NORDUGRID-TECH-26
5/10/2011

THE NORDUGRID GRIDFTP SERVER

Description and Administrator's Manual

A. Konstantinov*, D. Cameron

*aleks@fys.uio.no

Contents

1 Introduction

The NorduGrid [1] GridFTP service (GFS) has historically been the gateway between Grid users and their jobs running on ARC-enabled resources. It provides an interface for secure job submission and retrieval with authorization and authentication based on the Grid Security Infrastructure [4], and consists of a standard GridFTP server with NorduGrid modifications on top to handle Grid jobs. These jobs are then processed and sent to the local batch system by the Grid Manager (GM).

With the evolution of Grid technologies, and the need for more standardized interfaces to allow different Grid products to communicate with each other, the Hosting Environment Daemon (HED) [3] framework and associated Advanced Resource Execution Service (A-REX) web service were developed to replace the aging GFS and GM with a modern standards-compliant product. However, the transition to a new framework takes time and so the legacy GFS is still supported and maintained until such time as it becomes obsolete. This document is designed for administrators who wish to run the GFS alongside A-REX and describes how to configure and run the GFS service. For installation of the GFS, up to date instructions may be found on the NorduGrid website **`http://www.nordugrid.org`**.

2 Main Concepts

A detailed summary of job workflow in ARC may be found in the A-REX technical documentation [5]. The following instructions assume knowledge of concepts such as the *session directory*, job states and how data management is performed by A-REX.

The GFS provides a means to map GSI identities to local usernames, and thus can expose a local filesystem to the Grid using a highly configurable set of authentication policies. It also allows Grid job submission by providing a way for clients to upload a job description to a Grid resource, and for that job to be executed on a resource under a mapped local username.

Local file access in the GFS is implemented through plugins (shared libraries). There are 3 plugins provided: *fileplugin.so, gaclplugin.so* and *jobplugin.so* . The *fileplugin.so* is intended to be used for plain file access with the configuration sensitive to the user subject and is not necessary for setting up a NorduGrid compatible site. The *gaclplugin.so* uses GACL [6] to control access to the local file system. The *jobplugin.so* uses information about jobs being controlled by A-REX and provides access to session directories of the jobs owned by the user. It also provides an interface (virtual directory and virtual operations) to submit, cancel, clean, renew credentials and obtain information about the job.

To make GFS to inter-operate with other parts of ARC only one *jobplugin.so* needs to be configured.

3 Configuration

The GFS configuration is done through a single INI-style configuration file. The XML-style configuration supported by A-REX are not supported by the GFS, and so care must be taken if using the GFS with A-REX configured using XML-style configuration. The safest option is to use the same INI-style file for A-REX and the GFS. The default location of the GFS configuration file is

- */etc/arc.conf*

A different configuration file location can be specified by the environment variable ARC_CONFIG. The configuration file consists of empty lines, lines containing comments (lines starting with #) or configuration commands. It is separated into sections. Each section starts with a string containing

- *[section name/subsection name/subsubsection name].*

Each section continues until the next section or until the end of the file. The configuration file can have commands for multiple services/modules/programs. Each service has its own section named after it. The GFS uses the *[gridftpd]* section and sub-sections, along with other authorization-related sections. Commands in section *[common]* apply to all services configured in the configuration file. Command lines have the format

- *name="arguments string"*.

Example configurations are shown in Section 6.

3.1 General Configuration Parameters

The following parameters are defined in the *[gridftpd]* section of the configuration file.

