

# Providing the Grid Information Service with information of local farms

Massimo Biasotto – INFN LNL  
Massimo Sgaravatto – INFN Padova

Draft release 1.0.3  
November 17, 2000

## 1 Introduction

The goal of this document is to review the information related to the characteristics and status of a farm (a cluster of machines managed by a local resource management system), and the information concerning jobs submitted via Globus, that can/must be published in the GIS (Grid Information Service).

After an overview of the information that are published by default in the GIS, we propose a first, very preliminary modification of the default schema, taking off the default attributes that in our opinion are not useful at all for our needs, and considering some other information provided by the underlying resource management system.

In this document only LSF and Condor have been considered as underlying resource management systems for Globus: therefore to complete this analysis it will be necessary to consider the other resource management systems (i.e. PBS) that will be used in a Grid environment.

## 2 Information on characteristics and status of a farm

### 2.1 Information of a farm published by default in the GIS

When LSF or Condor is configured as job manager for a Globus machine, by default during the Globus deployment process the file:

*<globus-deploy-dir>/etc/grid-info-resource.conf*

is modified in order to publish in the GIS some information, according to a specific schema (reported in appendix A).

The following picture represents an example, where LSF has been considered as job manager for Globus (but for Condor there are no significant differences):

<b>dn</b>	<i>service=jobmanager-lsf, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid</i>
objectclass	GlobusTop
objectclass	GlobusDaemon
objectclass	GlobusService
objectclass	GlobusServiceJobManager
objectname	<a href="#">service=jobmanager-lsf, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid</a>
service	jobmanager-lsf
hn	lxde02.pd.infn.it
contact	lxde02.pd.infn.it:2119/jobmanager-lsf:/C=IT/O=INFN/OU=gatekeeper/L=PD/CN=lxde02.pd.infn.it/Email=root@lxde02.p
schedulerversion	0.1
schedulertype	lsf
deploydir	/etc/globus
cputype	i686
osversion	2.2.12-20
osname	linux
manufacturer	unknown
machine_type	unknown
gramversion	1.67
gramversiondate	2000/04/17 10:00:45
gramsecurity	ssleay
lastupdate	Thu Nov 09 15:35:49 GMT 2000
ttl	00:30:00

```

• o=Grid
  • dc=it
    • dc=infn
      • dc=pd
        • hn=lxde02.pd.infn.it
          • > service=jobmanager-lsf
            • queue=chisnt_reun_qu
            • queue=idle
            • queue=license
            • queue=mqueue
            • queue=night
            • queue=normal
            • queue=owners
            • queue=priority
            • queue=short

```

- *service* is the name of the job manager;
- *hn* represents the host name of the front-end machine;
- *contact* is the string identifying the contact information;
- *schedulerversion* is the number representing the scheduler version (note that this is not the version of LSF/Condor/...);
- *schedulertype* defines the type of scheduler (LSF/Condor/ ...);
- *deploydir* is the Globus deploy directory;
- *cputype* identifies the CPU information of the front-end machine;
- *osversion* represents the version of the operating system running in the front-end machine;
- *osname* identifies the operating system of the front-end machine;
- *manufacturer* should represent the manufacturer of the front-end machine;
- *machine\_type* should identify the machine type;
- *gramversion* represents the GRAM version;
- *gramversiondate* is the GRAM version date;
- *gramsecurity* identifies the GRAM security model information.

“Under” the *GlobusServiceJobManager* objectclass, some *GlobusQueue* objectclasses are defined.

For LSF these objectclasses represent the LSF queues (therefore there is one of these *GlobusQueue* objectclasses for each LSF queue configured for the cluster), as shown in the following example:

<b>dn</b>	<i>queue=mqueue, service=jobmanager-lsf, hn=lxde15.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid</i>
objectclass	GlobusTop
objectclass	GlobusQueue
queue	mqueue
maxtime	0
maxcputime	0
maxcount	0
maxrunningjobs	3
maxjobsinqueue	0
maxtotalmemory	0
maxsinglememory	0
totalnodes	5
freenodes	1
whenactive	0
status	Open:Active
dispatchtype	batch
priority	30
alloweduserlist	NULL
jobwait	NULL
schedulerspecific	NULL
ttl	00:01:00
lastupdate	Thu Aug 31 10:14:22 GMT 2000

