Table of Contents

SGE

ROADMAP

Name	Date	Status	Responsible
Common Documentation		DONE	ALL
IP on lcg-CE		DONE	IC
jobmanager on lcg-CE		DONE	CESGA
yaim on lcg-CE		DONE	LIP
Stress tests on lcg-CE		DONE	CESGA
Optimized SGE configuration		DONE	CESGA
Documentation: Installation Guide, Batch System Cookbook	June	DONE	LIP, CESGA
Official realease of SGE on lcg-CE	**July**		
IP on gLite-CE		DONE	
blah on gLite-CE	June	in progress	IC
APEL on gLite-CE	June	in progress	CESGA
yaim on gLite-CE	July		LIP
Stress tests on gLite-CE	August		CESGA
Installation Guide	August		LIP, CESGA
Official release of SGE on gLite-CE	**October**		

Instalation Guide SGE YAIM

If you just want to install SGE in your site the easiest would be to use yaim. The installation procedure is described in the "Installation Guide":

Installation Guide Version2

Installation Guide SGE gLite 3.1

If you want to know more details about how the integration has been done or if you want to do then installation manually you can find the details below.

Integrating SGE with Yaim

DISCLAIMER: THIS WORK IS STILL INCOMPLETE AND UNDER TESTING!

THANKS: THIS WORK WOULD NOT BE POSSIBLE WITHOUT JOAO PAULO MARTINS HELP, LIP SYSTEM ADMINISTRATOR.

This section reports the work which has been developed to distribute and install Sun Grid Engine together with the gLite/LCG middleware.

Sun Grid Engine needs a qmaster machine (which can be installed in the CE Gatekeeper) and several execution hosts (which must be installed in Grid Worker Nodes). Following the SGE V60u7_1 distribution, we have compiled the original source code under 32 bit machines running SL4, and built different rpms modules. The basic rpm packages which have to be present in Qmaster/Exec SGE machines are:

- edg-sge-utils-1.0.0-1: The equivalent SGE package of edg-pbs-utils-*_sl3 for configuring ssh
  - /opt/edg/etc/edg-sge-knownhosts.conf.template
  - /opt/edg/sbin/edg-sge-knownhosts

- sge-daemons-V60u7_1-2: The SGE daemons
  - /usr/local/sge/V60u7_1/bin/lx26-x86/sge_execd
  - /usr/local/sge/V60u7_1/bin/lx26-x86/sge_qmaster
  - /usr/local/sge/V60u7_1/bin/lx26-x86/sge_schedd
  - /usr/local/sge/V60u7_1/bin/lx26-x86/sge_shadowd

- sge-docs-V60u7_1-2: Documentation, Manuals and examples
  - /usr/local/sge/V60u7_1/3rd_party/3rd_party_licscopyrights
  - /usr/local/sge/V60u7_1/3rd_party/qmon/copyrights
  - /usr/local/sge/V60u7_1/3rd_party/qmon/ltree_changed.tar.gz
  - /usr/local/sge/V60u7_1/catman/cat/cat1/gethostbyaddr.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/gethostbyname.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/gethostname.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/getservbyname.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/hostnameutils.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qacct.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qalter.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qconf.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qdel.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qhold.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qhost.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qlogin.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qmake.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qmod.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qmon.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qping.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qresub.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qrls.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qrsh.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qselect.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qsh.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qstat.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qsub.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/qtcsh.1
  - /usr/local/sge/V60u7_1/catman/cat/cat1/sge_ckpt.1

- /usr/local/sge/V60u7_1/catman/cat/cat1/sge_intro.1
- /usr/local/sge/V60u7_1/catman/cat/cat1/sge_types.1
- /usr/local/sge/V60u7_1/catman/cat/cat1/sgepasswd.1
- /usr/local/sge/V60u7_1/catman/cat/cat1/submit.1
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_allocate_job_template.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_attributes.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_control.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_delete_job_template.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_exit.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_DRM_system.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_attribute.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_attribute_names.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_contact.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_next_attr_name.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_next_attr_value.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_next_job_id.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_vector_attribute.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_get_vector_attribute_names.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_init.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_job_ps.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_jobcontrol.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_jobtemplate.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_misc.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_release_attr_names.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_release_attr_values.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_release_job_ids.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_run_bulk_jobs.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_run_job.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_session.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_set_attribute.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_set_vector_attribute.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_strerror.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_submit.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_synchronize.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_version.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wait.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wcoredump.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wexitstatus.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wifaborted.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wifexited.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wifsignaled.3
- /usr/local/sge/V60u7_1/catman/cat/cat3/drmaa_wtermsig.3
- /usr/local/sge/V60u7_1/catman/cat/cat5/access_list.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/accounting.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/bootstrap.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/calendar_conf.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/checkpoint.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/complex.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/host_aliases.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/host_conf.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/hostgroup.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/project.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/qtask.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/queue_conf.5

- /usr/local/sge/V60u7_1/catman/cat/cat5/reporting.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sched_conf.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_aliases.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_conf.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_pe.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_priority.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_qstat.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sge_request.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/sgepasswd.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/share_tree.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/user.5
- /usr/local/sge/V60u7_1/catman/cat/cat5/usermapping.5
- /usr/local/sge/V60u7_1/catman/cat/cat8/sge_execd.8
- /usr/local/sge/V60u7_1/catman/cat/cat8/sge_qmaster.8
- /usr/local/sge/V60u7_1/catman/cat/cat8/sge_schedd.8
- /usr/local/sge/V60u7_1/catman/cat/cat8/sge_shadowd.8
- /usr/local/sge/V60u7_1/catman/cat/cat8/sge_shepherd.8
- /usr/local/sge/V60u7_1/doc/README-Autoinstall.txt
- /usr/local/sge/V60u7_1/doc/README-DRMAA.txt
- /usr/local/sge/V60u7_1/doc/arc_depend_irix.asc
- /usr/local/sge/V60u7_1/doc/arc_depend_solaris.asc
- /usr/local/sge/V60u7_1/doc/load_parameters.asc
- /usr/local/sge/V60u7_1/doc/logfile-trimming.asc
- /usr/local/sge/V60u7_1/examples/drmaa
- /usr/local/sge/V60u7_1/examples/drmaa/example.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto1.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto2.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto2_1.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto3.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto3_1.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto3_2.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto4.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto5.c
- /usr/local/sge/V60u7_1/examples/drmaa/howto6.c
- /usr/local/sge/V60u7_1/examples/jobs
- /usr/local/sge/V60u7_1/examples/jobs/array_submitter.sh
- /usr/local/sge/V60u7_1/examples/jobs/jobnet_submitter.sh
- /usr/local/sge/V60u7_1/examples/jobs/pascal.sh
- /usr/local/sge/V60u7_1/examples/jobs/pminiworm.sh
- /usr/local/sge/V60u7_1/examples/jobs/simple.sh
- /usr/local/sge/V60u7_1/examples/jobs/sleeper.sh
- /usr/local/sge/V60u7_1/examples/jobs/step_A_array_submitter.sh
- /usr/local/sge/V60u7_1/examples/jobs/step_B_array_submitter.sh
- /usr/local/sge/V60u7_1/examples/jobs/worker.sh
- /usr/local/sge/V60u7_1/examples/jobsbin
- /usr/local/sge/V60u7_1/examples/jobsbin/lx26-x86
- /usr/local/sge/V60u7_1/examples/jobsbin/lx26-x86/work
- /usr/local/sge/V60u7_1/man/man1/gethostbyaddr.1
- /usr/local/sge/V60u7_1/man/man1/gethostbyname.1
- /usr/local/sge/V60u7_1/man/man1/gethostname.1
- /usr/local/sge/V60u7_1/man/man1/getservbyname.1
- /usr/local/sge/V60u7_1/man/man1/hostnameutils.1
- /usr/local/sge/V60u7_1/man/man1/qacct.1
- /usr/local/sge/V60u7_1/man/man1/qalter.1

- /usr/local/sge/V60u7_1/man/man1/qconf.1
- /usr/local/sge/V60u7_1/man/man1/qdel.1
- /usr/local/sge/V60u7_1/man/man1/qhold.1
- /usr/local/sge/V60u7_1/man/man1/qhost.1
- /usr/local/sge/V60u7_1/man/man1/qlogin.1
- /usr/local/sge/V60u7_1/man/man1/qmake.1
- /usr/local/sge/V60u7_1/man/man1/qmod.1
- /usr/local/sge/V60u7_1/man/man1/qmon.1
- /usr/local/sge/V60u7_1/man/man1/qping.1
- /usr/local/sge/V60u7_1/man/man1/qresub.1
- /usr/local/sge/V60u7_1/man/man1/qrls.1
- /usr/local/sge/V60u7_1/man/man1/qrsh.1
- /usr/local/sge/V60u7_1/man/man1/qselect.1
- /usr/local/sge/V60u7_1/man/man1/qsh.1
- /usr/local/sge/V60u7_1/man/man1/qstat.1
- /usr/local/sge/V60u7_1/man/man1/qsub.1
- /usr/local/sge/V60u7_1/man/man1/qtcsh.1
- /usr/local/sge/V60u7_1/man/man1/sge_ckpt.1
- /usr/local/sge/V60u7_1/man/man1/sge_intro.1
- /usr/local/sge/V60u7_1/man/man1/sge_types.1
- /usr/local/sge/V60u7_1/man/man1/sgepasswd.1
- /usr/local/sge/V60u7_1/man/man1/submit.1
- /usr/local/sge/V60u7_1/man/man3/drmaa_allocate_job_template.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_attributes.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_control.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_delete_job_template.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_exit.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_DRM_system.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_attribute.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_attribute_names.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_contact.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_next_attr_name.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_next_attr_value.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_next_job_id.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_vector_attribute.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_get_vector_attribute_names.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_init.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_job_ps.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_jobcontrol.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_jobtemplate.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_misc.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_release_attr_names.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_release_attr_values.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_release_job_ids.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_run_bulk_jobs.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_run_job.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_session.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_set_attribute.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_set_vector_attribute.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_strerror.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_submit.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_synchronize.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_version.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wait.3

- /usr/local/sge/V60u7_1/man/man3/drmaa_wcoredump.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wexitstatus.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wifaborted.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wifexited.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wifsignaled.3
- /usr/local/sge/V60u7_1/man/man3/drmaa_wtermsig.3
- /usr/local/sge/V60u7_1/man/man5/access_list.5
- /usr/local/sge/V60u7_1/man/man5/accounting.5
- /usr/local/sge/V60u7_1/man/man5/bootstrap.5
- /usr/local/sge/V60u7_1/man/man5/calendar_conf.5
- /usr/local/sge/V60u7_1/man/man5/checkpoint.5
- /usr/local/sge/V60u7_1/man/man5/complex.5
- /usr/local/sge/V60u7_1/man/man5/host_aliases.5
- /usr/local/sge/V60u7_1/man/man5/host_conf.5
- /usr/local/sge/V60u7_1/man/man5/hostgroup.5
- /usr/local/sge/V60u7_1/man/man5/project.5
- /usr/local/sge/V60u7_1/man/man5/qtask.5
- /usr/local/sge/V60u7_1/man/man5/queue_conf.5
- /usr/local/sge/V60u7_1/man/man5/reporting.5
- /usr/local/sge/V60u7_1/man/man5/sched_conf.5
- /usr/local/sge/V60u7_1/man/man5/sge_aliases.5
- /usr/local/sge/V60u7_1/man/man5/sge_conf.5
- /usr/local/sge/V60u7_1/man/man5/sge_pe.5
- /usr/local/sge/V60u7_1/man/man5/sge_priority.5
- /usr/local/sge/V60u7_1/man/man5/sge_qstat.5
- /usr/local/sge/V60u7_1/man/man5/sge_request.5
- /usr/local/sge/V60u7_1/man/man5/sgepasswd.5
- /usr/local/sge/V60u7_1/man/man5/share_tree.5
- /usr/local/sge/V60u7_1/man/man5/user.5
- /usr/local/sge/V60u7_1/man/man5/usermapping.5
- /usr/local/sge/V60u7_1/man/man8/sge_execd.8
- /usr/local/sge/V60u7_1/man/man8/sge_qmaster.8
- /usr/local/sge/V60u7_1/man/man8/sge_schedd.8
- /usr/local/sge/V60u7_1/man/man8/sge_shadowd.8
- /usr/local/sge/V60u7_1/man/man8/sge_shepherd.8