- ***pidfile***=*path* – specifies file where process id of the gridftpd process will be stored. Defaults to */var/run/gridftpd.pid* if running as root and *$HOME/gridftpd.pid* otherwise.
- ***logfile***=*path* – specifies name of file for logging debug/informational output. Defaults to */var/log/arc/gridftpd.log*.
- ***logsize***=*size number* – restricts log file size to *size* and keeps *number* archived log files. If installed from packages, the log properties are managed by logrotate. If logrotate or another external log management tool is used then *logsize* should not be used.
- ***logreopen***=*yes|no* – specifies if log file must be opened before writing each record and closed after that. By default log file is kept open all the time (default is no).
- ***debug***=*number* – specifies level of debug information. More information is printed for higher levels. Currently the highest effective number is 5 (DEBUG) and lowest 0 (FATAL). Defaults to 3 (INFO).
- ***port***=*number* - specifies TCP/IP port number. Default is 2811.
- ***include***=*path* - include contents of another file. Generic commands cannot be specified there.
- ***encryption***=*yes|no* - specifies if server will allow data transfer to be encrypted. Default is yes.
- ***pluginpath***=*path* - specifies the path where plugin libraries are installed. Default is *$(ARC_LOCATION)/lib/arc*.
- ***allowunknown***=*yes|no* - if set to *yes*, clients are not checked against the grid-mapfile. Hence only access rules specified in this configuration file will be applied. Default is no.
- ***firewall***=*hostname* - use IP address of the *hostname* in response to PASV command instead of IP address of a network interface of the computer. An IP address can be used instead of *hostname*. This command may be useful if the server is situated behind a NAT.
- ***voms_processing***=*relaxed|standard|strict|noerrors* – specifies how to behave if failure happens during VOMS processing.
  - relaxed – use everything that passed validation.
  - standard – same as relaxed but fail if parsing errors took place and VOMS extension is marked as critical. This is a default.
  - strict – fail if any parsing error was discovered.
  - noerrors – fail if any parsing or validation error happened.

  Default is *standard*.
- ***unixgroup***=*group rule* - define local UNIX user and optionally UNIX group to which user belonging to specified authorization *group* is mapped (see Section 3.3 for definition of group). Local names are obtained from the specified *rule*. If the specified rule could not produce any mapping, the next command is used. Mapping stops at first matched rule. The following rules are supported:
  - ***mapfile*** *file* - the user's subject is matched against a list of subjects stored in the specified file, one per line followed by a local UNIX name.
  - ***simplepool*** *directory* - the user is assigned one of the local UNIX names stored in a file *directory/pool*, one per line. Used names are stored in other files placed in the same *directory*. If a UNIX name was not used for 10 days, it may be reassigned to another user.

  - ***lcmaps*** *library directory database* - call LCMAPS functions to do mapping. Here *library* is the path to the shared library of LCMAPS, either absolute or relative to *directory*; *directory* is the path to the LCMAPS installation directory, equivalent to the LCMAPS_DIR variable; *database* is the path to the LCMAPS database, equivalent to the LCMAPS_DB_FILE variable. Each argument except *library* is optional and may be either skipped or replaced with '*'. It is important to ensure that no configured LCMAPS plugin performs switch of local user identity (setuid). That may interfere with way the GFS handles local user identities.
  - ***mapplugin*** *timeout plugin* [*arg1* [*arg2* [...]]] - run external *plugin* executable with specified arguments. Execution of *plugin* may not last longer than *timeout* seconds. A rule matches if the exit code is 0 and there is a UNIX name printed on *stdout*. A name may be optionally followed by a UNIX group separated by ':'. In arguments the following substitutions are applied before the plugin is started:
    * %D - subject of user's certificate,
    * %P - name of credentials' proxy file.

- ***unixvo***=*vo rule* - same as ***unixgroup*** for users belonging to Virtual Organization (VO) *vo*.
- ***unixmap***=*[unixname][:unixgroup] rule* - define a local UNIX user and optionally group used to represent connected client. *rule* is one of those allowed for ***authorization groups*** (see Section 3.3) and for ***unixgroup***/***unixvo***. In case of a mapping rule, username is the one provided by the rule. Otherwise the specified *unixname:unixgroup* is taken. Both *unixname* and *unixgroup* may be either omitted or set to '*' to specify missing value.
- ***groupcfg***=*name* - is put into subsections representing a plugin or [group] section and defines if that section is effective. The only unaffected option is ***groupcfg***. If name is empty (or no groupcfg is used at all), following lines apply to all users.