For Condor, by default a single *GlobusQueue* objectclass is created: this queue represents all the machines of the Condor pool of the same architecture and of the same operating system of the front-end machine. The following picture is an example representing a *GlobusQueue* objectclass for Condor:

<b>dn</b>	<i>queue=intel-linux, service=jobmanager-condor, hn=lxde14.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid</i>
objectclass	GlobusTop
objectclass	GlobusQueue
queue	intel-linux
maxtime	0
maxcputime	0
maxcount	0
maxrunningjobs	0
maxjobsinqueue	0
maxtotalmemory	0
maxsinglememory	0
totalnodes	3
freenodes	1
whenactive	0
status	0
dispatchtype	batch
priority	NULL
alloweduserlist	NULL
jobwait	NULL
schedulerspecific	NULL
ttd	00:01:00
lastupdate	Thu Aug 31 10:22:55 GMT 2000

The meaning of the various attributes is the following:

- *queue*: the name of the queue.  
For LSF this is the name of the LSF queue.  
For Condor this name is “built” considering the name of the architecture and of the operating system of the front-end machine (i.e. *intel-linux*);
- *maxtime*: probably this parameter should represent the maximum allowed wall clock time for the jobs submitted to this queue, but it is always equal to 0, both for Condor and for LSF;
- *maxcputime*: probably this parameter should represent the maximum CPU time for the jobs submitted to this queue, but it is always equal to 0, both for Condor and for LSF;
- *maxcount*: the meaning of this parameter is not clear; in any case it is always equal to 0, both for LSF and Condor;
- *maxrunningjobs*: for LSF this parameter represents the number of running jobs submitted to this queue, while for Condor this value is always equal to 0;
- *maxjobsinqueue*: for LSF this parameter represents, if defined, the limit on the number of jobs dispatched from this queue at one time (otherwise the parameter is equal to 0), while for Condor this value is always equal to 0;
- *maxtotalmemory*: the meaning of this parameter is not clear; in any case it is always equal to 0, both for LSF and Condor;
- *maxsinglememory*: this parameter is always equal to 0, both for LSF and Condor (it is not clear what this parameter should represent);
- *totalnodes*: this parameter should represent the total number of processors associated to this queue.

For Condor this parameter is found considering the total number of hosts of the Condor pool of the same architecture and of the same operating systems of the front-end machine: we think this is the right way to calculate this parameter.

For LSF this parameter is calculated considering all the hosts known to LSF and configured as LSF servers (that is machines able to run batch jobs), and for these hosts the number of CPUs is gathered. We think that only the hosts to which this queue can dispatch jobs should be considered;

- *freenodes*: this parameter should represent the total number of processors associated to this queue, able to run, in that moment, jobs submitted to that queue.

For Condor this parameter is found considering the “unclaimed” nodes of the Condor pool of the same architecture and of the same operating systems of the front-end machine, that is the number of nodes able to run jobs, according to the policies defined by the administrators of these Condor resources. We think this is fine: this is the right way to find the value for this parameter.

For LSF this parameter is found calculating first the number of “busy” nodes (found considering the CPU load averaged over the last minute). The number of free nodes is then calculated as difference between the number of total nodes and the number of busy nodes. We think this is not the right way to calculate this parameter, since it should be equal to the number of nodes able to run, in that moment, jobs submitted to that queue, according to the policies defined for that queue, and according to the characteristics and status of the various hosts in that moment;

- *whenactive*: probably this parameter should report when the queue is active (that is when the jobs submitted to this queue can be dispatched to the executing hosts), but the value for this parameter is always equal to 0, both for LSF and Condor;
- *status*: for LSF this parameter reports the status of the queue: open or close (that is able or unable to accept new jobs) and active or inactive (that is able or unable to dispatch jobs to the executing machines).

