Nordic Testbed for Wide Area Computing and Data Handling

1/28/02

Comments on the Computing Element (CE) Information Provider of the European DataGrid (EDG) Testbed 1

Balázs Kónya

Abstract

NorduGrid carried out an investigation on the Testbed 1 Computing Element MDS entry, we have studied how a CE is represented in the Testbed 1 information system. This report presents our experience with the Computing Element Information Provider of the EDG Testbed 1 release. In particular, we investigated how accurately the status of a computing cluster, governed by the PBS local resource management system, is represented in the Grid Information System of EDG Testbed 1.

1. Introduction

According to Testbed 1 definition, a Computing Element (CE) is nothing more than a queue of the underlying Local Resource Management System (LRMS). The Information Provider is program which serves as an interface between the Globus Information system and the LRMS. The Information Provider acquires cluster status information from the LRMS and propagates this information further to the Globus.

The EDG Testbed 1 contribution to the Grid Information System (in case of our system describing a PBS-queue type CE) consists of the following parts:
-a schema definition (with the ComputingElement and the CloseStorageElement objectclasses)
-an info-provider perl script: ce-pbs
-a couple of .ldif templates to store static information
-some overcomplicated configuration procedure in order to the integrate the provider to the Globus MDS

The CE Information Provider is a real team work of the EDG, the LDAP schema was defined by the WP1 and the info-provider script was written by WP4. It is a bit confusing and this was the reason why I was not able to find the CeinformationProviders.rpm under WP3 (grid information system/monitoring) on the datagrid.in2p3.fr repository, from where you get the information system related rpms (edg-info-main.rpm and edg-info-se.rpm , edg-info-netmon.rpm). The CeinformationProvider.rpm is placed under the WP4/resource_mgt.

2. Integration into the Globus MDS

The ce-pbs info-provider script (which creates two MDS entries) can be easily integrated in a "standard Globus way" into the MDS tree. A reasonable choice is to link the entries directly under the *Mds-host-hn=machine.name, mds-vo-name=local, o=grid* branch in the default Globus GRIS tree. In order to add the provider to Globus2 MDS you only need to list the EDG schemas in the Globus LDAP configuration file (grid-info-slapd.conf), install the CeInformationProviders.rpm (which contains the ce-pbs script) together with the edg-info-main.rpm (this has the schema files) and add an entry like this to the grid-info-resource-ldif.conf for each CE:

```
#generate CE info for the PBS queue PC
dn: ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc, Mds-Host-hn=grid.
quark.lu.se, Mds-Vo-name=local, o=Grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
```

```
type: exec
path: /opt/edg/info/mds/bin
base: ce-pbs
args: -globus-path /opt/globus -static /opt/edg/info/mds/etc/ldif/ce-
static-pc.ldif -globus-config-file /opt/globus/etc/globus-job-manager.
conf -auth-users-from-grid-mapfile /etc/grid-security/grid-mapfile -queue
pc -dn ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc,Mds-Host-hn=grid.
quark.lu.se,Mds-Vo-name=local,o=Grid -cluster-batch-system-bin-path
/usr/local/bin -closeses /opt/edg/info/mds/etc/ldif/closese-pc.ldif
updateMethod: continuous
updatePeriod: 30
cachetime: 0
timelimit: 20
sizelimit: 20
```

We believe that the integration of the information providers into the Globus MDS is an integral part of the EDG Globus configuration and thus should be done by the globus-mds-edgconfig.rpm package.

3. The Test cluster

The CEInformationProvider has been set up on top of the Globus-enabled PBS clusters in Lund and Oslo (you can check our installation via the NorduGrid MDS explorer (http://grid.quark.lu.se/NorduGridMDS -> grid.quark.lu.se or grid.uio.no with mds-vo-name=local,o=grid).

The Lund cluster consists of a frontend machine (grid.quark.lu.se) and three computing nodes the detailed configuration is given at http://www.nordugrid.org/specs/lund.html.