3.2 Plugin Configuration

Subsections of the *gridftpd* section specify plugins which serve the virtual FTP path (similar to the UNIX mount command). The name of the subsection is irrelevant but it is useful to use a name related to the plugin, e.g. [gridftpd/jobs] for the *jobplugin*. Inside the subsection, the following commands are supported:

- ***plugin***=*library_name* - use plugin *library_name* to serve virtual path.
- ***path***=*path* - virtual path to serve.

The GFS comes with 3 plugins: *fileplugin.so, gaclplugin.so* and *jobplugin.so*.

3.2.1 JobPlugin

jobplugin.so supports the following options:

- ***configfile***=*path* - defines non-standard location of the A-REX configuration file,
- ***allownew***=*yes|no* - specifies if new jobs can be submitted. Default is *yes*.
- ***unixgroup***/***unixvo***/***unixmap*** - same options as in the top-level GFS configuration. If the mapping succeeds, the obtained local user will be used to run the submitted job.
- ***remotegmdirs***=*control_dir session_dir [drain]* - specifies control and session directories under the control of another A-REX to which jobs can be assigned (see Section 5). Remote directories can be added and removed without restarting the GFS. However, it may be desirable to drain them prior to removal by adding the *"drain"* option. In this case no new jobs will be assigned to these directories but their contents will still be accessible.
- ***maxjobdesc***=*size* - specifies maximal allowed size of job description in bytes. Default value is 5MB. If value is missing or set to 0 no limit is applied.

3.2.2 FilePlugin

fileplugin.so supports the following options:

- ***mount***=*path* - defines the place on local filesystem to which file access operations apply.
- ***dir***=*path options* - specifies access rules for accessing files in *path* (relative to virtual and real path) and all the files below.
  options is a list of the following keywords:
  - ***nouser*** - do not use local file system rights, only use those specified in this line.
  - ***owner*** - check only file owner access rights.
  - ***group*** - check only group access rights.
  - ***other*** - check only "others" access rights.

  The options above are exclusive. If none of the above are specified, the usual UNIX access rights are applied.
  - ***read*** - allow reading files.
  - ***delete*** - allow deleting files.
  - ***append*** - allow appending files (does not allow creation).
  - ***overwrite*** - allow overwriting of existing files (does not allow creation, file attributes are not changed).
  - ***dirlist*** - allow obtaining list of the files.
  - ***cd*** - allow to make this directory current.
  - ***create*** *owner:group permissions_or:permissions_and* - allow creating new files. File will be owned by *owner* and owning group will be *group*. If '*' is used, the user/group to which connected user is mapped will be used. The permissions will be set to *permissions_or* & *permissions_and* (the second number is reserved for future usage).
  - ***mkdir*** *owner:group permissions_or:permissions_and* - allow creating new directories.

3.2.3 GACLPlugin

gaclplugin.so supports the following options:

- ***gacl***=*gacl* - GACL XML.
- ***mount***=*path* - local path served by plugin.

The GACL XML may contain variables which are replaced with values taken from the client's credentials. The following variables are supported:

$subject - subject of user's certificate (DN),

$voms - subject of VOMS[2] server (DN),

$vo - name of VO (from VOMS certificate),

$role - role (from VOMS certificate),

$capability - capabilities (from VOMS certificate),

$group - name of group (from VOMS certificate) .

Additionally, the root directory must contain a *.gacl* file with initial ACLs. Otherwise the rule will be "deny all for everyone".

3.3 Authorization

ARC services which have to authorize remote client applications use the notion of *group* for authorization purposes. Each *group* is made of *rules* applied sequentially. If a client's credentials pass *all rules*, the client is treated as belonging to the specified *group*.

Each group is represented by a top level section named [***group***] or its subsection. Each such section represents a separate authorization group and its name is given by the ***name*** command inside that section. If there is no ***name*** command then the name of the subsection is used.

Authorization is performed by applying set of rules. Rules obey same format as the rest of the configuration file. Each rules command consists of a *rule word* prepended with optional *modifiers* - [+|-][!]