For Condor this parameter is always equal to 0;

- *dispatchtype*: this parameter is always equal to “batch”, both for LSF and Condor;
- *priority*: for LSF this parameter represents the priority of the queue, while for Condor it is always equal to 0;
- *alloweduserlist*: this parameter is always equal to NULL, both for Condor and LSF;
- *jobwait*: probably this parameter should report the number of “idle” jobs submitted to this queue, waiting for available resources, but it is always equal to NULL, both for Condor and LSF;
- *schedulerspecific*: this parameter is always equal to NULL, both for Condor and LSF.

## 2.2 Other information that could be provided by the underlying resource management system

This section describes which other useful information, provided by commands or tools of the underlying resource management systems, describing the characteristics and the status of a Condor pool/LSF cluster, could be published in the Grid Information Service.

### LSF

The command *bqueues* can be used to display information about LSF queues.

Many information for each queue are displayed, such as:

- the priority of the queue (already published by default in the GIS);
- the status of the queue (already published by default in the GIS);
- the limit on the number of jobs that can be dispatched from this queue at one time (already published by default in the GIS), specifying this value also for each user, for each processor, for each host;
- the number of jobs in the queue (already published by default in the GIS), specifying also the number of running jobs, pending jobs, suspended jobs (by the system or by the users);
- the maximum CPU time and wall clock time a process can use;
- the maximum file size a process can create, the maximum size of the data segment and of the stack segment of a process, the maximum size of a core file, the maximum running set size of a process, the swap space limit that a job may use;
- the scheduling parameters, that determine when a pending job or a suspended job in the queue can be dispatched to a processor, and when to suspend a running job in the queue;
- the scheduling policies (fairshare, preemptive, ...);
- the windows that define when the jobs submitted to this queue are dispatched to the processors for their execution;
- the list of users (or groups of users) allowed to submit jobs to this queue;
- the list of hosts to which this queue can dispatch jobs.

The LSF commands *bhosts*, *lshosts*, and *lsload* can then be used to view host information, such as:

- the host CPU architecture;
- the number of processors on the host;
- the host CPU factor, that defines the performance of this CPU (this value is not calculated using a benchmark, but it is defined by hand by the LSF administrator);
- the total and the available amount of physical memory;
- the total and the available swap space;
- the maximum number of jobs allowed on the host, specifying also the job limit per user;
- the number of jobs dispatched to the host, specifying the number of running jobs, of suspended jobs, of pending jobs;
- the time windows during which jobs can be started on the host;
- the scheduling and suspending thresholds for the host ;
- the number of disk drives directly attached to the host (but apparently the command doesn't work properly for this parameter, since this value is always equal to 1);
- the total and available size of the disk partition that contains the /tmp directory;
- the load status, specifying the CPU load averaged over different time interval, the percentage of time the CPU is in use, the paging rate, the number of login session, the idle time (the period of no activity by users).

Besides these “default” parameters, in LSF it is possible to define other new attributes, defining static or dynamic properties.

These “custom” parameters are then managed like the other “default” parameters.

Note that with the commands *bhosts*, *lshosts*, and *lsload*, used to find information related with the resources, it is not possible to display “global” or “aggregate” information for all the hosts, or for a set of hosts of a LSF cluster that meet a certain criterion. However this “aggregate” information could be easily found considering the information displayed by the various hosts, and then calculating the “aggregate” values (but it must be decided first how to calculate these “aggregate” values: a medium, a sum, ...).

## Condor

The Condor command *condor\_status* is used to get a summary of information from ClassAds about the resources in the pool.

Using the *-l* option, it is possible to display the entire ClassAds of a resource, that by default include:

- the architecture and the operating system of the machine;
- the number of CPUs;
- the amount of physical memory;
- the relative floating point performance as determined via a Linpack benchmark (Kflops) and the relative integer performance as determined by a Dhrystone benchmark (Mips);
- the amount of disk space on this machine available for running jobs;
- the amount of currently available swap space;
- the machine’s Condor state (used by the owner, unclaimed [available to run Condor jobs], claimed [running a Condor job], ...) and the time at which the machine entered the current state;
- the load average;
- the period of inactivity on the system console keyboard or console mouse;

Also in Condor other “custom” attributes can be defined, and these are then “managed” as the default ClassAds.

The command *condor\_status* allows to display also some “aggregate” information for all the machines of the Condor pool, or for a subset of machines of the pool that meet a specific criterion (for example all the machines of the pools with a specific architecture, and running a specific operating system).

