on a different host to the RB Master
  - Resolves logical data set names
  - Finds resource where the data needed by given jobs are stored
  - Matching job requirements…
- Once created by the RB Master, the RB Agent and client communicate on a different port
  - The client sends its JDL request and Input Sandbox
  - The RB translates the JDL fields into Condor ClassAds
  - RB Agent first queries the Information Index for a set of possible candidates (the Information Index is a local cache of the Information Services).
  - Search is the refined, directly querying these candidates to get more valuable and up-to-date information (which it also translates into the Condor ClassAds)
  - Condor ClassAd libraries are used to match the job to a CE
  - Having found a CE the job is added to the Jobs Registry and sent to the Job Submission Service
  - If no CE is found the job will be refused, however if they are simply busy the job will queue on the RB until they are free

Resource Broker distribution

- How many RBs are necessary?
  - Single Grid RB – ensures optimal resource allocation, but significant scalability issues
  - Single User RB – freedom of choice, but creates unfair races for distributed resources
  - Compromise (EDG) solution is one RB per Virtual Organisation (VO), i.e. a CMS RB, an LHCb RB… currently:
    - ?.cern.ch (EDG RB in CERN, serving LHC VOs)
    - gm04.hep.ph.ic.ac.uk – GridPP RB (serving both the GridPP and BaBar VOs)

Job Submission Service

- Responsible for actual job management operations
- Uses Grid Security Infrastructure and Globus Resource Allocation and Management protocol
- Main daemon listens on a well-known port
- Works in tandem with the RB (i.e. one JSS for every RB)
- Resilient to Failure
  - Local
    - Relevant information for all submitted jobs stored persistently in a local queue
  - Remote
    - Since JSS can't rely on CEs to inform it of problems, it periodically probes all CEs on which it has jobs
- JSS currently implemented using Condor-G (marriage of Condor and Globus toolkit) to submit jobs, however all commands are wrapped allowing for this to change if necessary since condor-G is not open source

Job Submission Service

- Job submission
  - The JSS receives a JDL file and a CE address on which the job is to be run from the RB (Agent)
  - Before submitting the job the job is wrapped in another job which creates the correct environment on the CE Worker Node:
    - Downloading the input sandbox
    - Setting environment variables
    - Running job
    - Uploading the output sandbox
  - The job is the submitted to the CE's Gatekeeper by CondorG

Job Submission Service Script

```
#!/bin/sh

newdir=$$
mkdir ${newdir}
cd ${newdir}

if [ ! -w . ]; then
  echo "Working directory not writable"
  exit 1
fi

workdir="`pwd`"

if [ -z "${GLOBUS_LOCATION}" ]; then
  echo "GLOBUS_LOCATION undefined"
  exit 1
elif [ -r "${GLOBUS_LOCATION}/etc/globus-user-env.sh" ]; then
  . ${GLOBUS_LOCATION}/etc/globus-user-env.sh
else
  echo "${GLOBUS_LOCATION}/etc/globus-user-env.sh not found or unreadable"
  exit 1
fi

umask 022

for f in "test.exe" ".BrokerInfo"; do
  globus-url-copy gsiftp://gm04.hep.ph.ic.ac.uk/stage/gm04/stage/RBtmp/https:__gm04.hep.ph.ic.ac.uk:7846_128.40.4.108_10553889192656_gm04.hep.ph.ic.ac.uk:7771/input/${f} file://${workdir}/${f}
  if [ $? != 0 ]; then
    echo "Cannot download ${f} from gsiftp://gm04.hep.ph.ic.ac.uk/stage/gm04/stage/RBtmp/https:__gm04.hep.ph.ic.ac.uk:7846_128.40.4.108_10553889192656_gm04.hep.ph.ic.ac.uk:7771/input/"
    exit 1
  fi
done

if [ -e "./test.exe" ]; then
  chmod +x "./test.exe"
else
  echo "./test.exe not found or unreadable"
  exit 1
fi

EDG_WL_RB_BROKERINFO="`pwd`/.BrokerInfo"; export EDG_WL_RB_BROKERINFO

"./test.exe" $* > "test.out" 2> "test.err"

echo "job exit status = " $?
                error=0
for f in test.out test.err test.exe; do
  if [ -r "${f}" ]; then
    globus-url-copy file://${workdir}/${f} gsiftp://gm04.hep.ph.ic.ac.uk/stage/gm04/stage/RBtmp/https:__gm04.hep.ph.ic.ac.uk:7846_128.40.4.108_10553889192656_gm04.hep.ph.ic.ac.uk:7771/output/${f}
    if [ $? != 0 ]; then
      echo "Cannot upload ${f} into gsiftp://gm04.hep.ph.ic.ac.uk/stage/gm04/stage/RBtmp/https:__gm04.hep.ph.ic.ac.uk:7846_128.40.4.108_10553889192656_gm04.hep.ph.ic.ac.uk:7771/output/"
      error=1
    fi
  fi
done

cd ..
rm -rf ${newdir}
exit ${error}
```