- sge-utils-V60u7_1-2: Instalation scripts and SGE utilities
  - /usr/local/sge/V60u7_1/inst_sge
  - /usr/local/sge/V60u7_1/install_execd
  - /usr/local/sge/V60u7_1/install_qmaster
  - /usr/local/sge/V60u7_1/util
  - /usr/local/sge/V60u7_1/util/arch
  - /usr/local/sge/V60u7_1/util/arch_variables
  - /usr/local/sge/V60u7_1/util/bdb_checkpoint.sh
  - /usr/local/sge/V60u7_1/util/create_settings.sh
  - /usr/local/sge/V60u7_1/util/dl.csh
  - /usr/local/sge/V60u7_1/util/dl.sh
  - /usr/local/sge/V60u7_1/util/dl1.csh
  - /usr/local/sge/V60u7_1/util/dl2.csh
  - /usr/local/sge/V60u7_1/util/install_modules
  - /usr/local/sge/V60u7_1/util/install_modules/DB_CONFIG
  - /usr/local/sge/V60u7_1/util/install_modules/backup_template.conf
  - /usr/local/sge/V60u7_1/util/install_modules/inst_berkeley.sh
  - /usr/local/sge/V60u7_1/util/install_modules/inst_common.sh

- /usr/local/sge/V60u7_1/util/install_modules/inst_execd.sh
- /usr/local/sge/V60u7_1/util/install_modules/inst_execd_uninst.sh
- /usr/local/sge/V60u7_1/util/install_modules/inst_qmaster.sh
- /usr/local/sge/V60u7_1/util/install_modules/inst_qmaster_uninst.sh
- /usr/local/sge/V60u7_1/util/install_modules/inst_schedd_high.conf
- /usr/local/sge/V60u7_1/util/install_modules/inst_schedd_max.conf
- /usr/local/sge/V60u7_1/util/install_modules/inst_schedd_normal.conf
- /usr/local/sge/V60u7_1/util/install_modules/inst_template.conf
- /usr/local/sge/V60u7_1/util/install_modules/inst_update.sh
- /usr/local/sge/V60u7_1/util/logchecker.sh
- /usr/local/sge/V60u7_1/util/qtask
- /usr/local/sge/V60u7_1/util/rctemplates
- /usr/local/sge/V60u7_1/util/rctemplates/darwin_template
- /usr/local/sge/V60u7_1/util/rctemplates/sgebdb_template
- /usr/local/sge/V60u7_1/util/rctemplates/sgeexecd_template
- /usr/local/sge/V60u7_1/util/rctemplates/sgemaster_template
- /usr/local/sge/V60u7_1/util/resources
- /usr/local/sge/V60u7_1/util/resources/calendars
- /usr/local/sge/V60u7_1/util/resources/calendars/day
- /usr/local/sge/V60u7_1/util/resources/calendars/day_s
- /usr/local/sge/V60u7_1/util/resources/calendars/night
- /usr/local/sge/V60u7_1/util/resources/calendars/night_s
- /usr/local/sge/V60u7_1/util/resources/centry
- /usr/local/sge/V60u7_1/util/resources/centry/arch
- /usr/local/sge/V60u7_1/util/resources/centry/calendar
- /usr/local/sge/V60u7_1/util/resources/centry/cpu
- /usr/local/sge/V60u7_1/util/resources/centry/h_core
- /usr/local/sge/V60u7_1/util/resources/centry/h_cpu
- /usr/local/sge/V60u7_1/util/resources/centry/h_data
- /usr/local/sge/V60u7_1/util/resources/centry/h_fsize
- /usr/local/sge/V60u7_1/util/resources/centry/h_rss
- /usr/local/sge/V60u7_1/util/resources/centry/h_rt
- /usr/local/sge/V60u7_1/util/resources/centry/h_stack
- /usr/local/sge/V60u7_1/util/resources/centry/h_vmem
- /usr/local/sge/V60u7_1/util/resources/centry/hostname
- /usr/local/sge/V60u7_1/util/resources/centry/load_avg
- /usr/local/sge/V60u7_1/util/resources/centry/load_long
- /usr/local/sge/V60u7_1/util/resources/centry/load_medium
- /usr/local/sge/V60u7_1/util/resources/centry/load_short
- /usr/local/sge/V60u7_1/util/resources/centry/mem_free
- /usr/local/sge/V60u7_1/util/resources/centry/mem_total
- /usr/local/sge/V60u7_1/util/resources/centry/mem_used
- /usr/local/sge/V60u7_1/util/resources/centry/min_cpu_interval
- /usr/local/sge/V60u7_1/util/resources/centry/np_load_avg
- /usr/local/sge/V60u7_1/util/resources/centry/np_load_long
- /usr/local/sge/V60u7_1/util/resources/centry/np_load_medium
- /usr/local/sge/V60u7_1/util/resources/centry/np_load_short
- /usr/local/sge/V60u7_1/util/resources/centry/num_proc
- /usr/local/sge/V60u7_1/util/resources/centry/qname
- /usr/local/sge/V60u7_1/util/resources/centry/rerun
- /usr/local/sge/V60u7_1/util/resources/centry/s_core
- /usr/local/sge/V60u7_1/util/resources/centry/s_cpu
- /usr/local/sge/V60u7_1/util/resources/centry/s_data
- /usr/local/sge/V60u7_1/util/resources/centry/s_fsize

- /usr/local/sge/V60u7_1/util/resources/centry/s_rss
- /usr/local/sge/V60u7_1/util/resources/centry/s_rt
- /usr/local/sge/V60u7_1/util/resources/centry/s_stack
- /usr/local/sge/V60u7_1/util/resources/centry/s_vmem
- /usr/local/sge/V60u7_1/util/resources/centry/seq_no
- /usr/local/sge/V60u7_1/util/resources/centry/slots
- /usr/local/sge/V60u7_1/util/resources/centry/swap_free
- /usr/local/sge/V60u7_1/util/resources/centry/swap_rate
- /usr/local/sge/V60u7_1/util/resources/centry/swap_rsvd
- /usr/local/sge/V60u7_1/util/resources/centry/swap_total
- /usr/local/sge/V60u7_1/util/resources/centry/swap_used
- /usr/local/sge/V60u7_1/util/resources/centry/tmpdir
- /usr/local/sge/V60u7_1/util/resources/centry/virtual_free
- /usr/local/sge/V60u7_1/util/resources/centry/virtual_total
- /usr/local/sge/V60u7_1/util/resources/centry/virtual_used
- /usr/local/sge/V60u7_1/util/resources/loadsensors
- /usr/local/sge/V60u7_1/util/resources/loadsensors/calendar.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/ibm-loadsensor
- /usr/local/sge/V60u7_1/util/resources/loadsensors/interix-loadsensor.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/load.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/maui.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/nuser.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/sgi_tty_sensor.sh
- /usr/local/sge/V60u7_1/util/resources/loadsensors/solaris-iidle.sh
- /usr/local/sge/V60u7_1/util/resources/pe
- /usr/local/sge/V60u7_1/util/resources/pe/make
- /usr/local/sge/V60u7_1/util/resources/pe/make.sge_pqs_api
- /usr/local/sge/V60u7_1/util/resources/schemas
- /usr/local/sge/V60u7_1/util/resources/schemas/qstat
- /usr/local/sge/V60u7_1/util/resources/schemas/qstat/detailed_job_info.xsd
- /usr/local/sge/V60u7_1/util/resources/schemas/qstat/message.xsd
- /usr/local/sge/V60u7_1/util/resources/schemas/qstat/qstat.xsd
- /usr/local/sge/V60u7_1/util/resources/spooling
- /usr/local/sge/V60u7_1/util/resources/spooling/disable_history.sql
- /usr/local/sge/V60u7_1/util/resources/spooling/history.sh
- /usr/local/sge/V60u7_1/util/resources/spooling/init_postgres.sh
- /usr/local/sge/V60u7_1/util/resources/spooling/init_postgres.sql
- /usr/local/sge/V60u7_1/util/resources/starter_methods
- /usr/local/sge/V60u7_1/util/resources/starter_methods/settaskid.sh
- /usr/local/sge/V60u7_1/util/resources/usersets
- /usr/local/sge/V60u7_1/util/resources/usersets/deadlineusers
- /usr/local/sge/V60u7_1/util/resources/usersets/defaultdepartment
- /usr/local/sge/V60u7_1/util/setfileperm.sh
- /usr/local/sge/V60u7_1/util/sgeCA
- /usr/local/sge/V60u7_1/util/sgeCA/renew_all_certs.csh
- /usr/local/sge/V60u7_1/util/sgeCA/sge_ca
- /usr/local/sge/V60u7_1/util/sgeCA/sge_ssl.cnf
- /usr/local/sge/V60u7_1/util/sgeCA/sge_ssl_template.cnf
- /usr/local/sge/V60u7_1/util/sge_aliases
- /usr/local/sge/V60u7_1/util/sge_log_tee
- /usr/local/sge/V60u7_1/util/sge_request
- /usr/local/sge/V60u7_1/util/sgeremoterun
- /usr/local/sge/V60u7_1/utilbin
- /usr/local/sge/V60u7_1/utilbin/lx26-x86

- /usr/local/sge/V60u7_1/utilbin/lx26-x86/adminrun
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/checkprog
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/checkuser
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/filestat
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/fstype
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/gethostbyaddr
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/gethostbyname
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/gethostname
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/getservbyname
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/infotext
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/loadcheck
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/now
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/qrsh_starter
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/rlogin
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/rsh
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/rshd
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/sge_share_mon
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/spooldefaults
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/spooledit
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/spoolinit
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/testsuidroot
- /usr/local/sge/V60u7_1/utilbin/lx26-x86/uidgid

- sge-V60u7_1-2: Contains the binaries and libraries needed to run sge commands:
  - /usr/local/sge/V60u7_1/bin/lx26-x86
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qacct
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qalter
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qconf
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qdel
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qhold
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qhost
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qlogin
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qmake
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qmod
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qping
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qresub
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qrls
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qrsh
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qselect
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qsh
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qstat
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qsub
  - /usr/local/sge/V60u7_1/bin/lx26-x86/qtcsh
  - /usr/local/sge/V60u7_1/bin/lx26-x86/sgepasswd
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libXltree.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libcrypto.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libcrypto.so.0.9.7a
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libdrmaa.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libspoolb.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libspoolc.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libssl.so
  - /usr/local/sge/V60u7_1/lib/lx26-x86/libssl.so.0.9.7a

There are also other base packages which were built but that will not be discussed in detail since they are out

of the scope of this first approach. If you do not want to implement the features they offer, in principle, they can be neglected from the instalation process:

- sge-ckpt-V60u7_1-2: For checkpointing purposes;
- sge-parallel-V60u7_1-2: For running parallel environments, as OpenMpi, Mpich, etc;
- sge-qmon-V60u7_1-2: The SGE GUI.