The “aggregate” information for a Condor pool (or a sub-pool) that can be displayed include:

- the total number of machines, specifying the claimed and the unclaimed machines;
- the total amount of memory;
- the total amount of disk space available to run jobs;
- the total MIPS and KFLOPS;

- the total number of “submitted” jobs, specifying the running and the “idle” jobs.

### 3 Information on jobs

#### 3.1 Information of a job published by default in the GIS

When LSF or Condor has been configured as job manager for a Globus machine, and the *-publish-jobs* option has been specified in the file:

*<globus-deploy-dir>/etc/grid-info-resource.conf*

by default some information about the jobs submitted to LSF/Condor via Globus, are published in the Grid Information Service as *GlobusQueueEntry* objectclasses, “under” the *GlobusQueue* objectclass, as shown in the following example (that refers to a Condor job, but for LSF the same attributes are published):

<b>dn</b>	<i>jobid=2481, queue=intel-linux, service=jobmanager-condor, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid</i>
<b>jobid</b>	2481
<b>objectclass</b>	GlobusTop
<b>objectclass</b>	GlobusQueueEntry
<b>localjobid</b>	2481
<b>specification</b>	<i>&amp;("executable" = "/home/sgaravat/ciao" \("stdout" = "/home/sgaravat/ox-cond.out" \("stderr" = "/home/sgaravat/ox-cond.err" \("count" = "1" \("arguments" = "one" "two" )</i>
<b>globaljobid</b>	<i>https://lxde02.pd.infn.it:1475/616/973770651/</i>
<b>count</b>	1
<b>status</b>	ACTIVE
<b>start time</b>	0
<b>schedspecific</b>	NULL
<b>ttl</b>	00:01:00
<b>lastupdate</b>	Thu Nov 09 11:51:07 GMT 2000

The meaning of the various fields is the following:

- *localjobid*: the LSF/Condor job id (note that for Condor only the cluster id, and not the process id, is reported);
- *specification*: the RSL string used to submit the job;
- *globaljobid*: the Globus id of the job

- *count*: this attribute reports the *count* parameter in the RSL string that has been used to submit the job;
- *status*: the status of the Globus job (ACTIVE, PENDING, ...);
- *start time*: this parameter should report the time the job was started, but the value is always equal to 0;
- *schedspecific*: this should be a catch-all attribute for use by any scheduler to provide some information, but in the current implementation is not defined.

### 3.2 Other information that could be provided by the underlying resource management system

This section describes which other information related to a Globus job could be made available and published in the GIS, using the commands and tools provided by the underlying resource management system.

#### LSF

The LSF command *bjobs* is used to view job information. For each job, the information that are displayed include:

- the job id;
- the name of the user;
- the status of the job;
- the submitting host;
- the executing host (for the jobs that are running);
- the submission time, and the time when the job started its execution;
- the pending or suspending reasons (for “idle” jobs);
- the name of the executable;
- the job resource usage (CPU time, memory, swap).

#### CONDOR

For Condor, the *condor\_q* command is used to display information about jobs in queue. All the attributes of a job ClassAd may be displayed; by default these include:

- the cluster and process id of the job;
- the name of the user;
- the time at which the job has been submitted to the job queue;
- the time at which the job first began running;
- the current status of the job;
- the size of the executable;
- the name of the executable;
- the host where the job is running;

- the CPU time accumulated by the job to date (but this parameter is updated only after any checkpoint of the job);
- the reason for which a job is not running (for “idle” jobs);
- the total number of bytes the job has read and written.

## 4 First proposal for modifying the default schema

This section presents a first, preliminary proposal for modifying the default schema, taking into account the information that are currently published in the GIS (dropping the attributes that in our opinion are not useful at all for our needs), and the other information that could be provided by the underlying resource management system (LSF/Condor).