+ accept credential if matches the following rule (positive match, default action),

- reject credential if matches the following rule (negative match),

! invert matching. Match is treated as non-match. Non-match is treated as match, either positive ("+" or nothing) or negative ("-").

Processing of rules in every group stops after the first positive or negative match, or failure is reached. If a rule does not match then processing continues. Failures are rule-dependant and may be caused by conditions like a missing file, unsupported rule, etc.

The following *rule words* and arguments are supported:

- [***subject***]=*subject* [*subject* [...]] - match user with one of specified subjects
- ***file***=[*filename* [...]] - read rules from specified files (format of file is similar to Globus grid-mapfile with user names ignored)
- ***remote***=[*ldap://host:port/dn* [...]] - match user listed in one of specified LDAP directories (uses network connection hence can take time to process)
- ***voms***=*vo group role capabilities* - accept user with VOMS proxy with specified *vo*, *group*, *role* and *capabilities*. '*' can be used to accept any value
- ***vo***=[*vo* [...]] - match user belonging to one of specified Virtual Organizations as defined in *vo* configuration section (see below).
- ***group***=[*groupname* [*groupname* [...]]] - match user already belonging to one of specified groups.
- ***plugin***=*timeout plugin* [*arg1* [*arg2* [...]]] - run external *plugin* (executable or function in shared library) with specified arguments. Execution of *plugin* may not last longer than *timeout* seconds. If *plugin* looks like *function@path* then function *int function(char*,char*,char*,...)* from shared library *path* is called (*timeout* is not functional in that case). Rule matches if exit code is 0. In arguments following substitutions are applied before plugin is started:
  - %D - subject of user's certificate,
  - %P - name of credentials' proxy file.
- ***lcas***=*library directory database* - call LCAS functions to check rule. Here *library* is path to shared library of LCAS, either absolute or relative to *directory*; *directory* is path to LCAS installation directory, equivalent of LCAS_DIR variable; *database* is path to LCAS database, equivalent to LCAS_DB_FILE variable. Each arguments except *library* is optional and may be either skipped or replaced with '*'.
- ***all*** - accept any user

Here is an example of authorization group:

```
[group/admins]

-subject="/O=Grid/OU=Wrong Place/CN=Bad Person"
file="/etc/grid-security/internal-staff"
voms="nordugrid * * admin"
```

In this example the following rules are applied to determine whether the identity presented is part of the group "admins":

- If the identity is "/O=Grid/OU=Wrong Place/CN=Bad Person" it is rejected
- If the identity is in the mapfile "/etc/grid-security/internal-staff" the identity is accepted
- If the identity is a VOMS proxy with a NorduGrid VO extension and admin capability, the identity is accepted

3.3.1 Virtual Organizations

VOs are defined in the *vo* configuration section. The following commands are supported:

- ***vo***=*vo_name* - specifies name of VO. Mandatory command.
- ***file***=*path* - path to file which contains list of users' DNs belonging to VO.
- ***source***=*URL* - specifies URL from which list of users may be obtained. May be multiple.

4 Running the service

An initialization script *gridftpd* for the GFS is provided in *$ARC_LOCATION/etc/init.d* (or equivalent depending on architecture).

```
Usage: gridftpd {start|stop|status|restart|reload|condrestart}
```

Upon starting and depending on the configured log level, messages will be logged in the log file specified in the configuration file.

5 Using Multiple A-REX services Under One GFS

For large clusters, using a single machine for all input and output file transfer, as well for the interaction with the LRMS and Information System, can limit the job throughput of the cluster. Running several A-REXes on separate hosts can help spread the hardware and network load. A single GFS can feed jobs to several A-REXes, hence a cluster with many A-REXes still appears as a single site to the outside world. When a job is submitted, the GFS jobplugin assigns a random control directory to use for the job from the main *controldir* specified in the A-REX configuration and any extra *remotegmdirs* specified in the jobplugin configuration. Note that it is not necessary to run an A-REX on the GFS host, in other words a configuration with only *remotegmdirs* is possible.

