



DataGrid

WP1 PM9 RELEASE



Document identifier: **DataGrid-01-TYP-0103-0_0**

Date: **26/05/2001**

Work package: **WP01**

Partner: **INFN**

Document status **DRAFT**

Deliverable identifier:

Abstract:

This document provides a brief overview of the functionalities foreseen for the first workload management system prototype (PM9 deliverable of WP1, Grid Workload Management)

Delivery Slip

	Name	Partner	Date	Signature
From	Massimo Sgaravatto	INFN Padova		
Verified by				
Approved by				

Document Log

Issue	Date	Comment	Author
0_0	26/05/2001	First draft	Massimo Sgaravatto

Document Change Record

Issue	Item	Reason for Change

Files

Software Products	User files
Microsoft Word 2000	WP1-PM9.doc
Adobe Acrobat 4.0	WP1-PM9.pdf



CONTENT

1. INTRODUCTION	4
1.1. APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS	4
1.2. TERMINOLOGY	4
2. WP1 PM9 RELEASE.....	6
2.1. JOB SUBMISSION	6
2.2. ACCESS TO STORAGE ELEMENTS.....	8
2.3. TRANSFERRING THE “APPLICATION SANDBOX”	8
2.4. DATA, REPLICATED CATALOG, STORAGE ELEMENT	9
2.5. STORAGE ELEMENT AND INFORMATION SERVICE.....	9
2.6. LOGGING AND BOOKKEEPING	10

1. INTRODUCTION

This document provides a brief overview of the functionalities foreseen for the first workload management system prototype (PM9 deliverable of WP1, Grid Workload Management).

Besides describing the functionalities of the system, the purpose of this document is to identify and make clear the external dependencies (i.e. functionalities provided by the other WP's).

1.1. APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

Applicable documents

Reference documents

[R1] Job Submission User Interface Man Pages

(http://www.infn.it/workload-grid/docs/DataGrid-01-TEN-0101-0_1-Document.pdf)

[R2]: Job Description Language HowTo

(http://www.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_0-Document.pdf)

1.2. TERMINOLOGY

Definitions

Glossary

CE: Computing Element

GIS: Grid Information Service (aka MDS)

JDL: Job Description Language

LRMS: Local Resource Management System

MDS: Metacomputing Directory Service (aka GIS)

RB: Resource Broker

RC: ReplicaCatalog

SE: Storage Element

UI: User Interface

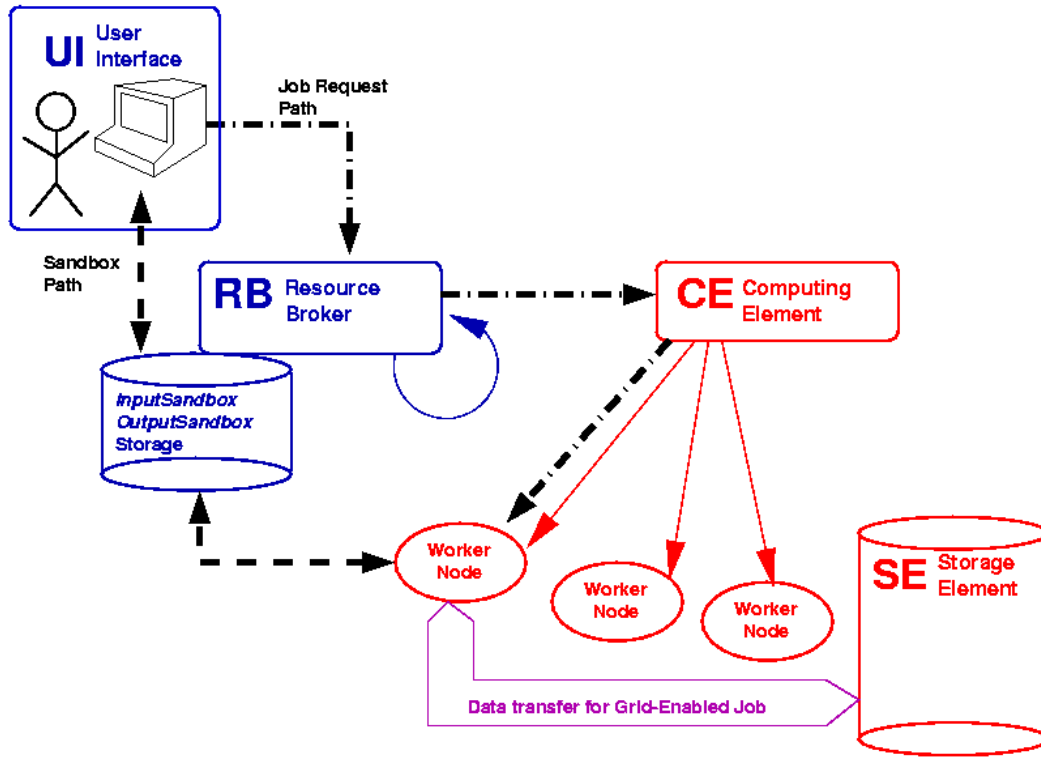


Figure 1. Data transfer paths for the WP1 PM9 prototype

2. WP1 PM9 RELEASE

For the PM9 release, users will be able to “interact” with the workload management system via a command-line user interface (UI), described in [R1].