First of all, we think that some information about the local resource management system should be provided, that could be represented by the following attributes:

- *hn*: the host name of the “front-end” machine;
- *service*: the name of the job manager (i.e. *jobmanager-lsf*, *jobmanager-condor*,...);
- *contact*: the string identifying the contact information;
- *ResourceManagementType*: defines the type of resource management system (LSF/Condor/...);
- *ResourceManagementVersion*: the version of the local resource management system. For Condor this value can be found, for example, using the command: *condor\_status*, while for LSF using the command *lsid*;
- *Gramversion*: the GRAM version.

We think that we can assume (or we must assume) that a farm can be represented by a set of queues, where a queue represents a set of “homogeneous” resources: when a job is submitted to a specific queue, it doesn’t matter in which host associated to this queue the job is dispatched. For LSF such queues are represented by the LSF queues, while for Condor we think that defining a queue as the set of machines of the Condor pool with the same architecture and the same operating system (as it is implemented now) is a correct approach.

In our opinion for each queue the following attributes should be considered:

- *Queue*: the name of the queue.  
For LSF this should be the name of the LSF queue.  
For Condor this name, as in the current implementation, should be “built” considering the name of the architecture and of the operating system (i.e. *intel-linux*) of the machines “associated” to this queue;
- *Architecture*: the architecture of the machines associated to this queue (we assume that all the machines “belonging” to this queue have the same architecture).

The value for this parameter could be found using the command: *uname -a* (for Unix platforms); in an heterogeneous LSF cluster the *uname* command has to be executed on one of the hosts associated to the queue: this is possible issuing the command with *lshosts -m*.

- *OpSys*: the operating system of the machines associated to this queue (assuming that all these machines run the same operating system).

For Unix platforms, the value for this parameter could be found using the command *uname -a* (as above, in LSF the command is executed on the proper host via *lshosts -m*);

- *TotalCpus*: the number of total CPUs “associated” to this queue.

For Condor this parameter should be found considering the total number of hosts in the Condor pool of the same architecture and of the same operating system of the front-end machine, as the attribute *totalnodes* is now calculated.

For LSF this parameter should be calculated considering all and only the hosts to which this queue can dispatch jobs (therefore not considering all the hosts known to LSF, as the *totalnodes* attribute is calculated in the current implementation): for this purpose the command *bqueues* can be used;

- *FreeCpus*: this parameter should represent the total number of free processors associated to this queue, processors able to run, in that moment, jobs submitted to that queue.

For Condor this parameter should be found considering the “unclaimed” nodes of the Condor pool of the same architecture and of the same operating systems of the front-end machine, as the *freenodes* attribute is now found.

For LSF we think that the method used to calculate the *freenodes* parameter in the current implementation is not correct: the *FreeCpus* parameter should be calculated considering only the hosts available to the queue (and not all the hosts in the cluster) and with the load parameters within the limits defined in the queue resource requirements and scheduling policies (this can be achieved taking the requirements from the output of *bqueues* and then using the *lshosts* command to find the current available hosts);

- *TotalJobs*: the number of jobs submitted to this queue, jobs that have not already been completed.

For LSF this parameter could be found using the *bqueues* command.

For Condor it could be calculated using the *condor\_status* and/or *condor\_q* commands;

- *RunningJobs*: the number of jobs submitted to this queue that are currently running.

For LSF this parameter could be calculated as the parameter *maxrunningjobs* is calculated in the current implementation (that is, using the command *bqueues*).

For Condor this value could be calculated using the *condor\_status* and/or *condor\_q* commands;

- *IdleJobs*: the number of jobs submitted to this queue, jobs that are not running since they are waiting for available resources.

This value could be calculated as difference between *TotalJobs* and *RunningJobs*;

- *MaxRunningJobs*: the maximum number of running jobs allowed for this queue. For LSF this is defined in the queue configuration and can be obtained with the *bqueues* command.

For Condor this value should indicate that there is not such limit;

- *MaxTotalJobs*: the maximum number of jobs (running and idle) allowed for this queue.

For LSF this can be obtained with the *bqueues* command, while for Condor there is not such limit;

- *Status*: the status of the queue (if it is ready or not to dispatch jobs to the executing machines).

For LSF this parameter could be found using the *bqueues* command, as it is now used to find the *status* attribute.