The instalation of sge-V60u7_1-2 rpm requires libXm.so.3 and libXp.so.6 provided by openmotif and xorg-x11-deprecated-libs packages, which have already to be installed in the system The sequence of rpm instalation should be:

```
[root@sgetest ~]# rpm -ivh sge-V60u7_1-2.i386.rpm
 Preparing...                ########################################### [100%]
   1:sge                     ########################################### [100%]
[root@sgetest ~]# rpm -ivh sge-utils-V60u7_1-2.i386.rpm
 Preparing...                ########################################### [100%]
   1:sge-utils               ########################################### [100%]
[root@sgetest ~]# rpm -ivh sge-daemons-V60u7_1-2.i386.rpm
 Preparing...                ########################################### [100%]
   1:sge-daemons             ########################################### [100%]
[root@sgetest ~]# rpm -ivh sge-docs-V60u7_1-2.i386.rpm
 Preparing...                ########################################### [100%]
   1:sge-docs                ########################################### [100%]
[root@sgetest ~]# rpm -ivh edg-sge-utils-1.0.0-1.noarch.rpm
 Preparing...                ########################################### [100%]
   1:edg-sge-utils           ########################################### [100%]
[root@sgetest ~]#
```

The SGE rpms will install all files under the directory /usr/local/sge/V60u7_1. Automatically, the rpms will link /usr/local/sge/pro to /usr/local/sge/V60u7_1 and $SGE_ROOT will be later defined as /usr/local/sge/pro. In this way, we can keep old SGE versions and use them when needed. For that, we just have to change the "pro link".

```
[root@sgetest ~]# ll /usr/local/sge
 total 4
 lrwxrwxrwx   1 root root    7 Nov  9 16:40 pro -> V60u7_1
 drwxr-xr-x  11 root root 4096 Nov  9 16:41 V60u7_1
```

Grid WNs instalation

To automatically install SGE GRID WNs using yaim we have created the glite-yaim-sge-3.0.0-1.i386.rpm (glite-yaim-sge-3.0.0-1.i386). This rpm requires that the following packages should already be present in your system...

- Requires: glite-yaim = 3.0.0-34
- Requires: sge = V60u7_1-2 (sge-V60u7_1-2)
- Requires: sge-daemons = V60u7_1-2 sge-daemons-V60u7_1-2)
- Requires: sge-docs = V60u7_1-2 (sge-docs-V60u7_1-2)
- Requires: sge-utils = V60u7_1-2 (sge-utils-V60u7_1-2)
- Requires: edg-sge-utils = 1.0.2-1(edg-sge-utils-1.0.2-1)

This rpm adds the following functions and files to yaim directories:

- **/opt/glite/yaim/scripts/node-info-sge.def**: File which defines a new node, called WN_sge, and the corresponding functions associated to that node. This file is copied to /opt/glite/yaim/scripts/node-info.def after the rpm instalation.

```
[root@sgetest ~]# cat /opt/glite/yaim/scripts/node-info-sge.def | grep WN_sge
```

```
 WN_sge_FUNCTIONS="${WN_FUNCTIONS} config_sge_client"
```

- **/opt/glite/yaim/functions/config_sge_client**: This is the function which configures the SGE WN. It builds a configuration file **/opt/glite/yaim/scripts/execd.conf** used as input to the perl script **opt/glite/yaim/scripts/configure_sgeclient.pm**. Most variables within this configuration file are static since they depend on the SGE instalation paths defined in the rpms excepting 3 ones which have to be defined in the site-info.def file:
  - SGE_QMASTER: Machines which will be working has the SGE QMASTER;
  - DEFAULT_DOMAIN: lip.pt for lip case;
  - ADMIN_MAIL: your email;

```
 [root@sgetest ~]# cat /opt/glite/yaim/functions/config_sge_client
 config_sge_client(){

requires CE_HOST SGE_QMASTER DEFAULT_DOMAIN ADMIN_MAIL

se_host="${SE_LIST%% *}"

cat <<EOF > /opt/glite/yaim/scripts/execd.conf
 SGE_ROOT="/usr/local/sge/pro"
SGE_QMASTER="$SGE_QMASTER"
SGE_QMASTER_PORT="536"
SGE_EXECD_PORT="537"
SGE_CELL_NAME="default"
EXEC_HOST_LIST="$HOSTNAME"
EXECD_SPOOL_DIR_LOCAL="/usr/local/sge/pro/default/spool"
HOSTNAME_RESOLVING="true"
DEFAULT_DOMAIN="$DEFAULT_DOMAIN"
SET_FILE_PERMS="true"
ADMIN_USER="none"
ADMIN_MAIL="$ADMIN_MAIL"
ADD_TO_RC="true"
EOF

ln -s /usr/local/sge/V60u7_1 /usr/local/sge/pro
/opt/glite/yaim/scripts/configure_sgeclient.pm /opt/glite/yaim/scripts/execd.conf
 chmod 0755 /etc/init.d/sgeexecd
 chkconfig --level 2345 sgeexecd on
/etc/init.d/sgeexecd start
 source /usr/local/sge/pro/default/common/settings.sh

cat <<EOF > /etc/ssh/ssh_config
 Host *
Protocol 2,1
   RhostsAuthentication yes
   RhostsRSAAuthentication yes
   RSAAuthentication yes
   PasswordAuthentication yes
   EnableSSHKeysign yes
   HostbasedAuthentication yes
 EOF

cat << EOF >  /opt/edg/etc/edg-sge-knownhosts.conf
 NODES     = $CE_HOST $se_host
 SGEBIN    = /usr/local/sge/pro/bin
 KEYTYPES   = rsa1,rsa,dsa
 KNOWNHOSTS = /etc/ssh/ssh_known_hosts
 EOF

# workaround for duplicate key entries (Savannah bug 5530)
for hostname in $CE_HOST $se_host; do
   if [ -f /etc/ssh/ssh_known_hosts ];then
      grep -v $hostname /etc/ssh/ssh_known_hosts  > /etc/ssh/ssh_known_hosts.tmp
      /usr/bin/ssh-keyscan -t rsa $hostname >> /etc/ssh/ssh_known_hosts.tmp 2>/dev/null
      if [ $? = 0 ]; then
```

```
         mv /etc/ssh/ssh_known_hosts.tmp  /etc/ssh/ssh_known_hosts
       fi
    fi
 done

/opt/edg/sbin/edg-sge-knownhosts
 cron_job edg-sge-knownhosts root "03 1,7,13,19 * * * /opt/edg/sbin/edg-sge-knownhosts"
return 0
}
```

- **/opt/glite/yaim/scripts/configure_sgeclient.pm**: This is the perl script which sets the proper environment, checks the configuration files, configures the sge ports and builds/starts the sge daemons.

After instaling the glite-yaim-sge-3.0.0-1.i386.rpm, you can than run the WN configuration using the following syntax:

```
[root@sgetest ~]# /opt/glite/yaim/scripts/configure_node /root/site-cfg/site-info-egee.def WN_sge
```

For a sucessful configuration, the machine has to be already declared in the SGE qmaster as an **administrative and exec host** and should be already added to a given queue. You should not forget that some SGE related variables will have to be defined in your site-info-egee.def file. For LIP case, they are:

- SGE_QMASTER="sge01.lip.pt"
- DEFAULT_DOMAIN="lip.pt"
- ADMIN_MAIL="goncalo@lip.pt"

If everything goes OK, this should be the output related to config_sge_client function:

```
(...)
Configuring config_sge_client
 ln: `/usr/local/sge/pro/V60u7_1': File exists
/usr/local/sge/pro/default directory does not exist!
Creating /usr/local/sge/pro/default ...
/usr/local/sge/pro/default/common directory does not exist!
Creating /usr/local/sge/pro/default/common ...
/usr/local/sge/pro/default/common/act_qmaster file does not exist!
Creating /usr/local/sge/pro/default/common/act_qmaster ...
/usr/local/sge/pro/default/common/bootstrap file does not exist!
Creating /usr/local/sge/pro/default/common/bootstrap ...
/usr/local/sge/pro/default/common/settings.sh file does not exist!
Creating /usr/local/sge/pro/default/common/settings.sh ...
   starting sge_execd
 Configuration Complete
```

It simply reports the building of files and directories needed by SGE. The sge_execd daemon should be running after the WN configuration...

```
[root@sgetest ~]# ps xua | grep sge_execd
 root     25009  0.0  0.2  6108 1536 ?        S    19:38   0:00 /usr/local/sge/pro/bin/lx26-x86/s
```

and the machine should be recognized as an execution host in SGE QMASTER:

```
[root@sge01 ~]# qhost
 HOSTNAME          ARCH          NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------------------------
sgetest                  lx26-x86        1  0.29  503.1M   50.8M 1019.7M  208.0K
```

NOTES:

- The packages have dependencies with libdb-4.2:
  # rpm -ihv sge-V60u7_1-2.i386.rpm sge-utils-V60u7_1-2.i386.rpm
  sge-daemons-V60u7_1-2.i386.rpm sge-qmon-V60u7_1-2.i386.rpm sge-ckpt-V60u7_1-2.i386.rpm
  sge-parallel-V60u7_1-2.i386.rpm sge-docs-V60u7_1-2.i386.rpm
  error: Failed dependencies:
  libdb-4.2.so is needed by sge-V60u7_1-2
  libdb-4.2.so is needed by sge-utils-V60u7_1-2 install them with the options ' **--force** ' and ' **--nodeps** '.
- In the CE configuration node-info.def is not overwritten by the one that the package provided. Maybe this instruction is missing. In the WN it works.
- After running "configure_node site-info.def CE_sge BDII_Site' the gatekeeper is not launched, because the file lcgsge.pm is not found (only the file lcgsge.conf).
- In order to install perl_XML_Simple we had to download it by using CPAN:
  perl -MCPAN -e 'install XML::Simple'
- According to the document, a previous requisite is to have a basic installation with LCG CE and WN. We think this is not necessary, because the configuration with CE_sge and WN_sge calls to the CE and WN functions.
- Opened a bug (https://savannah.cern.ch/bugs/index.php?23638) related to the obligation of using the parameter TORQUE_SERVER, even when it is not needed. Have a look to the functions config_gip (line 318) and config_gip_scheduler_pluging (line 3).

LIP SGE implementation

DISCLAIMER: PART OF THE WORK DESCRIBED IN THESE PAGES WAS DEVELOPED BY LIP STAFF AND IT IS BASED ON EXISTING WORK IMPLEMENTED BY OTHER INSTITUTES. IT MAY BE NOT FULLY OPTIMIZED OR CORRECT AND THEREFORE, IT SHOULD BE CONSIDERED AS EXPERIMENTAL.

LIP SGE jobmanager

The standard LCG CE middleware is distributed with a standard jobmanager perl script which allows to submit and query job status using PBS batch system. Obviously, if local administrators wish to use a different scheduler, the jobmanager script has to be adapted to it. LIP is testing **SGE** scheduler and the corresponding jobmanager implementation was built by Juan Fontan@CESGA (CESGA SGE jobmanager). CESGA jobmanager should be implemented for **non-shared home directories** and basicaly derives from the standard PBS script.

Introduction

LIP **SGE** lcgsge.pm file (LIP SGE jobmanager) is found under the CE /opt/globus/lib/perl/Globus/GRAM/JobManager directory. Very small changes were made with respect to the original CESGA implementation. We will give a very short explanation of the lcgsge.pm workflow since a more detailed description will be reported by CESGA staff.