Job Description Language

- Is the Classified Advertisement language Defined by the Condor Project for describing jobs, workstations and other resources
  - (http://www.cs.wisc.edu/condor/classad)
- Key ClassAd features:
  - Symmetric – both jobs and computing elements described through classads
  - Declarative – advertisements merely describe requirements rather than the procedure for matching
  - Simple – can be easily understood and/or automated
  - Portable – can be implemented on many hardware and software platforms

JDL – ClassAd expressions

- A ClassAd is constructed with the classad construction operator []
- It is a sequence of zero or more pairs (name, expression) separated by semi-colons.
- ClassAds can be arbitrarily nested
  - [foo=10; bar=[adr=20; adl=30]]
- Every ClassAd value has three implicit attributes references
  - self – the classad in current evaluation scope
  - parent – the lexical parent of the current evaluation scope
  - root – the classad at the root of the current evaluation scope
- For a more complete description of references see "JDL HowTo"

JDL Types

- Rich set of types including numeric, string, Boolean, timestamps, undefined and error
  - **Undefined** is generated when an attribute reference cannot be resolved
  - **Error** is generated when there are type errors
- **Lists** are constructed with {} and indexed C\C++ style
  - E.g. {10, 17*2, 30}[1]==34 is **true**
- **Comparison operators** are strict when evaluating things as undefined
  - E.g. other.MinPhysicalMemory > 32 (or <, ==, !=) will all evaluate to undefined if other does not have a MinPhysicalMemory Attribute
- **Non-strict comparisons** are achieved using the **is** and **isnt** operators
  - E.g. [Other.MinPhysicalMemory isnt undefined]
- The **Boolean operators ||** and **&&** are however non-strict on both arguments
  - E.g [other.MinPhysicalMemory > 32 || other.MaxRunningRobs > 10] will be true if one is true and one undefined

JDL – Boolean logic

AND	F	T	U	E		OR	F	T	U	E		NOT	
F	F	F	F	E		**F**	F	T	U	E		**F**	T
T	F	T	U	E		**T**	T	T	T	E		**T**	F
U	F	U	U	E		**U**	U	T	U	E		**U**	U
E	E	E	E	E		**E**	E	E	E	E		**E**	E

JDL Functions

- Many functions, described in "JDL HowTo", including
  - **Type predicates** – IsUndefined(V), IsError(V), IsString(V), IsList(V)…
  - **List Membership** – Member(V,L), IsMember(V,L)
  - **Time Queries** – CurrentTime(), TimeZoneOffset(), DayTime()
  - **Time Construction** – MakeDate(M,D,Y), MakeAbsTime(N), MakeRelTime(N)
  - **Absolute Time** – GetYear(A), GetMonth(A), GetDayOfYear(A)…
  - **Relative Time** – GetDays(R), GetHours(R)…
  - **Time Conversion** – InDays(T), InHours(T)…
  - **String Operations** – StrCat(V1,…,Vn), SubStr(S, offset, [,len]), ToUpper(S)…
  - **Type Conversion** – Int(V), Real(V), String(V)…
  - **Mathematical Operations** – Round(N)…

User provided JDL attributes

- **Executable** – This is mandatory and is an absolute path on the host machine or an executable/script name that is submitted in the input sandbox
- **Arguments** – All command line arguments for the executable in a single string
- **Environment** – A list of strings representing the necessary environment settings for the job
- **StdInput** – A String or a filename specifying the standard input to the job
- **StdOutput**, **StdError** – Filenames for these to be piped to (in order to retrieve them they must be in the output sandbox)
- **InputSandbox** – List of files on the local UI disk necessary for the job (wildcards allowed)
- **OutputSandbox** – List of files to be returned from the job (all other files will be deleted from the Worker Node and not sent to the RB, wildcards again allowed)

User provided JDL attributes

- **OutputSE** – SE where output files will be stored (also used in job matching since CE must be able to contact this SE)
- **InputData** – A list of logical and/or physical filenames used as input for the job and stored on an SE
- **ReplicaCatalogue** – If InputData has been specified with one or more LFNs this is mandatory
- **DataAccessProtocol** – Specifies the protocol used to retrieve input data from SE (also mandatory id InputData specified)
- **Requirements** – A Boolean ClassAd expression which all CEs must satisfy to be considered
- **Rank** – A floating point ClassAd expression that can be used to rank all CEs passing the Requirements (default is –other.EstimatedTraversalTime)