For Condor this parameter could always be defined as “Active”;

- *RunWindows*: the time windows that define when the queue is active, that is ready to dispatch jobs to the executing machines.  
For LSF this parameter could be found using the *bqueues* parameter.  
For Condor this value should indicate that the queue is always active;
- *Priority*: the priority of the queue.  
For LSF this parameter should be found considering the priority of the LSF queue, that can be displayed using the *bqueues* command, as it is now used to find the *priority* attribute.  
For Condor this parameter can be undefined;
- *MaxCpuTime*: the maximum CPU time allowed for jobs submitted to this queue.  
For LSF this value could be found using the *bqueues* command.  
For Condor this value can be undefined, in order to specify that there is not such this constraint;
- *MaxWallTime*: the maximum wall clock time allowed for jobs submitted to this queue.  
For LSF this value could be found using the *bqueues* command.  
For Condor this value can be undefined, since there is not such this constraint;

It would be very useful to include also some information related with the total CPU power of the machines associated to the queue, specifying also the CPU power for the “free” machines. Unfortunately this information can be provided by Condor (that is able to provide the MIPS and the KFLOPS for a set of machines of the Condor pool, also specifying these values only for the unclaimed hosts) but not for LSF (for LSF only a CPU factor, defined by hand by the administrator, can be provided).

We think that we can assume (or we must assume) that all the machines “associated” to a queue have the same scheduling policies that define when a job submitted to the queue starts running, and when a running job must be suspended.

These policies are very important information, and therefore they should be published as an attribute of the queue. For LSF it is possible to have the policies of the queues using the command *bqueues*, while for Condor, assuming that all the machines of the pool consider the same policies, they can be found examining the ClassAds of a single machine (for example the front-end machine) with the command *condor\_status*.

It must be understood, however, how to publish, in a “uniform” and common way, the scheduling policies for different resource management systems, since the different resource management systems consider different criteria, parameters, etc.. to decide when to start/suspend the jobs.

It must be understood if it is viable also to publish, as information for the queue, the users allowed to submit jobs on this queue.

For Condor this could be done considering the *grid-mapfile* of the front-end machine, since all authorized users are allowed to submit jobs to the Condor pool (assuming that the administrators of the Condor resources didn’t define different policies).

For LSF it should be necessary to find, using the command *bqueues*, the local users (or groups of local users) allowed to submit jobs to this queue, and then “map” these local users to “Grid users” (this could be done considering the *grid-mapfile*).

Some information about the jobs submitted to the queue, jobs that have not been already completed, should be published as well.

In our opinion the following attributes could be considered:

- *GlobalJobId*: the Globus id of the job, as currently defined in the *globaljobid* attribute;
- *LocalJobId*: the id of the job in the underlying resource management system (the LSF job id, the Condor cluster id and the process id);
- *GlobalUser*: the id of the Grid user, that is the subject line from his/her certificate;
- *LocalUser*: the username in the local system “mapped” to the considered Grid user;
- *Status*: the status of the job (running, pending,...) as now defined in the *status* attribute;
- *PendigReason*: the reason for which a job is not running.  
This value can be obtained with the command *condor\_q* for Condor, and with the command *bjobs* for LSF;
- *RSLCommand*: as in the current implementation for the *specification* attribute, the RSL string used to submit the job;
- *SubmitTime*: the time at which the job has been submitted.  
For LSF this value can be found using the *bjobs* command, while for Condor the *condor\_q* command can be considered;
- *StartTime*: the time at which the job first began running.  
For LSF this value can be found using the *bjobs* command, while for Condor the *condor\_q* command can be used;
- *WallClockTime*: the wall clock time “accumulated” for this job.  
For LSF this value can be calculated using the *bjobs* command, while for Condor the *condor\_q* command can be considered to find this value.

It would be interesting to publish also the CPU time for the job. Unfortunately this parameter is available for LSF, but not for Condor (Condor updates this parameter only after a checkpointing, and we cannot assume that the checkpointing mechanism is used).