The lcgsge.pm workflow

The **lcgsge.pm** script is divided in many different sections and subroutines. It starts with a section defining all relevant inputs which allow a clean **SGE** interaction, such as SGE binaries paths and environment variables:

```
BEGIN
{
        ###
        # [GBorges]
        # Define SGE_ROOT enviroment variables
        $SGE_ROOT        = '/usr/local/sge/V60u7_1';
        $SGE_CELL        = 'default';
        $SGE_RELEASE     = '6.0u7_1';
        $ENV{'SGE_ROOT'} = $SGE_ROOT;
        $ENV{'SGE_CELL'} = $SGE_CELL;

        ###
        # [GBorges]
        # Define SGE commands
 $qsub  = '/usr/local/sge/V60u7_1/bin/lx26-x86/qsub';
 $qstat = '/usr/local/sge/V60u7_1/bin/lx26-x86/qstat';
 $qdel  = '/usr/local/sge/V60u7_1/bin/lx26-x86/qdel';
        $qconf = '/usr/local/sge/V60u7_1/bin/lx26-x86/qconf';

        ###
        # [GBorges]
        # Define supported jobs
        $mpirun = 'no';
        $sge_mpirun = 'no';
        if (($mpirun eq "no") && ($sge_mpirun eq "no"))
        {
            $supported_job_types = "(single|multiple)";
        }
        else
```

```
        {
            $supported_job_types = "(mpi|single|multiple)";
        }

 $cluster = 0;
 $cpu_per_node = 0;
 $remote_shell = '/usr/bin/ssh';
}
```

Afterwards, it defines different subroutines which are called by **Globus package scripts** and allows to interact with **SGE** batch system:

- **subroutine submit**: Checks if the RSL arguments transmitted by **Globus package** are valid and supported. If the RSL arguments are not supported, then the corresponding globus error code should be activated;

```
sub submit
{
 my $self = shift;
 my ($cpu_time,$wall_time);

        my $description = $self->{JobDescription};
        local $arguments     = $description->arguments();
        local $count      = $description->count();
        local $directory      = $description->directory();
        local $email_address     = $description->email_address();
        local $emailonabort     = $description->emailonabort();
        local $emailonexecution     = $description->emailonexecution();
        local $emailonsuspend     = $description->emailonsuspend();
        local $emailontermination   = $description->emailontermination();
        local $executable     = $description->executable();
        local $hostcount            = $description->host_count();
        local $jobtype       = $description->jobtype();
        local $logfile      = $description->logfile();
        local $max_cpu_time     = $description->max_cpu_time();
        local $max_memory     = $description->max_memory();
        local $max_time      = $description->max_time();
        local $max_wall_time     = $description->max_wall_time();
        local $parallel_environment = $description->parallel_environment();
        local $project      = $description->project();
        local $queue                = $description->queue();
        local $stderr      = $description->stderr();
        local $stdin      = $description->stdin();
        local $stdout      = $description->stdout();

        ###
        # [GBorges]
        # Debug file in /tmp/JobManager-lcgsge.log
 chop($date=`date`);
        open FILELOG,'>>/tmp/JobManager-lcgsge.log';
        print FILELOG "ARGUMENTS            = ",$arguments,"\n";
        print FILELOG "COUNT                = ",$count,"\n";
        print FILELOG "DIRECTORY            = ",$directory,"\n";
        print FILELOG "EMAIL ADRESS         = ",$email_address,"\n";
        print FILELOG "EMAIL ON ABORT       = ",$emailonabort,"\n";
        print FILELOG "EMAIL ON EXECUTION   = ",$emailonexecution,"\n";
        print FILELOG "EMAIL ON SUSPEND     = ",$emailonsuspend,"\n";
        print FILELOG "EMAIL ON TERMINATION = ",$emailontermination,"\n";
        print FILELOG "EXECUTABLE           = ",$executable,"\n";
        print FILELOG "HOST COUNT           = ",$hostcount,"\n";
        print FILELOG "JOBTYPE              = ",$jobtype,"\n";
        print FILELOG "LOGFILE              = ",$logfile,"\n";
        print FILELOG "MAX CPU TIME         = ",$max_cpu_time,"\n";
        print FILELOG "MAX MEMORY           = ",$max_memory,"\n";
        print FILELOG "MAX TIME             = ",$max_time,"\n";
```

```
        print FILELOG "MAX_WALL_TIME        = ",$max_wall_time,"\n";
        print FILELOG "PARALLEL ENVIROMENT  = ",$parallel_environment,"\n";
        print FILELOG "PROJECT              = ",$project,"\n";
        print FILELOG "QUEUE                = ",$queue,"\n";
        print FILELOG "STDERR               = ",$stderr,"\n";
        print FILELOG "STDIN                = ",$stdin,"\n";
        print FILELOG "STDOUT               = ",$stdout,"\n";
        close FILELOG;

$self->log("Entering sge submit");

        #####
# check jobtype
        if (defined($jobtype))
        {
           if ($jobtype !~ /^$supported_job_types$/)
           {
              return Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED;
           }
        }

        #####
        # check directory
if ( !defined $directory || $directory eq "")
{
   return Globus::GRAM::Error::RSL_DIRECTORY();
}

        #####
        # check executable
        if ( !defined $executable || $executable eq "")
        {
   return Globus::GRAM::Error::RSL_EXECUTABLE();
        }
        elsif ( !defined $stdin || $stdin eq "")
        {
   return Globus::GRAM::Error::RSL_STDIN;
        }

        #####
        # Determining job max time cpu from job description
        $self->log("Determining job max time cpu from job description");
if (defined($description->max_cpu_time()))
{  $cpu_time = $description->max_cpu_time();
   $self->log("   using maxcputime of $cpu_time");
}
elsif (! $cluster && defined($description->max_time()))
{  $cpu_time = $description->max_time();
   $self->log("   using maxtime of $cpu_time");
}
else
{  $cpu_time = 0;
   $self->log('   using queue default');
}

        #####
        # Determining job max wall time from job description
$self->log("Determining job max wall time limit from job description");
if (defined($description->max_wall_time()))
{  $wall_time = $description->max_wall_time();
   $self->log("   using maxwalltime of $wall_time");
}
elsif ($cluster && defined($description->max_time()))
{  $wall_time = $description->max_time();
   $self->log("   using maxtime of $wall_time");
}
else
```

```
 {  $wall_time = 0;
    $self->log('   using queue default');
 }

 foreach ($description->environment())
 {  if(!ref($_) || scalar(@$_) != 2)
    {  return Globus::GRAM::Error::RSL_ENVIRONMENT();
    }
 }

 foreach($description->arguments())
 {  if(ref($_))
    {  return Globus::GRAM::Error::RSL_ARGUMENTS;
    }
 }

 my $job_id = $self->queue_submit("lcgsge",$cpu_time."|".$wall_time);

 $self->log("Leaving sge submit");

 return
 {  JOB_ID => $job_id,
    JOB_STATE => Globus::GRAM::JobState::PENDING
 } if defined $job_id;

 return Globus::GRAM::Error::NO_RESOURCES;
}
```

* **subroutine poll_batch_system**: Allows to know the status of jobs running in the site cluster parsing the output of the **qstat SGE** command.

```
sub poll_batch_system
{
        ###
        # [GBorges]
        # Debug outputs
        chop($date=`date`);
        open FILELOG,'>>/tmp/JobManager-lcgsge.log';
        print FILELOG "POLL_BATCH_SYSTEM subroutine: Entering on $date \n";

 my $self = shift;
        # @_ is the parameter array for subroutines
 my ($data_ref,$time_ref) = @_;
 my $good_query = 0;
 do
 {  @$data_ref = ();
    $$time_ref = time();
    local(*JQ);
    if (open(JQ,"export SGE_ROOT=/usr/local/sge/V60u7_1 ; $qstat 2>/dev/null |"))
    {   my $jid;
        while(<JQ>)
        {   chomp(my $line = $_);
                ####
                # [GBorges]
                # Try to understand the regular expression
                # "/ /"        = Search $line variable for a given pattern
                # "^"          = Matches the beguining of the target
                # " *"         = Matches the space character zero or more times
                # $1 = "(\d+)" = Matches digit one or more times
                # "\d\.\d+"    = Matches one digit followed by a dot followed by one or more d
                # "[^ ]*"      = Matches anything except the space character zero or more time
                # $2 = "(\w+)" = matches alphanumeric one or more times
                #
     if ($line =~ /^ *(\d+) +\d\.\d+ +[^ ]* +[^ ]* +(\w+)/)
     {    my $st = $2;
```

```
          $jid = $1;
                        ###
                        # GBorges
                        # Stores jobid and its status
          push(@$data_ref,$jid." ".$st);
                        print FILELOG "---> Parsing QSTAT output: Job ",$jid," status is ",$st,"\
     }
        }
        close(JQ);
        $good_query = 1 if $? == 0;
        sleep 30 if !$good_query;
    }
    else
    {   ###
               # [GBorges]
               # Debug outputs
        print FILELOG "---> QSTAT output not available \n";
        $good_query = -1;
    }
 } while($good_query != 1);

        ###
        # [GBorges]
        # Debug outputs
        chop($date=`date`);
        print FILELOG "POLL_BATCH_SYSTEM subroutine: Leaving on $date \n";
        close FILELOG;
}
```