Each control directory is used by a separate A-REX, therefore for every *remotegmdirs* command in the GFS configuration, there must be a A-REX running which defines the corresponding *controldir* and *sessiondir* in its configuration file. Each A-REX can run independently on its own host, the only requirement is that the control and session directories must be accessible on the GFS host, and the GFS user must have write access to these directories. It is recommended that these directories are local to the GFS host and exported (via NFS for example) to the other hosts, rather than being on a remote filesystem and exported to the GFS host. This means that any glitches in the network do not cause the GFS host to hang. It is also important that the local user accounts on each host must be synchronised with the GFS host. Any A-REX is not aware that any other A-REXes are running, as they only see what the GFS decides should go into their own control directory. All communication between the GFS and a A-REX is through the A-REX's control directory. Note that in remote control directories there is no way to specify control directories per

user, as with the *control* command. Only the *controldir* command can be used and all users will use the same control directory.

One feature of this design is that multiple A-REXes can share the same LRMS, and hence compete with each other to submit jobs. Therefore any LRMS settings in the configuration files must be carefully matched in order not to bias one A-REX over another. In most cases each host's configuration file can be identical apart from the control and session directories. Some configuration sections such as the GFS and infosys sections will be ignored by the remote A-REX hosts as these services are not running.

Cacheing can be set up in a variety of different ways. Each A-REX can have its own cache, completely independent from any other, which will lead to popular files being replicated in many caches. Or, all caches can be shared with all A-REXes, which means no replication between caches but heavy intra-site network traffic if the cache file systems are hosted on different hosts. Another option is to give each A-REX its own cache, but access to the other caches as remote caches. This avoids replicating files and intra-site network traffic. Replication can still be enabled by specifying "replicate" as the link_path for remote cache dirs, and the advantage of this is that files are copied from the remote cache rather than being downloaded again from source.

When setting up an extra A-REX, it is important that no other services (GFS, infosys) run on the host. The instances of these services running on the "main" host take care of all the A-REXes. It is recommended therefore that packages containing *gridftpd* and *grid-infosys* are not installed on these hosts, and if they are, care should be taken to remove or disable scripts which could be started automatically (usually in *$ARC_LOCATION/etc/init.d/*). No host certificates are required for hosts which only run a A-REX instance but CA certificates are needed for contacting remote services when downloading and uploading job input and output files. Note that in this multiple A-REX set up, there is a one to one relationship between control and session directories, hence multiple *sessiondir* commands cannot be used in a A-REX configuration.

If no A-REX service is run on the GFS host then the option *infosys_compat="enable"* must be set in the *[infosys]* section of the GFS host configuration (the infosys and GFS must be on the same host). This means the legacy infoproviders in the infosys are used instead of those in A-REX. If this option is not set then the A-REX infoproviders must be used and A-REX started on the GFS host. However it is possible to start A-REX without any (local) control directories defined - the grid-manager thread for processing jobs will exit straight away but the thread running the infoproviders will continue and will collect information from all the remote A-REXes.

When adding a new *remotegmdirs* option to the configuration, there is no need to restart the GFS, as the configuration is read dynamically by the job plugin upon job submission. However the A-REX serving those new directories must be started before adding them to the GFS configuration. If an A-REX is to be removed from the system, it is best to set the *remotegmdirs* to a draining state (for how see description of this parameter in configuration section), so that running jobs are still accessible but no new jobs will be assigned there. When the A-REX has completed all jobs and all job output has been retrieved (for example there are no jobs found when running gm-jobs), it is safe to remove from the GFS configuration.

The status of all A-REXes is monitored by the information system through heartbeat files in each control directory. If all A-REXes are running ok then jobs can be submitted to the site. If some A-REXes are up and some are down then the site will publish a "degraded" state and no new jobs can be submitted. If all A-REXes are down then the site publishes a "critical" state and no new jobs can be submitted.

6 Configuration Examples

The examples presented below contain full configuration examples for the GridFTP server. The A-REX and information system configurations are not shown - this is described in other documents.