The following tables summarize the new proposed schema:

<b>dn</b>	<b>ServiceJobManager=</b>
hn	
service	
contact	
ResourceManagementType	
ResourceManagementVersion	
Gramversion	

<b>dn</b>	<b>Queue=</b>
Queue	
Architecture	
OpSys	
TotalCpus	
FreeCpus	
TotalJobs	
RunningJobs	
IdleJobs	
MaxTotalJobs	
MaxRunningJobs	
Status	
RunWindows	
Priority	
MaxCpuTime	
MaxWallTime	

<b>dn</b>	<b>QueueEntry=</b>
GlobalJobId	
LocalJobId	
GlobalUser	
LocalUser	
Status	
PendingReason	
RSLCommand	
SubmitTime	
StartTime	
WallClockTime	

#### 4.1 How to modify the default schema

To modify the default schema, in order to publish this new and “custom” information, there are two possibilities.

For the first option it is necessary to modify the queue scripts (<globus-deploy-dir>/libexec/globus-script-\*-queue). Then it is necessary to define the new variables to the *globus\_gram\_scheduler\_t* structure in the header file:

```
<globus>/gram/libraries/jobmanager/globus_gram_job_manager.h.in
```

and the files:

```
<globus>/gram/libraries/jobmanager/globus_gram_scheduler.c
<globus>/gram/programs/reporter/globus_gram_scheduler.c
```

must be modified as well, in order to write the set of variables/values from the queue scripts to a cldif file.

The second possibility is that the new information belongs to separate LDAP objects from the ones created by the scheduler code (the *GlobusServiceJobManager*, *GlobusQueue* and *GlobusQueueEntry* objectclasses).

Therefore one or more information providers (scripts/programs) must be developed: these programs must create the required LDAP objects, “emitting” LDIF objects on stdout.

These new information providers must then be “inserted” in the file:

```
<globus-deploy-dir>/etc/grid-info-resource.conf.
```

We think the second approach is preferable, since it is more straightforward and allows much more flexibility.

## 5 Conclusions and future work

This document presented a preliminary discussion about the information related to the characteristics and status of a farm, and the information concerning the Globus jobs submitted to this farm, that could/should be published in the Grid Information Service.

Only LSF and Condor have been considered as local resource management systems.

The information published by default have been examined, in order to understand which ones are useful or useless for our needs, and other information provided by the tools/commands of the underlying resource management systems have been analyzed as well, in order to evaluate if it is viable to publish them.

This, of course, must be considered as a first step. It will become much more clear in the future which information must be published in the Information Service, in order to be used for example by a broker that must decide to which resources the jobs must be submitted.

We think that, first of all, this document must be “integrated” taking into account the other local resource management systems that must be considered in a Grid environment (for example PBS).

When the common set of attributes that must be published for all these resource management systems has been defined, we could start writing the scripts/programs that create and “populate” the LDAP objects implementing these new “custom” parameters.

If this information published in the Information Service is not enough, it will be necessary to implement specific “Information Providers”.

## Appendix A – Default schema for GRAM information

```
#$Header: /nfs/globus1/src/master/openldap-  
2.0/servers/slapd/schema/grid.gramscheduler.schema,v 1.3 2000/09/13 02:11:23  
june Exp $  
#$Log: grid.gramscheduler.schema,v $  
#Revision 1.3 2000/09/13 02:11:23 june  
#resolve changes between 2.0 & 2.0.1  
#  
#Revision 1.2 2000/09/12 22:28:19 june  
#Schema file for grid information services
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.1  
    NAME 'contact'  
    DESC 'A string identifying the contact information'  
    EQUALITY caseIgnoreMatch  
    SUBSTR caseIgnoreSubstringsMatch  
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
    NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.2  
    NAME 'cputype'  
    DESC 'A string identifying the cpu information'  
    EQUALITY caseIgnoreMatch  
    SUBSTR caseIgnoreSubstringsMatch  
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
    NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.3  
    NAME 'deploydir'  
    DESC 'A string identifying the deploy directory'  
    EQUALITY caseIgnoreMatch  
    SUBSTR caseIgnoreSubstringsMatch  
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  
    SINGLE-VALUE  
    NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.4  
    NAME 'dispatchtype'  
    EQUALITY caseIgnoreMatch  
    ORDERING caseIgnoreOrderingMatch  
    SUBSTR caseIgnoreSubstringsMatch  
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
    SINGLE-VALUE  
    NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.5  
    NAME 'freenodes'  
    DESC 'An integer identifying the number of free nodes'  
    EQUALITY integerMatch  
    ORDERING caseIgnoreOrderingMatch  
    SUBSTR caseIgnoreSubstringsMatch  
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
    SINGLE-VALUE  
    NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.6  
    NAME 'gramsecurity'  
    DESC 'A string identifying the gram security module information'  
    EQUALITY caseIgnoreMatch
```

ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.7  
NAME 'gramversion'  
DESC 'A number identifying the gram version'  
EQUALITY caseIgnoreMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.8  
NAME 'gramversiondate'  
DESC 'A number identifying the gram version date'  
EQUALITY caseIgnoreMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.9  
NAME 'hn'  
DESC 'A string identifying the host name'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.10  
NAME 'jobwait'  
DESC 'A string identifying the waiting jobs'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.11  
NAME 'lastupdate'  
DESC 'A time string identifying the lastupdate time'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.12  
NAME 'machinetype'  
DESC 'A string identifying the machine type'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.13  
NAME 'manufacturer'  
DESC 'A string identifying the manufacturer of the machine'

EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.14  
NAME 'maxcount'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.15  
NAME 'maxcputime'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.16  
NAME 'maxjobsinqueue'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.17  
NAME 'maxrunningjobs'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.18  
NAME 'maxsinglememory'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.19  
NAME 'maxtime'  
EQUALITY integerMatch  
ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.20  
NAME 'maxtotalmemory'  
EQUALITY integerMatch

ORDERING caseIgnoreOrderingMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.21  
NAME 'objectname'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.22  
NAME 'osname'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.23  
NAME 'osversion'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.24  
NAME 'priority'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.25  
NAME 'queue'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.26  
NAME 'schedulerspecific'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.27  
NAME 'schedulingtype'  
EQUALITY caseIgnoreMatch  
SUBSTR caseIgnoreSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44  
SINGLE-VALUE  
NO-USER-MODIFICATION )

attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.28  
NAME 'schedulerversion'  
DESC 'A number identifying the scheduler version'

```
EQUALITY numericStringMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.29
  NAME 'service'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
  NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.30
  NAME 'status'
  EQUALITY integerMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
  NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.31
  NAME 'totalnodes'
  EQUALITY integerMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
  NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.32
  NAME 'ttl'
  EQUALITY integerMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
  NO-USER-MODIFICATION )
```

```
attributetype ( 1.3.6.1.4.1.3536.2.3.3.1.33
  NAME 'whenactive'
  EQUALITY integerMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE
  NO-USER-MODIFICATION )
```

```
objectclass ( 1.3.6.1.4.1.3536.2.4.1.1
  NAME 'GlobusTop'
  SUP top
  ABSTRACT
  MUST objectClass )
```

```
objectclass ( 1.3.6.1.4.1.3536.2.4.1.2
  NAME 'GlobusDaemon'
  SUP GlobusTop
  ABSTRACT
  MUST objectClass )
```

```
objectclass ( 1.3.6.1.4.1.3536.2.4.1.3
  NAME 'GlobusService'
```

```
SUP GlobusDaemon
ABSTRACT
MUST objectClass )
```

```
objectclass ( 1.3.6.1.4.1.3536.2.4.1.4
  NAME 'GlobusServiceJobManager'
  SUP GlobusTop
  STRUCTURAL
  MAY ( objectname $ service $ hn $ contact $ schedulerversion $
        schedulerlytype $ deploydir $ cputype $ osversion $ osname $
        manufacturer $ machinetype $ gramversion $ gramversiondate $
        gramsecurity $ lastupdate $ ttl ) )
```

```
objectclass ( 1.3.6.1.4.1.3536.2.4.1.5
  NAME 'GlobusQueue'
  SUP GlobusTop
  STRUCTURAL
  MAY ( queue $ maxtime $ maxcputime $ maxcount $ maxrunningjobs $
        maxjobsinqueue $ maxtotalmemory $ maxsinglememory $
        totalnodes $ freenodes $ whenactive $ status $
        dispatchtype $ priority $ jobwait $ schedulerstpecific $
        ttl $ lastupdate ) )
```