- **subroutine poll**: Allows to link the present status of jobs running in the site cluster with the correct Globus messages which have to generated.

```
sub poll
{
        ###
 # [GBorges]
        # Debug outputs
        chop($date=`date`);
        open FILELOG,'>>/tmp/JobManager-lcgsge.log';
        print FILELOG "POLL subroutine: Entering on $date \n";

 my $self = shift;
 my $description = $self->{JobDescription};
        my $job_id = $description->job_id();
 my $state;
        my $status_line;
        my $job_out = $description->stdout();
        my $job_err = $description->stderr();
        my $internal_id = $description->jobid();
        my ($batch_id,$job_submit_time);

        $self->lookup_or_submit(\$batch_id,\$job_submit_time,\$state);
        $self->log("polling job $batch_id") if defined $batch_id;

        ###
        # [GBorges]
        # The "make_a_poll_query" function belongs to the Helper.pm and will call the "pool_batch
        # jobmanager. A list of jobs identifiers and their status (filled in the "pool_batch_syst
        my @data;
 my $query_ret = $self->make_a_poll_query(".lcgjm","sgequeue.cache",$job_submit_time,\@data);
        print FILELOG "---> QUERY_RET ",$query_ret,"\n";
        print FILELOG "---> STATE ",$state,"\n";

 if ( !defined $state )
 {   my $exit_code = 153;
```

```
   my $status_line;
   foreach my $line (@data)
   {   my $jid;
               print FILELOG "---> LINE ",$line,"\n";
               ###
               # [GBorges]
               # Regular expression
               # "^"          = Matches the beguining of the target
               # $1 = "(\S+)" = Matches no whitespaces one or more times
               # "\s+"        = Matches whitespaces one or more times
               # $2 = "(\S+)" = Matches no whitespaces one or more times
               # "$"          = End of string
if ($line =~ /^(\S+)\s+(\S+)$/)
       {  ($jid,$status_line) = ($1,$2);
                  ###
                  # [GBorges]
                  # Current job id matches one of the few job ids stored in the @data array
   if ($jid eq $batch_id)
   {   print FILELOG "---> Job ",$jid," found in the batch system \n";
       $self->log("  Job found: " . $jid . ' | ' . $status_line );
         $exit_code = 0;
       last;
   }
}
   }
          ###
   # return code 153 = "Unknown Job Id". Verifying that the job is no longer there.
   if ($exit_code == 153)
   {   if ($query_ret)
{   ###
                  # [GBorges]
                  # Debug Outputs
    print FILELOG "---> Job is DONE with exit_code 153 \n";
    $self->log("qstat rc is 153 == Unknown Job ID == DONE");
    $state = Globus::GRAM::JobState::DONE;
}
else
{   ###
                  # [GBorges]
                  # Debug Outputs
    print FILELOG "---> Job is PENDING with exit_code 153 \n";
    $self->log("Job not found, assuming it is PENDING");
    $state = Globus::GRAM::JobState::PENDING;
}
   }
   else
   {   $_ = $status_line;
               ###
               # [GBorges]
               # Check if the last char of variable $_ matches any of the following chars
if(/[EqhwtT]$/)
{   if (/[E]$/)
    {   ###
                      # [GBorges]
                      # Debug Outputs
 print FILELOG "---> Job has FAILED \n";
 $self->cancel();
 $state = Globus::GRAM::JobState::FAILED;
    }
    else
    {   ###
                      # [GBorges]
                      # Debug Outputs
 print FILELOG "---> Job is PENDING \n";
 $state = Globus::GRAM::JobState::PENDING;
    }
}
```

```
  elsif(/[sS]$/)
  {   ###
                  # [GBorges]
                  # Debug Outputs
      print FILELOG "---> Job is SUSPENDED \n";
      $state = Globus::GRAM::JobState::SUSPENDED
  }
  elsif(/[rR]$/)
  {   ###
             # [GBorges]
                  # Debug Outputs
      print FILELOG "---> Job is RUNNING \n";
      $state = Globus::GRAM::JobState::ACTIVE;
  }
  else
  {  ###
     # This else is reached by an unknown response from SGE. It could be that SGE was temporarily
                  # recover and the submitted job is fine. So, we want the JM to ignore this pol
                  # Returning an empty hash below will tell the JM to ignore the respose.
     $self->log("qstat returned an unknown response.  Telling JM to ignore this poll");
     return {};
  }
     }
 }

        ###
        # [GBorges]
        # Debug Outputs
        chop($date=`date`);
        print FILELOG "POLL subroutine: Leaving on $date \n";
        close FILELOG;

        $self->helper_cache_import($internal_id,$batch_id,$job_submit_time,\$state,"batch.out","b
        return { JOB_STATE     => $state};
}
```

- **subroutine cancel_in_batch_system**: Cancels jobs in the local cluster using fork

```
sub cancel_in_batch_system
{       ###
        # [GBorges]
        # Debug output
        chop($date=`date`);
        open FILELOG,'>>/tmp/JobManager-lcgsge.log';
        print FILELOG "CANCEL_IN_BATCH_SYSTEM subroutine: Entering on $date \n";

 my $self = shift;
 my ($batch_id) = @_;
 my $description = $self->{JobDescription};

        ###
        # [GBorges]
        # Debug output
        print FILELOG "Canceling Job ID $batch_id with $qdel cmd on $date \n";

        $self->log("cancel job $batch_id");
 $self->cd_fork_and_exec_cmd(undef,0,0,$qdel,$batch_id);

        ###
        # [GBorges]
        # Debug output
        chop($date=`date`);
        print FILELOG "CANCEL_IN_BATCH_SYSTEM subroutine: Leaving on $date \n";
        close FILELOG;
```

```
        if($? == 0 || $? == 153)
        {  return { JOB_STATE => Globus::GRAM::JobState::FAILED };
        }
 return Globus::GRAM::Error::JOB_CANCEL_FAILED();
}
```