The cluster runs the PBS local resource manager system configured with two queues pc and pclong managing the 3 computing nodes (altogether 5 processors in the 2+2+1 distribution). The queues have been configured as follows: The queue pc has max_userrun 3, max_running 5, no queue limit, default.cput 2h, max.cput 2h. The queue pclong has max_userrun 2, max_runnin 2, no queue limit, default.cput 6h, no max.cput and it has a default.need.nodes=single setting, which means that this queue is effectively assigned to the single-processor-node. The nodes are defined as cluster nodes, they are job-exclusive nodes.

The implemented PBS setup is a rather typical, but not a trivial one, it is still very simple, the PBS configuration of real production systems has much more queue parameters and limits set.

The above PBS configuration implies that according to the EDG terminology there are two Computing Elements on the grid.quark.lu.se cluster:

ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc
ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc-long

4. CE-PBS script

The information provider ce-pbs script populates the computing element entries of the MDS. The script reads static data from the files ce-static-queuename.ldif and closese-static-queuename.ldif (these files are edited by sysadmins according to the local setup) and acquires information about the status of the PBS by using the qmgr PBS command. We would suggest to replace the qmgr with other PBS status query commands like qstat and pbsnodes. The qmgr (according to the man) is not for this purpose: "The qmgr command provides an administrator interface to the batch system", clearly qmgr is for PBS administration, and not for system status queries. We note here that the Globus GRAM itself interfaces to the PBS by utilising the qstat in the globus-script-pbs-queue script.

5. Detailed Analysis

In our test scenario we submitted 7 jobs through Globus to the queue pc of the grid. quark.lu.se:/jobmanager-pbs (i.e. to the CE ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc):

```
$globus-job-submit grid.quark.lu.se:/jobmanager-pbs -q pc
/bin/sleep 600
```

The PBS status was queried with the qstat and pbsnodes commands:

```
$ qstat
Job id      Name       User       Time Use S Queue
--------------- --------------- --------------- -------- - -----
690.grid     STDIN      gridtest         0 R pc
691.grid     STDIN      gridtest         0 R pc
692.grid     STDIN      gridtest         0 R pc
693.grid     STDIN      gridtest         0 Q pc
694.grid     STDIN      gridtest         0 Q pc
695.grid     STDIN      gridtest         0 Q pc
696.grid     STDIN      gridtest         0 Q pc
```

The output of qstat shows that there are 7 jobs in the queue pc, 3 are running and 4 are queued (because of the max_userrun 3 of the queue pc)

```
$pbsnodes -a
node1.qfarm.lu.se
   state = job-exclusive
   np = 1
   properties = PIII,single
   ntype = cluster
   jobs = 0/692.grid.quark.lu.se
```

```
node2.qfarm.lu.se
   state = free
   np = 2
   properties = PIII,dual
   ntype = cluster
   jobs = 0/690.grid.quark.lu.se

node3.qfarm.lu.se
   state = free
   np = 2
   properties = PIII,dual
   ntype = cluster
   jobs = 0/691.grid.quark.lu.se
```

From the pbsnodes we know that the three running jobs are placed on the node1 node2, node3 respectively, and there are two free processors left in the cluster (one on node2 and node3).

Our next step was to query the MDS and compare its CE entry to the direct PBS info obtained from the qstat, pbsnodes local commands.

Please find below the commented output of the MDS query with <comments like this below the actual attribute and "*quotations like this from the EDG schema description document, WP1 inputs to the datagrid grid information service schema specification, DataGrid-01-TEN-0104-0_6* " >.

```
$ldapsearch -h grid.quark.lu.se -p 2135 -x -b "Mds-vo-
name=local,o=grid"

# grid.quark.lu.se:2119/jobmanager-pbs-pc, grid.quark.lu.
se, local, Grid
dn: ceId=grid.quark.lu.se:2119/jobmanager-pbs-pc,Mds-Host-
hn=grid.quark.lu.se,
 Mds-Vo-name=local,o=Grid
```