6.1 Simple Example

In the following minimal example we use a single static mapfile which contains all possible user mappings for this site.

```
[common]
hostname="myhost.org"
lrms="fork"
```

```
gridmap="/etc/grid-security/grid-mapfile"

[gridftpd]
debug="3"
encryption="no"
allowunknown="no"
maxconnections="200"

[gridftpd/jobs]
path="/jobs"
plugin="jobplugin.so"
```

6.2 Detailed Example

Here we configure a simple PBS based cluster according to the following use case. John is member of the VO "smscg" where he belongs to the group "atlas" and has been assigned the roles "production" and "test". Since groups and roles are fully decoupled, John can request proxies that can include one (or several) of the following different group-role combinations (termed "Fully Qualified Names" (FQAN)):

- /smscg (notice it's the same as /smscg/Role=NULL)
- /smscg/Role=production
- /smscg/Role=test
- /smscg/atlas
- /smscg/atlas/Role=production
- /smscg/atlas/Role=test

A-REX serves as front-end to a batch-system that provides a "low_prio_queue" and a "high_prio_queue". Assignment to the different queues is done via local user identites. More precisely, the local users "smscg001, smscg002, smscg003" will be assigned to the low_prio_queue, whereas users "smscgP001, smscgP002, smscgP003" to the high_prio_queue (the configuration of the batch-system to support this is out of scope of this example).

Users sending jobs to A-REX should be assigned to one of the queues depending on the credentials they present in their proxy certificate. The assignment shall look as follows:

- /smscg , /smscg/Role=test , /smscg/Role=production => shall map to one of the smscg00[1-3] local identities (thus low_prio_queue)
- /smscg/atlas , /smscg/atlas/Role=test , /smscg/atlas/Role=production => shall map to one of the smscgP00[1-3] local identities (thus high_prio_queue)

The following usage pattern is considered. User John first wants to run a monitoring job on the high_prio_queue. He performs a voms-proxy-init and specifies his "/smscg/atlas/Role=test" FQAN to be used. When he submits his monitoring-job, John will be mapped to one of the smscgP001, smscgP002, smscgP003 accounts. John's job will thus run on the high_prio_queue.

After submitting the monitoring job, John submits regular jobs with his FQAN "/smscg". These jobs will run on the low_prio_queue. Later John switches back to the FQAN "/smscg/atlas/Role=test" to fetch the result of his monitoring job.

The discrimination to what queue John is to be mapped is done with VO information only and not on the basis of the DN of John's certificate. Hence the choice to what queue to be mapped is under control of John (we silently presumed John knows the mappings at the source).

Notes:

- a DN based grid-mapfile is generated on the front-end with a default mapping entry for John. The grid-mapfile is only used by the information system (GIIS) to make the grid resource look eligible for jobs submitted by John.

- the DN based grid-mapfile per se does not permit John to access the grid resource under different local identities (e.g. once as smscg001 and later as smscgP001), since the first matching DN defines the local identity John is to be mapped to. This is not a flaw since NorduGrid has support for lcmaps, which allows a 're-mapping' of a user.
- the mapping of the FQAN to the local user identity (e.g. "/smscg" to local user "smscg001") shall be done with lcmaps (in detail the lcmaps framework + lcmaps voms plugins). Direct VOMS based mapping is also possible.

If user John creates a proxy certificate with the "grid-proxy-init" command instead of "voms-proxy-init", hence the proxy certificate will not contain any VO information and submits a job to A-REX (the matchmaking will still work, since it's done with John's DN) he shall not be authorized.