- **submit_to_batch_system**: Allows to submit jobs to the batch system. From the variables filled with the RSL arguments, this subroutine starts to build the script which will latter be submitted to the local **SGE** batch system. The main difference with respect to CESGA jobmanager script concerns about a "local hack" used to run mpi jobs with mpich2 using **SGE** parallel environments.

```
sub submit_to_batch_system
{       ###
 # [GBorges]
        # Debug Output
        chop($date=`date`);
        open FILELOG,'>>/tmp/JobManager-lcgsge.log';
        print FILELOG "SUBMIT_TO_BATCH_SYSTEM subroutine: Entering on $date \n";

 my $self = shift;
 my ($submit_arg) = @_;
 my ($cpu_time,$wall_time) = split('\|',$submit_arg);
 my $description = $self->{JobDescription};
 my $tag = $description->cache_tag() || $ENV{GLOBUS_GRAM_JOB_CONTACT};
 my $cache_pgm = "$Globus::Core::Paths::bindir/globus-gass-cache";

        print FILELOG "---> Cpu time = ",$cpu_time,"\n";
        print FILELOG "---> Wall time = ",$wall_time,"\n";

        ###
 # [JFontan]
 # Define real and fake stdout/stderr
 my $real_stdout;
 my $real_stderr;
 my $fake_stdout;
 my $fake_stderr;

 my $script_url = "$tag/sge_job_script";
 $self->cd_fork_and_exec_cmd(undef,0,0,$cache_pgm,"-add","-t",$tag,"-n",$script_url,"file:/dev/nu
 my $sge_job_script_name = `$cache_pgm -query -t $tag $script_url`;

 chomp($sge_job_script_name);
 if($sge_job_script_name eq "")
 {  return "FAILED";
 }

        ###
        # [GBorges]
        # Start to build SGE script: Writting script header
 my $sge_job_script = new IO::File($sge_job_script_name, '>');
 $sge_job_script->print("#!/bin/sh\n");
        $sge_job_script->print("# SGE batch job script built by Globus job manager\n");
        $sge_job_script->print("#\$ -S /bin/sh\n");
        $sge_job_script->print("#\$ -l h_fsize=5G\n");

        ###
        # [GBorges]
        # Define to whom and when send email. I don't find the necessary jdl instructions to impl
 if (defined $description->email_address() && $description->email_address() ne '')
 {  print FILELOG "---> Monitoring job emailing to ",$description->email_address(),"\n";
           $self->log("Monitoring job emailing to " . $description->email_address() );
    $sge_job_script->print("#\$ -M " . $description->email_address() . "\n");
 }
 if(defined $description->emailonabort() && $description->emailonabort() eq 'yes')
 {  print FILELOG "---> Email when job is aborted \n";
```

```
          $self->log("Email when job is aborted");
   $email_when .= 'a';
}
if(defined $description->emailonexecution() && $description->emailonexecution() eq 'yes')
{  print FILELOG "---> Email at the beginning of job \n";
          $self->log("Email at the beginning of job");
   $email_when .= 'b';
}
if(defined $description->emailontermination() && $description->emailontermination() eq 'yes')
{  print FILELOG "---> Email at the end of job \n";
          $self->log("Email at the end of job");
   $email_when .= 'e';
}
       if(defined $description->emailonsuspend() && $description->emailonsuspend() eq 'yes')
       {  print FILELOG "---> Email when job is suspended \n";
          $self->log("Email when job is suspended");
          $email_when .= 's';
       }
       # Matches one of the chars inside square brackets one or more times
if($email_when =~ /[abes]+/ )
{  $sge_job_script->print("#\$ -m $email_when\n");
}
       else
       {  print FILELOG "---> Do not send email(s) \n";
          $self->log("Do not send email(s)");
       }

       ###
       # [GBorges]
       # Submit to correct queue
if (defined $description->queue() && $description->queue() ne '')
{  $sge_job_script->print("#\$ -q ". $description->queue() . "\n");
}

       ###
       # [GBorges]
       # Some important SGE definitions
       # s_cpu   : The per-process CPU time limit in seconds.
       # s_core  : The per-process maximum core file size in bytes.
       # s_data  : The per-process maximum memory limit in bytes.
       # s_vmem  : The same as s_data (if both are set the minimum is used).
       # h_cpu   : The per-job CPU time limit in seconds.
       # h_data  : The per-job maximum memory limit in bytes.
       # h_vmem  : The same as h_data (if both are set the minimum is used).
       # h_fsize : The total number of disk blocks that this job can create.
       # s_rt and h_rt define the "real time" or also called "elapsed" or
       # "wall clock" time having passed since the start of the job. If h_rt is
       # exceeded  by  a job running in the queue, it is aborted via the SIGKILL
       # signal (see kill(1)). If s_rt is exceeded, the job is  first  "warned"
       # via  the  SIGUSR1 signal  (which can be caught by the job) and finally
       # aborted after the notification time defined in the queue  configuration
       # parameter notify (see above) has passed.

       ###
       # [GBorges]
       # Cpu time and Wall time are passed as arguments (array @_) to this subroutine. All value
       # to compute the appropriate SGE format [hh_mm_ss]
if($cpu_time != 0)
{  my $total_cpu_time;
          if($description->jobtype() eq 'multiple')
          {  $total_cpu_time = $cpu_time * $description->count();
          }
          else
          {  $total_cpu_time = $cpu_time;
          }
   ###
          # [GBorges]
```

```
          # Put in proper SGE format
          my $total_cpu_time_m = $total_cpu_time % 60;
          my $total_cpu_time_h = ($total_cpu_time - $total_cpu_time_m)/60;
   # $sge_job_script->print("#\$ -l h_cpu=${total_cpu_time_h}:${total_cpu_time_m}:00\n");
          print FILELOG "---> Max cpu time defined = ",${total_cpu_time_h},":",${total_cpu_time_
}
       else
       {  print FILELOG "---> No max cpu time defined. Use a default value of 10 hours \n";
          $self->log("No max cpu time defined. Use a default value of 10 hours");
          # $sge_job_script->print("#\$ -l h_cpu=10:00:00\n");
}

       if($wall_time != 0)
       {  my $total_wall_time;
          if($description->jobtype() eq 'multiple')
          {  $total_wall_time = $wall_time * $description->count();
          }
          else
          {  $total_wall_time = $wall_time;
          }
          ###
          # [GBorges]
          # Put in proper SGE format
          my $total_wall_time_m = $total_wall_time % 60;
          my $total_wall_time_h = ($total_wall_time - $total_wall_time_m)/60;
          $sge_job_script->print("#\$ -l h_rt=${total_wall_time_h}:${total_wal_time_m}:00\n");
          print FILELOG "---> Max wall time defined = ",${total_wall_time_h},":",${total_wall_ti
       }
       else
       {  print FILELOG "---> No max wall time defined. Use a default value of 20 hours \n";
          $self->log("No max wall time defined. Use a default value of 20 hours");
          # $sge_job_script->print("#\$ -l h_rt=20:00:00\n");
       }

       ###
       # [GBorges]
       # Define max memory to run job
if(defined $description->max_memory() && $description->max_memory() != 0)
{  my $max_memory;
   if($description->jobtype() eq 'multiple')
   {  $max_memory = $description->max_memory() * $description->count;
   }
   else
   {  $max_memory = $description->max_memory();
   }
   # $sge_job_script->print("#\$ -l s_vmem=${max_memory}M\n");
}
else
{  # $sge_job_script->print("#\$ -l h_vmem=512M\n");
}

chomp(my $my_hostname = `hostname -f`);
mkdir '.lcgjm', 0700;
chomp(my $pwd=`pwd`);
chomp(my $cache_export_dir = `mktemp -d $pwd/.lcgjm/globus-cache-export.XXXXXX`);

$self->helper_init_cache_export_url($cache_export_dir);

       ###
# [JFontan]
# Deal with stdout/stderr
# make .out and .err unique names
my $r_number=int(rand(65536));
$real_stdout=$cache_export_dir."/batch.out";
$real_stderr=$cache_export_dir."/batch.err";
$fake_stdout=$pwd."/.out.".$r_number;
$fake_stderr=$pwd."/.err.".$r_number;
```

```
$sge_job_script->print("#\$ -o " . $fake_stdout . "\n");
$sge_job_script->print("#\$ -e " . $fake_stderr . "\n");

       ###
       # GBorges
       # Code is not rerunable
$sge_job_script->print("#\$ -r n\n");

my @tmp_list = split("/",$cache_export_dir);
my $gpg_file = pop(@tmp_list);
$gpg_file .= ".gpg";

       ###
       # [GBorges]
       # It seems there is no use to this
# [JFontan]
# stagein is not implemented in SGE, have to deal with that other way
# $sge_job_script->print("#PBS -W stagein=".$gpg_file."@".$my_hostname.":".$cache_export_dir."/"

       ###
       # GBorges
       # Get user environment variables
       # X509_USER_PROXY, GLOBUS_REMOTE_IO_URL, GLOBUS_TCP_PORT_RANGE,
       # GLOBUS_LOCATION, GLOBUS_GRAM_JOB_CONTACT, GLOBUS_GRAM_MYJOB_CONTACT,
       # SCRATCH_DIRECTORY, HOME, LOGNAME, EDG_WL_JOBID
       my @user_env;
       foreach my $tuple ($description->environment())
       { $tuple->[0] =~ s/"/\\\"/g;
         $tuple->[1] =~ s/"/\\\"/g;
         $self->helper_armour(\$tuple->[0]);
         $self->helper_armour(\$tuple->[1]);
  push(@user_env, $tuple->[0] . "=" . '"' . $tuple->[1] . '"');
       }

       ###
       # [GBorges]
       # This is a very tricky part !!!
       # We pretend to run mpi jobs without using mpirun. These jobs are
       # treated as jobtype="single" but with count() > 1. In this case,
       # we have to initialize the parallel environment and define the number
       # of slots needed to run the job, which in general can be different from
       # the number of cpus.
       my $userlog;
       my $userdir;
if (defined $description->count() && $description->count() > 1 )
{  ###
          # Search for grid user HOME and LOGNAME environment variables
          #
          foreach my $env_line (@user_env)
          { # split string in array
            @env_strings=split(/=/,$env_line);
            if (@env_strings[0] eq "HOME")
            {  $userdir=@env_strings[1];
               # regular expression: replace all occurances of pattern "
               $userdir=~s/"//g;
            }
            if (@env_strings[0] eq "LOGNAME")
            {  $userlog=@env_strings[1];
               # regular expression: replace all occurances of pattern "
               $userlog=~s/"//g;
            }
   }

          ###
          # [GBorges]
   # Check if old .smpd file exists and delete it
          my $smpdfile = "$userdir/.smpd";
```

```
          my $smpdfilesize = -s $smpdfile;
          if ($smpdfilesize >= 0)
             { $self->cd_fork_and_exec_cmd(undef,0,0,"/bin/rm","-f",$smpdfile); }

          ###
          # [GBorges]
          # Create .smpd file in the grid user home directory
          # Change permissions and ownership of the .smpd file
          my $pass_number=int(rand(65536));
          open(SPMD, "> $smpdfile");
   print SPMD "phrase=",$pass_number,"\n";
          close SPMD;
          # change file owner
          ($login,$pass,$uid,$gid)=getpwnam($userlog);
          chown $uid, $gid, $smpdfile;
          # change file permissions
          my $mode = 0600;
          chmod $mode, $smpdfile;

          ###
          # [GBorges]
          # Get the parallel environment parsing though the
          # queue configuration output given by "qconf -sq <queue>"
          my $queue = $description->queue();
          open(QCONF, "$qconf -sq $queue |") ||
               die "Unable to lookup queue parallel environment with qconf: $!";
          foreach (<QCONF>)
   {  # If line contains only whitespace, skip.
             if (/^\s*$/) { next; }
             # parse the key and value from the output
             my ($key, $value) = split;
             if ($key eq "pe_list")
        { chomp $value;
               my $sge_pe = $value;
               # Set parallel environment with the number of slots
               $sge_job_script->print("#\$ -pe $sge_pe " .$description->count(). "\n");
             }
          }
          close QCONF;
       }

my @library_vars=('LD_LIBRARY_PATH');
if($Config::Config{osname} eq 'irix')
{  push(@library_vars,'LD_LIBRARYN32_PATH');
   push(@library_vars,'LD_LIBRARY64_PATH');
}

       ###
# [GBorges]
       # Fill environment variables in a single array
my $rsh_env = "";
my $local_x509 = '-';
my @new_env;

foreach my $tuple ($description->environment())
{  $tuple->[0] =~ s/"/\\\"/g;
   $tuple->[1] =~ s/"/\\\"/g;
   $self->helper_armour(\$tuple->[0]);
   $self->helper_armour(\$tuple->[1]);
   push(@new_env, $tuple->[0] . "=" . '"' . $tuple->[1] . '"');
   $local_x509 = $tuple->[1] if $tuple->[0] eq 'X509_USER_PROXY';
   $rsh_env .= $tuple->[0] . "=\"" . $tuple->[1] . "\"\n"
          .  "export " . $tuple->[0] . "\n";
}

       ###
       # [GBorges]
```

```
        # Add environment variable needed for mpi jobs
        # MPIEXEC_RSH must be set for mpirun to use ssh
        # PBS_NODEFILE must be set because it is the default machine
        # file used in the RB wrapper
        my $queue = $description->queue();
        my $machinefile = "/tmp/\$JOB_ID.1.$queue/machines";
        push(@new_env,"MPIEXEC_RSH=" . '"' . "ssh_sge_mpich2" . '"');
        push(@new_env,"PBS_NODEFILE=" . '"' . $machinefile . '"');

        ###
 # [JFontan]
 # Add environment directly in the script
 foreach my $env_line (@new_env)
 {   $sge_job_script->print("export " . $env_line . "\n");
 }

 if (defined $description->library_path() && $description->library_path() ne '')
 {  my @library_path;
    if (ref $description->library_path())
    {  foreach my $tuple ($description->library_path())
       {  push(@library_path,@$tuple);
       }
    }
    else
    {  @library_path = ($description->library_path());
    }
    my $library_path = join(":",@library_path);
    foreach my $lib_var (@library_vars)
    { ###
            # [JFontan]
      # Changed EOF by ""
      $sge_job_script->print("
            if test \"X\${$lib_var}\" != \"X\"; then
               $lib_var=\"\${$lib_var}:$library_path\"
            else
               $lib_var=\"$library_path\"
            fi
            export $lib_var
");
      $rsh_env .= "if test \"X\${$lib_var}\" != \"X\"; then\n";
      $rsh_env .= "$lib_var=\"\${$lib_var}:$library_path\"\n";
      $rsh_env .= "else\n";
      $rsh_env .= "$lib_var=\"$library_path\"\n";
      $rsh_env .= "fi\n";
      $rsh_env .= "export $lib_var\n";
    }
 }

        ###
 # [JFontan]
 # Here we can have a problem if the first parameter is "0"
 my $args="";
 my @arguments = $description->arguments();
 if($arguments[0])
 {  foreach my $arg (@arguments)
    {  $self->log("Transforming argument \"$arg\"\n");
       $self->helper_armour(\$arg);
       $self->log("Transformed to \"$arg\"\n");
       $args .= '"' . $arg . '" ';
    }
 }

        ###
 # [JFontan]
 # Link fake stdout/stderr to the real one
 $sge_job_script->print("mkdir -p $cache_export_dir\n");
 $sge_job_script->print("touch $fake_stdout\n");
```

```
 $sge_job_script->print("ln $fake_stdout $real_stdout\n");
 $sge_job_script->print("touch $fake_stderr\n");
 $sge_job_script->print("ln $fake_stderr $real_stderr\n");

 @tmp_list=();
 $self->helper_get_from_tmp_file("scratch",\@tmp_list);
 my $gram_scratch_dir = (scalar(@tmp_list)>0) ? $tmp_list[0] : '-';

        ###
 # [JFontan]
 # Changed method to send the proxy from GPG to SELF (embeded in the script)
 if(($description->jobtype() eq 'multiple' && !$cluster) || $description->jobtype() eq 'single')
 {
#        $self->helper_write_non_mpi_script($sge_job_script,$my_hostname,$local_x509,
#                                           "X509GPG",$cache_export_dir,$gram_scratch_dir,"NONE",
  $self->helper_write_non_mpi_script($sge_job_script,$my_hostname,$local_x509,
        "X509SELF",$cache_export_dir,$gram_scratch_dir,"NONE",$args);
 }
 elsif($description->jobtype() eq 'multiple')
 {
#        $self->helper_write_fake_mpi_script($sge_job_script,$rsh_env,$my_hostname,$local_x509,
#                                           "X509GPG",$cache_export_dir,$gram_scratch_dir,"NONE",
  $self->helper_write_fake_mpi_script($sge_job_script,$rsh_env,$my_hostname,$local_x509,
        "X509SELF",$cache_export_dir,$gram_scratch_dir,"NONE",$args);

  my $count = $description->count;

  # [JFontan]
  # Changed EOF by ""
  # [TODO]
  $sge_job_script->print("
 hosts=\`cat \$PBS_NODEFILE\`;
 counter=0
 while test \$counter -lt $count; do
     for host in \$hosts; do
  if test \$counter -lt $count; then
                   rsh_bootstrap=\`$remote_shell \$host mktemp /tmp/rsh_bootstrap.XXXXXX\`
                   $remote_shell \$host \"cat > \$rsh_bootstrap; chmod 700 \$rsh_bootstrap;\
      counter=\`expr \$counter + 1\`
  else
      break
  fi
     done
 done
 wait
        rm \$local_bootstrap
");
 }

        ###
 # [JFontan]
 # Delete cache dir and stdout/stderr
 $sge_job_script->print("rm -rf ".$cache_export_dir."\n");
 $sge_job_script->print("rm -f ".$fake_stdout."\n");
 $sge_job_script->print("rm -f ".$fake_stderr."\n");

        ###
        # [GBorges]
        # Delete .smpd
        my $smpdfile = "$userdir/.smpd";
        $sge_job_script->print("rm -rf ". $smpdfile ."\n");

        ###
        # Close SGE script
 $sge_job_script->close();

 my $errfile;
```

```
if ($description->logfile() ne "")
{  $errfile = "2>>" . $description->logfile();
}

$self->helper_cache_export();
$self->helper_remove_tmp_file();

return "FAILED" if ! -e $local_x509;

symlink $local_x509,$cache_export_dir."/.emergency-x509";
symlink $sge_job_script_name,$cache_export_dir."/.submit-script";
$self->cd_fork_and_exec_cmd(undef,0,0,"/bin/cp",$local_x509,$cache_export_dir."/.emergency-x509.

       ## $MYLDPATH    = '/usr/local/sge/V60u7_1/lib/lx26-x86:/opt/globus/lib:/opt/edg/lib:/usr/
       ## $MYPATH      = '/opt/glite/bin:/opt/glite/externals/bin:/opt/globus/bin:/usr/java/j2sd

       ## $ENV{'LD_LIBRARY_PATH'} = $MYLDPATH;
       ## $ENV{'PATH'} = $MYPATH;

       foreach $key (sort keys(%ENV)) {
          print FILELOG "$key = $ENV{$key}","\n";
       }

       ###
       # Job submission
my $batch_id;
my $submit_status;
my $attempts=0;
       my $tmp_file;
do
{ ###
  # [JFontan]
  # DEBUG: copy script to /tmp
  system("cp -f $sge_job_script_name /tmp/sge_job_script");
         chomp($tmp_batch_id=`$qsub < $sge_job_script_name`);
  $submit_status = $?;
         print FILELOG "COMMAND =",$qsub,"\n";
         print FILELOG "SCRIPT =",$sge_job_script_name,"\n";
         print FILELOG "TMP_BATCH_ID =",$tmp_batch_id,"\n";
         print FILELOG "SUBMIT_STATUS =",$submit_status,"\n";
  if ($tmp_batch_id =~ /^Your job (\d*)/) {
      $batch_id = $1;
  }
  $self->log("Batch ID: " . $batch_id);
  if ($submit_status != 0)
  {  $batch_id = undef;
      sleep 5;
  }
  $attempts++;
} while ($submit_status != 0 && $attempts<6);

if (!defined $batch_id)
{  $batch_id = "FAILED";
}

       ###
       # [GBorges]
       # Debug Output
       chop($date=`date`);
       print FILELOG "---> Job has been submitted with ID ",$batch_id,"\n";
       print FILELOG "SUBMIT_TO_BATCH_SYSTEM subroutine: Leaving on $date \n";
       close FILELOG;

return $batch_id;
}
```

LIP SGE infoprovider

The standard LCG CE middleware is distributed with a set of perl and python scripts which generate information regarding the actual state of resources inside a given cluster or site. This section describes the main changes implemented to these scripts, called hereafter **infoprovider scripts**, in order to adapt them to use **Sun Grid Engine (SGE)** scheduler instead of the standard PBS scheduler. All changes were performed in such a way that the final output remains the same as the one produced in the PBS case.