In particular, after having invoked the user interface, it will be possible to:

- Submit a job for execution on a Computing Element (CE) (command *dg-job-submit*)
- Find the list of resources suitable to run a specific job (command *dg-list-job-match*)
- Cancel one or more submitted jobs (command *dg-job-cancel*)
- Retrieve the output files of a completed job (command *dg-get-job-output*)
- Display bookkeeping information about submitted jobs (command *dg-job-status*)
- Display logging information about jobs (command *dg-get-logging information*)

2.1. JOB SUBMISSION

The command *dg-job-submit* (see [R1]) is used to submit a job to a Computing Element (CE).

As CE, for the PM9 release, we consider a Globus resource represented by:

- A queue of an underlying Local Resource Management System (LRMS), such as LSF, PBS, etc..., assuming that this queue represents a set of “homogeneous” resources (that is when a job is submitted to a specific queue, it doesn’t matter in which node of this queue the job is dispatched)
- A “single” node, that doesn’t rely on a LRMS (using the fork system call)

The command *dg-job-submit* requires as input a file specified using a Job Description Language (JDL) [R2], based on Condor Class-Ads. The JDL expression specified by the user in this input file is used to express the job characteristics, and the (required and preferred) resources for this job.

In particular in this JDL expression the user will have to specify:

- The executable
- The standard input/output/error files
- The input data (attribute *InputData*) and the identifier of the RC (attribute *ReplicaCatalog*) where these data are “published”: see section 2.4
- A list of input files (attribute *InputSandbox*) that must be staged from the UI machine to the local disk of the executing machine. These are “small” files (i.e. the executable, the shell script that invokes this executable, the standard input, etc...) necessary for the execution, and not managed by the grid services (e.g. they are not stored in any SE, they are not “published” in any RC, ...): see section 2.3. The size of the *InputSandbox* may be limited by the staging disk resources available on the RB.
- A list of output files (attribute *OutputSandbox*) that must be retrieved when the job completes its execution. These are “small” files (i.e. the standard output and error files, etc...) and do not represent the data files that will be stored in one SE, and “published” in one RC (see section

2.3). The size of the *OutputSandbox* may be limited by the staging disk resources available on the RB

- The job requirements on resources (attribute *Requirements*)
- The job preferences on resources, for those resources that have already met the *Requirements* expression (attribute *Rank*)

The JDL expression is then “passed” from the UI to a Resource Broker (RB), the component of the workload management system responsible to choose the “right” resource where to submit the job.

Our implementation will work at the personal (a RB for each submitting machine) and community level (a RB used by many users, for example users belonging to a same group/experiment/virtual organization): however we recommend the community level.

The RB to use is specified when the UI is invoked.

Therefore the UI and the RB in general will run on different hosts.

The RB will find the “right” resource considering:

- Where the input data (represented by the attribute *InputData*) are stored.
For the PM9 release, WP1 is not going to trigger any data transfer from one SE to an other one, and therefore the job will be submitted to a CE “close enough” to a SE where the required input data are stored. The notion of “close enough” will be represented in the MDS by WP5 and/or WP4 (see Section 2.5).
Moreover, user can specify in the JDL expression (as requirement or rank) also a SE where the output data produced by his/her job should be stored: in this case the job will be submitted to a CE “close enough” to this SE. Again, WP1 will rely on the MDS information to find the appropriate CE.
- The authorization policies
A job will be submitted only on a CE “accessible” by the user.
As already reported, as authorization policies we will consider the Globus *grid-mapfile*, which therefore must be published in the MDS: if the *grid-mapfile* of a resource is not published in the MDS, this resource will not be considered by the RB, and therefore will not be used to run jobs.
- The (required and preferred) resources for this job (attributes *Requirements* and *Rank* in the JDL expression) “matched” with the characteristics and status of the available Grid resources (information available in the MDS).
The list of attributes (referring to the characteristics and status of computing resources) that can be used in the *Requirements* and *Rank* expressions can be found in the Annexes of [R1]: these attributes have been agreed with WP4, responsible to implement the required information providers (the elements responsible to provide the MDS with the required information about the computing resources).

We would like to know from WP5 which attributes about SE's will be published in the MDS for the PM9 release, in order to understand which other attributes can be specified in the *Requirements* and *Rank* expressions (see section 2.5).