Example configuration:

```
[common]
pbs_bin_path="/usr/bin"
pbs_log_path="/var/spool/pbs/server_logs"
hostname="myhost.org"
lrms="pbs"


[vo]
# We will use this configuration block for a few purposes.
# 1. To generate grid-mapfile needed for information system.
#    For that purpose nordugridmap utility will have to be
#    run periodically.
# 2. To provide coarse-grained information to authorization
#    rules used to define authorization groups. If needed of
#    course.
id="smscg_vo"
vo="smscg_vo"

# Here we define path to file to which nordugridmap will write DNs of
# users matching rules below. Because we are going to use it as
# grid-mapfile for other purposes it is going to reside at default
# location.
file="/etc/grid-security/grid-mapfile"

# Now we tell nordugridmap to pull information from
# VOMRS/VOMSS/or_whatever_it_is_called_now service and to ask for
# users belonging to smscg VO.
source="vomss://voms.smscg.org:8443/voms/smscg"

# Now we specify default mapping to local *NIX id. It is possible to
# completely redefine mapping in [gridftpd] block. But this one will
# be used by information system to compute and present resources
# available to user.  Let's use one of lowest priority account defined
# in use-case.
mapped_unixid="smscg001"


[group]
# In this authorization group we are going to check if user presents
# any proof that he belongs to 'smscg' VO. We can use that information
# later to explicitly limit access to resources. If such access
# control is not needed this group can be removed.
name="smscg_auth"

# Here we can use internal support of ARC for VOMS attributes
```

```
# voms="smscg * * *"
# If we want to limit access to resources also by other VOMS
# attributes then other voms rules similar to those defined
# below in [gridftpd] section may be used.

# Or we can ask some external executable to analyze delegated
# credentials of user. In this example executable vomatch
# is called with first argument containing path to delegated
# proxy certificate and second - required VO name.
# plugin="10 /opt/external/bin/vomatch %P smscg"

# Or - probably preferred way in this use case - we can use
# LCAS to analyze delegated proxy.
# First element after '=' sign is path to LCAS library whatever
# it is called in current implementation. Second is LCAS installation
# path - it will be used to set environment variable LCAS_DIR.
# And third element is path to LCAS database file - it will be passed
# to environment variable LCAS_DB_FILE.
# Function 'lcas_get_fabric_authorization' of specified LCAS library
# will be called with following 3 arguments
#  1. char* pointer to string containing DN of user
#  2. gss_cred_id_t variable pointing at delegated credentials of user
#  3. char* pointer to empty string
# Returned 0 int value is treated as positive response
lcas="/opt/glite/lib/liblcas.so /opt/glite /opt/glite/share/lcas.db"

# As coarse grained solution it is also possible to check if user
# belongs to one of defined VOs as specified in _previously_ defined
# [vo] group. Here we refer to VO group smscg_vo defined above.
#vo="smscg_vo"


[gridftpd]
debug="2"
logfile="/var/log/arc/gridftpd.log"
logsize="10000000 2"
pidfile="/var/run/gridftpd.pid"
port="2811"
pluginpath="/usr/local/lib/arc"
encryption="no"

# By specifying 'no' here we limit users allowed to establish
# connection to this server to those specified in grid-mapfile. This
# may be not necessary if additional authorization is applied as done
# below. But this provides additional layer of protection so let it
# be.
allowunknown="no"

maxconnections="200"

# Here we start fine-grained user mapping. Let's first define few VOMS
# mappings using embedded functionality of ARC.  These lines should
# map Grid users to high-priority and low-priority *NIX users smscg001
# and smscgP001. Mind order - those with more attributes defined come
# first.  I do not know if missing attribute is passed by VOMS as
# empty string or as string containing NULL keyword. Here I assume
# empty string. If it is NULL then "" has to be replaced with NULL.
#unixmap="smscgP001 voms smscg atlas test *
```

```
#unixmap="smscgP001 voms smscg atlas production *
#unixmap="smscgP001 voms smscg atlas "" *