Introduction

The information system supported by the LCG CE middleware is based on standard information stored on static files. Such static files are found under **/opt/lcg/var/gip/ldif** and represent the default inputs to the site information system. However, they do not represent the current working status of the cluster or site and have to be updated by dynamic scripts. Such dynamic scripts are found under **/opt/lcg/var/gip/plugin** and must run whenever the information system needs to provide a correct and up to date output.

```
[root@ce02 gip]# cd /opt/lcg/var/gip

[root@ce02 gip]# cd ldif/
[root@ce02 ldif]# ll
 total 56
-rw-r--r--  1 root root 20118 Aug 11 16:23 static-file-CE.ldif
-rw-r--r--  1 root root  7736 Aug 11 16:23 static-file-CESEBind.ldif
-rw-r--r--  1 root root  4453 Aug 11 16:23 static-file-Cluster.ldif
-rw-r--r--  1 root root   705 Aug 11 16:23 static-file-Site.ldif

[root@ce02 gip]# cd ../plugin/
[root@ce02 plugin]# ll
 total 16
-rwxr-xr-x  1 root root 102 Aug 21 14:49 lcg-info-dynamic-ce
-rwxr-xr-x  1 root root 106 Jul  7 13:46 lcg-info-dynamic-scheduler-wrapper
-rwxr-xr-x  1 root root 100 Aug 11 16:24 lcg-info-dynamic-software-wrapper
```

The lcg-info-dynamic-ce wrapper and the lcg-info-dynamic-sge script

The lcg-info-dynamic-ce script is one of the most important dynamical wrapper scripts and runs the following commands:

```
[root@ce02 gip]# cat /opt/lcg/var/gip/plugin/lcg-info-dynamic-ce
#!/bin/sh
/opt/lcg/libexec/lcg-info-dynamic-sge /opt/lcg/var/gip/ldif/static-file-CE.ldif ce02.lip.pt
```

/opt/lcg/libexec/lcg-info-dynamic-sge is a perl script which substitutes the standard **/opt/lcg/libexec/lcg-info-dynamic-pbs** script. It is based in a previous version of the SGE infoprovider script developed at London e-Science Center (LeSC SGE official distribution) and it can be downloaded in LIP lcg-info-dynamic-sge perl script. The basic structure of this script is the following:

- Starts by defining a set of site dependent constants: cluster queue array (**@clusterqueues**), SGE environment variables (**$SGE_ROOT** and **$SGE_CELL**) and path to important SGE binary comands (**qstat** and **qconf**);

```
#####
# Name of cluster queue to query
 my (@clusterqueues)  = ("atlasgrid","cmsgrid","dteamgrid","edteamgrid","eelagrid","opsgrid","swe
```

```
#####
# Set path to root of SGE tools installation.
$ENV{"SGE_ROOT"} = "/usr/local/sge/V60u7_1";

#####
# Set name of SGE cell to use.
$ENV{"SGE_CELL"} = "default";

#####
# Path to qstat and qconf commands.
my ($qstat)   = "/usr/local/sge/V60u7_1/bin/lx26-x86/qstat";
my ($qconf)   = "/usr/local/sge/V60u7_1/bin/lx26-x86/qconf";
```

- Defines the scripts subroutines: **processCommandLine, parseConfig, lookupDNs, lookupSGEVersion, lookupPolicy, lookupQueueState, calculateResponseEstimate, sgeToGlue**;
- Finishes with the script "Main Program" where all subroutines are called and the final output is produced.

```
########################
# --- Main program --- #
########################

#####
# Lookup the path to the static file and the hostname we should use from the commandline.
my ($configpath)  = processCommandLine();

#####
# Lookup the path to statically defined LDIF data file and check if we are able to open it
 my ($ldifpath)   = parseConfig($configpath);

#####
# Obtain a list of the DNs defined in the LDIF data file and store them iin array @dn. It will be
 my (@dn)   = lookupDNs($ldifpath);

#####
# Build up our hash datastructure.
my (%data);

#####
# Set LRMS information.
$data{"GlueCEInfoLRMSType"}  = "sge";
$data{"GlueCEInfoLRMSVersion"}  = lookupSGEVersion();

#####
# Collected relevant information and produce output it in the expected form. This will be done th

my ($clusterqueue);
foreach $clusterqueue (@clusterqueues) {

        #####
        # Define temporary "LocalQueue" key in the data hash
        $data{"GlueCELocalQueue"}  = $clusterqueue;

        #####
        # Lookup queue policy information.
        lookupPolicy(\%data);

        #####
        # Lookup queue state information.
        lookupQueueState(\%data);

        #####
        # Calculate queue response time estimates.
        calculateResponseEstimate(\%data);
```

```
        #####
        # Delete the Local Queue key from the data hash
        delete $data{"GlueCELocalQueue"};

        #####
        # Simply regurgitate all of the collected state information for each of the defined queue
        my ($dn);
        foreach $dn (@dn) {
                # If the string $dn contains the substring $clusterqueue, returns true
                if ($dn =~ m/lcgsge-$clusterqueue/) {
                print $dn;
                }
        }
        foreach (sort (keys %data)) {
                print "$_" . ": " . $data{$_} . "\n";
 }
        print "\n";
}
```

The lcg-info-dynamic-sge script workflow

The core of the lcg-info-dynamic-sge perl script is defined by its "main program" which runs in a sequencial basis:

- It starts by calling **processCommandLine** and **parseConfig** subroutines. These subroutines check the arguments format received by the perl script and if the static file (1st argument) exists and can be opened.

```
#####
# Process commandline. The only valid configurations are with exactly 1 or 2 arguments:
# The first argument is always the path to our configuration file.
# The second argument will be used as the FQDN SGE control host.
sub processCommandLine() {
 if (scalar(@ARGV) == 0) {
  print STDERR "Usage: $0 <config file>\n";
  exit 1;
 } else {
  my ($configfile) = @ARGV;
  return ($configfile);
 }
}

#####
# Given the path to the local static configuration file check if we can open the file
 sub parseConfig($) {
 my ($configpath) = shift; # Path to the configuration file.

 # Open the file.
 open(CONFIG, $configpath) ||
  die "Unable to open configuration file $configpath: $!";
        close(CONFIG);
        my ($ldif_file) = $configpath;
        return $ldif_file;
}
```

- It parses through the static configuration file and searches for lines beguinning by **dn:GlueCEUniqueIDs**. This is done by subroutine **lookupDNs**. The different dns values are then stored in array **@dn** which will be later used to produce part of the final output.

```
#####
# Given the path to the local static LDIF information file, extract a list of DNs as defined in t
# The each DN will include the 'dn: GlueCEUniqueID=' prefix.
```

```
sub lookupDNs($) {
 my ($ldifpath) = shift;  # Path to the LDIF file.
 my (@DN);   # List of DNs that we will return.

        ###
        # Open the file.
 open(LDIFDATA, $ldifpath) ||
  die "Unable to open static LDIF information file $ldifpath: $!";

 ###
 # For every line:
 while(<LDIFDATA>) {
  # If the line contains a DN definition
  if (/dn:\s+GlueCEUniqueID=/) {
   push @DN, $_;
  }
 }
 close(LDIFDATA);
 return @DN;
}
```

- Afterwards, a hash datastructure is defined and filled with up to date * "Glue key = value" * information. Since **SGE** is a queue based scheduler, where different queues can have very different definitions, all the most interesting quantities are built quering the batch system for each queue configuration. This is done through a loop running over all queues * (see lcg-info-dynamic-sge "Main Program" definition)* where specific **SGE** commands are executed and the produced output is parsed to isolate the relevant quantities. The first called subroutine is **lookupPolicy** which searches for the **Maximum Wall Clock Time** and **Maximum CPU Time** queue definitions. The subroutine parses the output produced by **qstat -sq < *name of the queue* > SGE** command and searches for the **h_rt** and **h_cpu** key words. The final step is to convert the obtained SGE time values (HH:MM:SS) in the Glue format running the **sgeToGlue** subroutine.

```
#####
# Lookup global MaxWallClockTime and MaxCPUTime. Acquire this by running `qconf -sq` and looking
sub lookupPolicy($) {
        my ($dataref)      = shift; # Reference to $data hash.
        my ($clusterqueue) = $dataref->{"GlueCELocalQueue"};

 open(QCONF, "$qconf -sq $clusterqueue |") ||
  die "Unable to lookup queue policy with qconf: $!";

 foreach (<QCONF>) {
                # If line contains only whitespace, skip.
  if (/^\s*$/) { next; }

                # parse the key and value from the output
  my ($key, $value) = split;

  if ($key eq "h_rt") {
   # We've found the max wallclock time.
   chomp $value;
   $dataref->{"GlueCEPolicyMaxWallClockTime"} = sgeToGlue($value);
   next;
  }
  if ($key eq "h_cpu") {
   # We've found the max CPU time.
   chomp $value;
   $dataref->{"GlueCEPolicyMaxCPUTime"} = sgeToGlue($value);
   next;
  }
 }
 close QCONF;
}
```

```
#####
# Simply converts an SGE value into one that the Glue schema uses. At present, simply:
#  - replaces "INFINITY" with "0", and
#  - replaces AA:BB:CC with ((AA * 60) + BB ).
#    (i. e. converts from hours:mins:seconds to minutes)
sub sgeToGlue($) {
 my ($value) = shift;
 $value =~ s/INFINITY/0/;
 if ($value =~ /^(\d+):(\d+):(\d+)$/) {
  $value = (($1 * 60) + $2);
 }
 return $value;
}
```