< the CEentry is inserted to the right place in the DIT, under the Mds-Host-hn= >

```
objectClass: DataGridTop
objectClass: ComputingElement
CEId: grid.quark.lu.se:2119/jobmanager-pbs-pc
```

< this one shows that a CE is just a PBS queue. Some of the real PBS systems have a so-called routing queue, which is not a real one, used only for rerouting jobs, moreover this routing queue is the default and only available queue for job submission. Furthermore there are production systems with queues not open for user submissions, reserved just for pbs admins to manage unusual jobs.
The above two example suggests that the CE <---> PBS queue mapping can be problematic in some cases. Maybe the CE <---> cluster mapping should be considered in the future. >

```
GlobusResourceContactString: :/jobmanager-pbs:
/O=Grid/O=NorduGrid/CN=grid.quark.lu.se
```

< this is the contact string of the gatekeeper and NOT the CE (queue),
if you want to submit a job to this CE (queue) you need the queue name too.
anyway, the gatekeeper name is already in the MDS (mds-host-hn). >

`GRAMVersion: ?`

< the "?" value is hardcoded in the ce-pbs script :

$gramversion = "?" # undefined with Globus 2?

why is it interesting at all? >

`Architecture: intel`

< static, comes from the .ldif file >

`OpSys: Mandrake 8.0`

< static, comes from the .ldif file >

`MinPhysicalMemory: 256`

<static, comes from the .ldif file, the minimum installed memory of the nodes,
should be dynamic in the future, it is not the installed minimum which really matters
but the available minimum memory on the nodes should be stored.
For example the PBS do have dynamic information available (obtained from the
pbs_mom daemons running on the nodes) about the free/used memory of the nodes.>

`MinLocalDiskSpace: 16`

<static, *"represents the minimum local space (to nodes) in working directory where the job computation will take place, available to a running job running on a worker node"* , this must be dynamic, otherwise really has no sense. Easy to implement (grep free space from the output of df on the working directory and take into account the user quotas) >

`TotalCPUs: 5`

< dynamic from qmgr, could be a wrong output in case virtual cpus are used in PBS config, and in case a queue uses dedicated nodes (as in the case of the cpulong queue, a standard job submitted to the cpulong queue experiences only a single CPU)>

`FreeCPUs: 4`

< dynamic from qmgr, wrong, there are just two free CPUs (compare to the pbsnodes output above) >

`NumSMPs: 2`

< static, the sysadmin specifies it, in the future could be read from the pbs>

`MinSPUProcessors: 2`

< static, the sysadmin sets it >

`MaxSPUProcessors: 2`

< static, the sysadmin sets it >

`TotalJobs: 7`

<dynamic from qmgr, correct value, *"number of jobs submitted to the CE: running + idle"* >

`RunningJobs: 3`

<dynamic, correct value>

`IdleJobs: 4`

<dynamic, correct value, number of queuing jobs>

`MaxTotalJobs: 5`

<dynamic, *"the maximum number of jobs (running and idle) allowed for the CE"*, wrong, the correct value is infinite since there is no queue limit, moreover the actual number of totaljobs (running and idle) in the queue pc is 7 (3 running 4 queueing). >

`MaxRunningJobs: 5`

<dynamic, *"defines the maximum number of running jobs allowed for the CE"*, wrong, 5 is the max_running from this queue but there is a max_userrun (3) -> a grid user will never run more than 3 jobs in this queue! >

`WorstTraversalTime: 36000`
`EstimatedTraversalTime: 50400`

`Active: TRUE`

<dynamic, correct, would be nice to have info about open time windows too, i.e. how long the CE will remain active>

`Priority: 100`

<dynamic, irrelevant, queue priorities in PBS are used by pbs admins to distinguish between their queues (from which queue should the next job be taken), the numbers are arbitrary and completely site specific --> a local pbs queue priority carries no information for the grid. >

```
MaxCPUTime: 7200
```