# These 3 lines are not needed if grid-mapfile defines default mapping
# to smscg001 user. But we can have them for consistence and if mapping
# to nobody is defined below for safety reasons.
#unixmap="smscg001 voms smscg "" test *
#unixmap="smscg001 voms smscg "" production *
#unixmap="smscg001 voms smscg "" "" *

# Instead of using multiple unixmap commands above we may define
# 2 authorization groups using [group] blocks. Let's say their
# names are smscg_low and smscg_high. Then 'group' matching rule
# may be used.
#unixmap="smscgP001 group smscg_high"
#unixmap="smscg001 group smscg_low"

# Or if we want to use all 6 local accounts and let mapping choose
# randomly within 2 group accounts 'simplepool' may be used. In
# example below 'unixgroup' ensures proper choice of group and
# 'simplepool' makes a choise from accounts in pool. Last argument
# specifies directory containing file named 'pool'. That file contains
# list of local user accounts. Also this directory will be used for
# writing information about current mappings.
#unixgroup="smscg_high simplepool /var/nordugrid/smscg_high"
#unixgroup="smscg_low simplepool /var/nordugrid/smscg_low"

# And mapping preferred in this use case - through LCMAPS.  First
# element after '=' sign is path to LCMAPS library whatever it is
# called in current implementation. Second is LCMAPS installation path
# - it will be used to set environment variable LCMAPS_DIR.  And third
# element is path to LCMAPS database file - it will be passed to
# environment variable LCMAPS_DB_FILE. Those 3 arguments are followed
# list of policy names.
# Function 'lcmaps_run_and_return_username' of specified LCMAPS library
# will be called with following arguments
#  1. char* pointer to string containing DN of user
#  2. gss_cred_id_t variable pointing at delegated credentials of user
#  3. char* pointer to empty string
#  4. char** pointer for chosen username.
#  5. int variable containing number of policies
#  6. char** list of policy names
# Expected 0 int value returned and argument 4 set. Value returned in
# 4th argument is used as username of local account.
unixmap="* lcmaps /opt/glite/lib/liblcmaps.so /opt/glite \
  /opt/glite/share/lcmaps.db policy1 policy2"

# Here we can specify mapping to some harmless local user account for
# safety reasons. If that account is not allowed to submit jobs to
# LRMS then this will also work as authorization effectively cutting
# off users without proper VOMS attributes.
unixmap="nobody:nobody all"


[gridftpd/jobs]
# This block defines job submission service
path="/jobs"
plugin="jobplugin.so"
```

```
# Line below specifies that this plugin/service is only available to
# users belonging to authorization group. If such behavior is not
# required then this line must be commented.
groupcfg="smscg_auth"


[queue/low_prio_queue]
name="low_prio_queue"
homogeneity="True"
scheduling_policy="FIFO"
comment="This queue is low priority"
nodecpu="adotf"
nodememory="512"
architecture="adotf"
opsys="Mandrake 8.0"
opsys="Linux-2.4.19"
benchmark="SPECINT2000 222"
benchmark="SPECFP2000 333"
cachetime="30"
timelimit="30"
sizelimit="5000"


[queue/high_prio_queue]
name="high_prio_queue"
homogeneity="True"
scheduling_policy="FIFO"
comment="This queue is high priority"
nodecpu="adotf"
nodememory="512"
architecture="adotf"
opsys="Mandrake 8.0"
opsys="Linux-2.4.19"
benchmark="SPECINT2000 222"
benchmark="SPECFP2000 333"
```

References

[1] The NorduGrid Collaboration. URL `http://www.nordugrid.org`. Web site.

[2] R. Alfieri et al. From gridmap-file to VOMS: managing authorization in a Grid environment. *Future Gener. Comput. Syst.*, 21(4):549–558, 2005. ISSN 0167-739X.

[3] D. Cameron et al. *The Hosting Environment of the Advanced Resource Connector middleware*. URL `http://www.nordugrid.org/documents/ARCHED_article.pdf`. NORDUGRID-TECH-19.

[4] I. Foster et al. A Security Architecture for Computational Grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92. ACM Press, November 1998. ISBN 1-58113-007-4.

[5] A. Konstantinov. *The ARC Computational Job Management Module - A-REX*. URL `http://www.nordugrid.org/documents/a-rex.pdf`. NORDUGRID-TECH-14.

[6] A. McNab. The GridSite Web/Grid security system: Research Articles. *Softw. Pract. Exper.*, 35(9):827–834, 2005. ISSN 0038-0644.