The job is then submitted by a Job Submission Service (JSS), a wrapper of Condor-G, to the Globus resource chosen by the RB.

Note that a user can also decide to submit a job to a specific resource, without using the broker functionalities provided by the RB: the *dg-list-job-match* can be used as first step to find the suitable resources for the job (in this case the RB just find these suitable resources, without performing any job submission), and then the user can choose which resource to use.

2.2. ACCESS TO STORAGE ELEMENTS

Since it has been decided that a job running on a CE can't access a "attached" SE using a local protocol (NFS, AFS, ...), but instead a grid protocol must be considered, this means that if the job needs to access the input data stored in one SE and/or write output data to a SE, this job must be "grid-enabled" (see figure 1): WP1 is not going to provide any mechanisms to stage data from a SE to the CE's local disk and vice versa .

Instead, WP1 is going to provide mechanisms for transferring the "Application Sandbox" (files that need to go from the end user submission machine to the farm worker nodes and vice versa): see section 2.3.

Moreover, WP1 assumes that it is up to the grid-enabled job to "publish" the new output data in the RC, using the appropriate WP2 services.

2.3. TRANSFERRING THE "APPLICATION SANDBOX"

As described in section 2.1, a user can specify in the JDL expression (using the attribute *InputSandbox*) a list of input files required for the execution of the job (i.e. the executable, a script that invokes the executable, the standard input file, etc...). As represented in figure 1, these files will be transferred at job submission time to the RB that will store them temporarily on its local disk, and then they will be staged in the local disk of the executing node (probably using gridftp).

At submission time the user can also specify a list of files for the *OutputSandbox* attribute: these files represent the output files that user wants back in its machine after job completion. When the job completes its execution, these files are transferred (probably using gridftp) to the RB's local disk, and they can then be retrieved by the user with the *dg-get-job-output* command (see [R1]).

To allow these data transfers, outbound connectivity for the executing machines must be allowed.

2.4. DATA, REPLICA CATALOG, STORAGE ELEMENT

As reported in section 2.1, in the JDL expression the input data required for the job (data stored in one or more SE, and “published” in one RC) must be specified, using the attribute *InputData* (the “identifier” of this RC must be specified as well, using the attribute *ReplicaCatalog*, since we can not assume that a single RC will be used for the whole Grid).

After some discussions with WP2, it seems that we could/should rely on the Globus Replica Catalogue for the RC, for the PM9 release.

As far as we know, the Globus Replica Catalogue foresees logical collections and logical files, and physical locations where these are stored.

We would like to know from the applications what do they plan to specify as attribute *InputData*:

- A logical collection ?
- A list of logical collections ?
- A list of logical files ?
- ... ?

The RB will query the *ReplicaCatalog* to find the SE(s) where these input data are available.

We would like to know from WP5 and WP2:

- How are SE’s going to be identified at PM9 ?
- How it is possible to find out the SE(s) where a particular logical collection/logical file is stored, querying the RC ?

We are assuming that, considering the Globus Replica Catalogue model, the physical location where a logical collection/logical file is stored “represents” the SE, and the *hostname* field of this physical location can be used to uniquely identify this SE.

We would like to make sure that this matches with WP2-WP5 plans.

2.5. STORAGE ELEMENT AND INFORMATION SERVICE

We would like to know from WP5 how are SE’s going to be represented and described in the MDS at PM9.

As described above, first of all we would like to know are they are going to be identified.

Using the *hostname* attribute ?

As described in section 2.1, WP1 has to be able to match a CE with SE’s that are “close enough” (e.g. on the same LAN), since no data transfers will be triggered by WP1 for the PM9 release.

We agreed that WP4 will publish the CE → SE association in the MDS.

Having also the reverse mapping (SE → CE) could provide a very powerful resource selection tool for PM9. Is WP5 going to publish this information in the MDS ?

Which other information about Storage Elements is WP5 planning to publish in the MDS ?

Note that, as already reported, WP1 is planning to rely on the Globus MDS-2 as backbone of the PM9 information infrastructure, since we think that we can't rely on the current implementation of the Globus GIS (MDS), not even for a prototype system.

2.6. LOGGING AND BOOKKEEPING

The command *dg-job-status* (see [R1]) is used to display bookkeeping information about submitted jobs (job status, the resource where the job runs, the submission/scheduled/start/stop time, etc...) while the command *dg-get-logging-information* (see [R1]) is used to get logging information (the complete JDL expression plus the job status, the resource where the job runs, the submission/scheduled/start/stop time).

The difference is that bookkeeping information represents short term (volatile) data about currently active jobs, while logging information are long term (persistent) data about jobs and the workload management system, used for debugging and post-mortem analysis of job runs.