<dynamic, *"defines the maximum cputime (in seconds) allowed for jobs submitted to the CE"*, correct, the maxcputime is 2 hours = 7200s, what about displaying this value in hours? (long-running jobs measure their time in hours not seconds) >

```
MaxWallClockTime: 0
```

<dynamic, "*is the maximum wall clock time (in seconds) allowed for jobs submitted to the CE*", wrong, this is unset and not zero >

```
AverageSI00: unknown
MinSI00: unknown
MaxSI00: unknown
```

< these SI00 attributes are static, the sysadmin specifies them. "*MaxSI00 is the maximum value of the SpecInt2000 benchmark among the processors associated to this CE*"
Benchmarking is not an easy stuff, the SpecInt2000 is NOT a single cpu benchmark (as it is explicitly stated on the http://www.specbench.org/osg/cpu2000:
These benchmarks measure the performance of the processor, memory and compiler on the tested system.) Therefore if we want to use the specs, the sites should run these benchmarks (because of the difference in memory, compiler, bus system). However the spec is a commercial software . So how should I set these attributes on my system suppose I have a PIII/1Ghz processor with 512MB ram? Is there an "EDG -SI00 table" which relates cpus to spec numbers? Another question is, which specint2000 value should be used, the base or the peak? To close this remark I copy here a few specint2000 value for 1GHz intel processors from the official site: 423,457,410,448. This is the reason I chose the "unknown" value. >

```
AuthorizedUser: /O=Grid/O=NorduGrid/OU=tsl.uu.
se/CN=Mattias Ellert
AuthorizedUser: /O=Grid/O=NorduGrid/OU=uio.no/CN=Borge
Kile Gjelsten
AuthorizedUser: /O=Grid/O=NorduGrid/OU=uio.no/CN=Aleksandr
Konstantinov
AuthorizedUser: /O=Grid/O=NorduGrid/OU=quark.lu.
se/CN=Oxana Smirnova
AuthorizedUser: /O=Grid/O=NorduGrid/OU=uio.no/CN=Farid
Ould-Saada
AuthorizedUser: /O=Grid/O=NorduGrid/OU=quark.lu.
se/CN=Balazs Konya
AuthorizedUser: /O=Grid/O=NorduGrid/OU=nbi.dk/CN=Anders
Waananen
```

<dynamic, the entire content of the grid-mapfile is parsed here.
Could be wrong, what if a queue has user limitations and certain local unix users are

excluded from the queue? User restrictions of the queueing system should be taken into account in the future.>

```
RunTimeEnvironment: none
```

<static, taken from a textfile, you have to use the right keyword. i.e. Atlas-1.1 is not the same as Atlas_1.1 >

```
AFSAvailable: FALSE
OutboundIP: TRUE
InboundIP: FALSE
```

< these are static>

```
QueueName: pc
LRMSType: PBS
LRMSVersion: OpenPBS_2.3
```

<dynamic from qmgr>

```
# none, grid.quark.lu.se:2119/jobmanager-pbs-pc, grid.quark.
lu.se, local, Grid
dn: closeSE=none,ceId=grid.quark.lu.se:2119/jobmanager-pbs-
pc,
Mds-Host-hn=grid.quark.lu.se,Mds-Vo-name=local,o=Grid
objectClass: CloseStorageElement
objectClass: DataGridTop
objectClass: DynamicObject
CEId: grid.uio.no:2119/jobmanager-pbs-pc
CloseSE: none
MountPoint: none
```

<the closeSE is a subentry to the CEinfo, everything is static from the close-se.ldif (at least in my config) >

6. Conclusion

The information content of the CEentry is mostly static, or not in agreement with the actual state of the cluster, it contains several incorrect, misleading values. The ce-pbs provider should be considerably rewritten and the Computing Element schema could be reconsidered in order to provide a useful image of a Computing Element.

In order to be able to write a better provider we definitely need to know what kind of information is used by the broker from the CEentry. We would like to see the complete list of MDS attributes being used by the broker in its decision making algorithm.