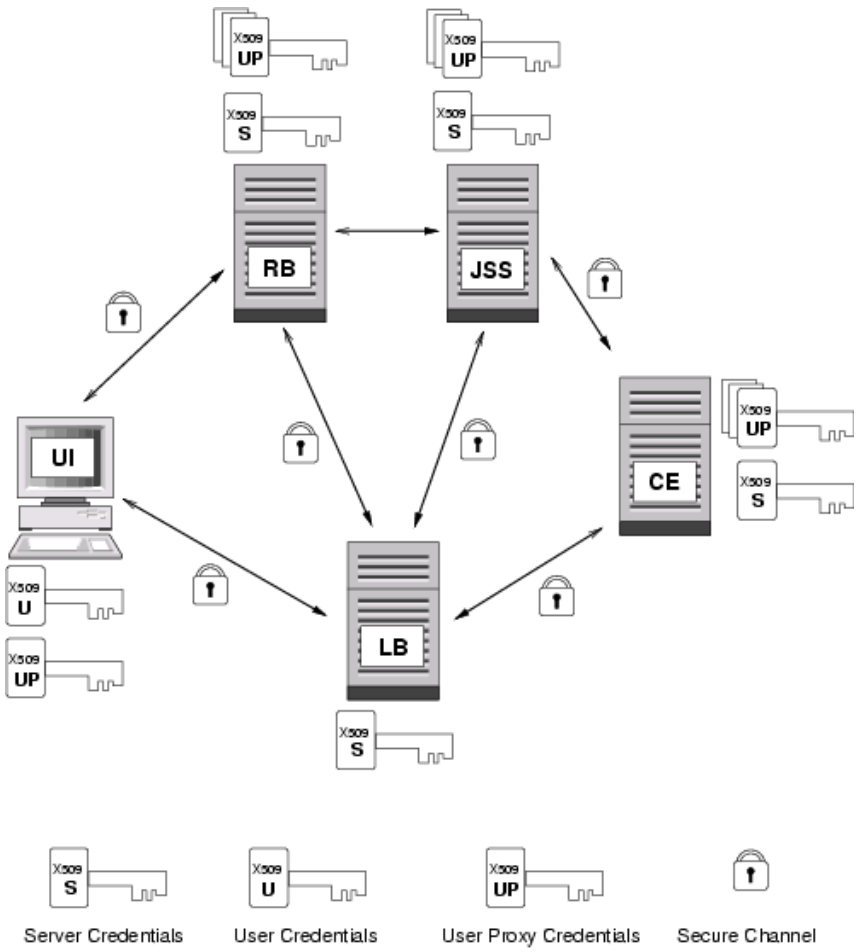
UI provided JDL attributes

- **dg_jobId** – Generated by the UI
- **CertificateSubject** – comes from X509_USER_CERT environment variable and used to evaluate user restrictions to CEs
- **UserContact** – A valid email address (if the –notify option has been chosen)
- **SubmitTo** – A specific resource to submit the job to (e.g. that obtained from a dg-job-list-match)

Resource (IS provided) JDL attributes

- For a full list and description see the datagrid "JDL Attributes" note.
- CE
  - LRMSType, LRMSVersion, QueueName
  - GlobusResourceContactString, GRAMVersion
  - Architecture, OpSys
  - MinPhysicalMemory, MinLocalDiskSpace, TotalCPUs, FreeCPUs, NumSMPs, MinSPUProcessors, MaxSPUProcessors
  - TotalJobs, RunningJobs, IdleJobs, MaxTotalJobs, MaxRunningJobs
  - WorstTraversalTime, EstimatedTraversalTime
  - Priority, MaxCPUTime, MinSI00
  - AFSAvailable…
- SE
  - SEId
  - SEProtocol
  - CloseCE

Security



- User Interface, Resource Broker, Job Submission Service, and Computing Element all need a delegated user credential allowing them to act on behalf of the user
- Credentials limited in time to prevent security problems
- Introduces problems of how to renew an about to expire delegated user credential
  - How this will be done is not yet clear

Laptop Installation

- Simply download an install the UI package list from http://marianne.in2p3.fr/datagrid/testbed1/repositories/pkg-repository.html
- LCFG configuration will also be possible in future
- Only a user certificate is necessary for a UI machine (i.e. no host certificate)
- UI configuration contained in UI_ConfigEnv.cfg (in /opt/edg/etc on pc18) which contains the following info:
  - address and port of accessible RBs
  - address and port of accessible LBs
  - default location of the local storage areas for the Input/Output sandbox files
  - default values for the JDL mandatory attributes
  - default number of retrials on fatal errors when connecting to the LB.

Try it yourself

- v1.1.0 of EDG UI tools installed on PC18
  - GridPP workload management pages https://www.gridpp.ac.uk/workload/ describe the installation
- Are you in the GridPP Virtual Organisation?
  - ldapsearch -x -h vo.gridpp.ac.uk –b 'ou=testbed,dc=gridpp,dc=ac,dc=uk' '(objectClass=*)'
  - If not go to https://www.gridpp.ac.uk/vo/
- A sample JDL file test.jdl and executable test.exe are in pc18:~bjw/
  - test.exe takes one argument, the number of loops to run the executable (~1s per loop on pc18)
- Try the commands from the UI section…

Summary

- Websites:
  - WP1- http://server11.infn.it/workload-grid/
  - GridPP Workload Management Work Group - http://www.gridpp.ac.uk/workload/
- Reference documents (on WP1 site):
  - Overview – DataGrid-01-D1.2-0112-0-3
  - RB – "Resource Broker Architecture and APIs" S. Cavalieri & S. Monforte
  - L&B – DataGrid-01-TEN-0109-1_0
  - JSS – DataGrid-01-TEN-0108-0_0
  - UI – DataGrid-01-TEN-0103-0_0
  - JDL HowTo – DataGrid-01-TEN-0102-0_2
  - JDL attrib. – DataGrid-01-NOT-0101-0_6