- The **lookupQueueState subroutine** is of most importance since it generates information regarding the current status of resources within the cluster. It starts by reporting the current status of the queue parsing the output generated by the **$qstat -f | grep < *name of the queue* > SGE** command. By default, all queues are able to receive and run jobs and therefore, the **GlueCEStateStatus** variable of the hash datastructure is filled with the **Production** string. However, If the queue is locally disabled, the **Draining** status is assigned to the **GlueCEStateStatus**. Other important variables such as the **number of waiting jobs**, **number of running jobs**, **number of maximum running jobs**, **number of maximum total jobs**, **number of free CPUs** or **total number of CPUs** are obtained within the same subroutine using the same previous procedures, either parsing the **$qstat -s p -q < *name of the queue* >** output or the **$qstat -g c** output.

```
sub lookupQueueState($) {
        my ($dataref) = shift;
        my ($clusterqueue) = $dataref->{"GlueCELocalQueue"};

        #####
        # Lookup the system status: "Queueing", "Production", "Closed", "Draining".
        open(QSTAT, "$qstat -f | grep $clusterqueue |")
                || die "Unable to lookup state of the queue $!";
        foreach (<QSTAT>) {
         my ($queuename,$qtype,$used_tot,$load_avg,$arch,$states) = split;
                $dataref->{"GlueCEStateStatus"} = "Production";
                if ($states eq "d")  {$dataref->{"GlueCEStateStatus"} = "Draining";}
        }

        ###
        # Lookup number of waiting jobs.
        open(QSTAT, "$qstat -s p -q $clusterqueue |")
                || die "Unable to lookup number of waiting jobs: $!";
        my ($jobcount) = 0;
        foreach (<QSTAT>) {
                s/^\s+//; # remove whitespaces (at the beguinning) from the $_ variable using the
                s/\s+$//; # remove whitespaces (at the end) from the $_ variable using the substi

                #####
                # If the line begins with the string "job-ID", skip it. It's a column header.
                if (/^job-ID/) { next; }

                #####
                # If the line begins with a number of '-'s, skip it. It's a header / data separat
                if (/^-+/) { next; }

                #####
                # If the line begins with a number, increment the job count. It's a job entry.  (
                if (/^\d+/) { $jobcount++ };
        }
        $dataref->{"GlueCEStateWaitingJobs"} = $jobcount;

        #####
        # Lookup the number of running jobs. If we're running on SGE 6.*, use `qstat -g c` and lo
```

```
        my $version = $dataref->{"GlueCEInfoLRMSVersion"};
        if ($version =~ /^6/) {
                    open(QSTAT, "$qstat -g c |")
                            || die "Unable to run qstat to lookup queue load: $!";

                    foreach (<QSTAT>) {
                         if (/^$clusterqueue/) {
                                my ($name, $load, $used, $avail, $total) = split;
                                $dataref->{"GlueCEStateRunningJobs"} = $used;
                                $dataref->{"GlueCEPolicyMaxTotalJobs"} = $total;
                                $dataref->{"GlueCEPolicyMaxRunningJobs"} = $total;
                         }
                 }
                 close QSTAT;
        }
        else {
                  die "Unsupported LRMS version $version.";
        }

        ###
        # Lookup the number of free CPUs.
        if ($version =~ /^6/) {
               open(QSTAT, "$qstat -g c |")
                        || die "Unable to run qstat to lookup free cpus: $!";

               foreach (<QSTAT>) {
                        if (/^$clusterqueue/) {
                                my ($name, $load, $used, $avail, $total) = split;
                                $dataref->{"GlueCEStateFreeCPUs"} = $avail;
                                $dataref->{"GlueCEInfoTotalCPUs"} = $total;
                        }
               }
        }
        else {
                  die "UnsupportedLRMS version $version.";
        }

        ###
        # Calculate the total number of jobs.
        $dataref->{"GlueCEStateTotalJobs"} =
                $dataref->{"GlueCEStateWaitingJobs"} +
                $dataref->{"GlueCEStateRunningJobs"};
}
```

- Finally, the **expected response time** and **worst response time** for job execution are computed by the **calculateResponseEstimate** subroutine. Nevertheless, the present implemented procedures are not accurate since the **worst response time** is the **total number of jobs** multiplied by the **maximum wall clock time** allowed in the queue and divided by the **total number of CPUs**. The **espected response time** is estimated as half of the **worst response time**.

```
#####
# Compute the expected response time
 sub calculateResponseEstimate($) {
        my ($dataref) = shift;
        my ($defaultworstcase)  = 9 * 24 * 60 * 60; # 9 days.
        my ($worstjobtime) = $dataref->{"GlueCEPolicyMaxWallClockTime"};
        my ($jobcount) = $dataref->{"GlueCEStateTotalJobs"};
        my ($cpucount) = $dataref->{"GlueCEInfoTotalCPUs"};

        if ($worstjobtime == 0) {
                #####
                # There was no maximum wallclock time set. Assume the $defaultworstcase under the
                $worstjobtime = $defaultworstcase;
        }
```

```
        #####
        # The predicted worst-case response time is if every job will run ahead of any new job, a
        if ($cpucount == 0) {
                #####
                # No CPUs are available.  Put in a large wait time.
                $dataref->{"GlueCEStateWorstResponseTime"} = 99999999;
        }
        else {
                #####
                # Algorithm taken from PBS information reporter minus the job-time-already-used c
                $dataref->{"GlueCEStateWorstResponseTime"} = int(($jobcount * $worstjobtime) / $c
 }

        #####
        # The estimated response time is simply half of the worst-case response time.
        $dataref->{"GlueCEStateEstimatedResponseTime"} = int($dataref->{"GlueCEStateWorstResponse
}
```

- Running the lcg-info-dynamic-sge perl script in a standalone mode, this is the kind of output which must be produced:

```
[root@ce02 libexec]# /opt/lcg/libexec/lcg-info-dynamic-sge /opt/lcg/var/gip/ldif/static-file-CE.l

dn: GlueCEUniqueID=ce02.lip.pt:2119/jobmanager-lcgsge-atlasgrid,mds-vo-name=local,o=grid
 GlueCEInfoLRMSType: sge
 GlueCEInfoLRMSVersion: 6.0u7
 GlueCEInfoTotalCPUs: 2
 GlueCEPolicyMaxCPUTime: 4320
 GlueCEPolicyMaxRunningJobs: 2
 GlueCEPolicyMaxTotalJobs: 2
 GlueCEPolicyMaxWallClockTime: 8640
 GlueCEStateEstimatedResponseTime: 15120
 GlueCEStateFreeCPUs: 0
 GlueCEStateRunningJobs: 0
 GlueCEStateStatus: Draining
 GlueCEStateTotalJobs: 7
 GlueCEStateWaitingJobs: 7
 GlueCEStateWorstResponseTime: 30240

dn: GlueCEUniqueID=ce02.lip.pt:2119/jobmanager-lcgsge-cmsgrid,mds-vo-name=local,o=grid
 GlueCEInfoLRMSType: sge
 GlueCEInfoLRMSVersion: 6.0u7
 GlueCEInfoTotalCPUs: 2
 GlueCEPolicyMaxCPUTime: 12960
 GlueCEPolicyMaxRunningJobs: 2
 GlueCEPolicyMaxTotalJobs: 2
 GlueCEPolicyMaxWallClockTime: 25920
 GlueCEStateEstimatedResponseTime: 58320
 GlueCEStateFreeCPUs: 0
 GlueCEStateRunningJobs: 2
 GlueCEStateStatus: Draining
 GlueCEStateTotalJobs: 9
 GlueCEStateWaitingJobs: 7
 GlueCEStateWorstResponseTime: 116640

dn: GlueCEUniqueID=ce02.lip.pt:2119/jobmanager-lcgsge-dteamgrid,mds-vo-name=local,o=grid
 GlueCEInfoLRMSType: sge
 GlueCEInfoLRMSVersion: 6.0u7
 GlueCEInfoTotalCPUs: 1
 GlueCEPolicyMaxCPUTime: 120
 GlueCEPolicyMaxRunningJobs: 1
 GlueCEPolicyMaxTotalJobs: 1
 GlueCEPolicyMaxWallClockTime: 240
 GlueCEStateEstimatedResponseTime: 0
 GlueCEStateFreeCPUs: 1
 GlueCEStateRunningJobs: 0
```

```
 GlueCEStateStatus: Production
 GlueCEStateTotalJobs: 0
 GlueCEStateWaitingJobs: 0
 GlueCEStateWorstResponseTime: 0

(...)
```

The lcg-info-dynamic-scheduler-wrapper and the lcg-info-dynamic-scheduler-sge script

A different section of the output produced by the information system is generated by the **/opt/lcg/var/gip/plugins/lcg-info-dynamic-scheduler-wrapper** script. This file runs the **/opt/lcg/libexec/lcg-info-dynamic-scheduler-sge** python script with the **/opt/lcg/etc/lcg-info-dynamic-scheduler.conf** configuration file as argument.

```
[root@ce02 libexec]# more /opt/lcg/var/gip/plugin/lcg-info-dynamic-scheduler-wrapper
#!/bin/sh
/opt/lcg/libexec/lcg-info-dynamic-scheduler-sge -c /opt/lcg/etc/lcg-info-dynamic-scheduler.conf
```

The configuration file defines some standard options inputs (relevant static file, VO/queue mapping, path to relevant python files, etc ...) which must be given according to the following format:

```
[root@ce02 libexec]# cat /opt/lcg/etc/lcg-info-dynamic-scheduler.conf
[Main]
static_ldif_file: /opt/lcg/var/gip/ldif/static-file-CE.ldif
 vomap :
   atlasgrid:atlas
   cmsgrid:cms
   dteamgrid:dteam
   swetestgrid:swetest
   eelagrid:eela
   edteamgrid:edteam
   opsgrid:ops
   biomedgrid:biomed
   imaingrid:imain
 module_search_path : ../lrms:../ett
[LRMS]
lrms_backend_cmd : /opt/lcg/libexec/lrmsinfo-sge
[Scheduler]
cycle_time : 0
 vo_max_jobs_cmd : /opt/lcg/libexec/vomaxjobs-sge
```

The lcg-info-dynamic-scheduler-sge workflow

The lcg-info-dynamic-scheduler-sge file (LIP lcg-info-dynamic-scheduler-sge script) is basicaly the same as the default lcg-info-dynamic-scheduler script distributed with the LCG CE middleware. It starts by parsing the configuration file options, runs important python scripts and parses their produced output. In the following, we will describe how those python files (**/opt/lcg/libexec/vomaxjobs-sge** and **/opt/lcg/libexec/lrmsinfo-sge**) are adapted to run **SGE** commands and how the desired information output is generated.

The vomaxjobs-sge python script

The **/opt/lcg/libexec/vomaxjobs-sge** substitutes the standard **/opt/lcg/libexec/vomaxjobs-maui** python scrypt distributed with LCG CE middleware. The original **MAUI** file was developed by J. A. Templon and it is now adapted to interact with **SGE** (LIP vomaxjobs-sge script). Its main purpose it to generate a python dictionary with the maximum number of jobs which are presently allowed to run in each queue. After defining the SGE environment variables and the path to the SGE binaries, the script parses the **qstat -g c SGE** command output. Spliting the output in columns, it searches for specific column headers, defines the index

number of each column and obtains the number of free CPUs for each queue.

```
!/usr/bin/python2
# vomaxjobs-sge
#
# Original script by J. A. Templon, NIKHEF/PDP 2005
#
# Last changes by Goncalo Borges <goncalo@lip.pt>
# Laboratorio de Fisica Experimental de Particulas
# Lisboa, Portugal
#
# Generate generic information on maximum job counts per VO/QUEUE.
# ------------------------------------------------------------------------

import sys

###
# Define script usage
 def usage():
    print "Usage: vomaxjobs-sge [-h <schedulerhost>]"

import getopt
 import string
 import os

try:
    opts, args = getopt.getopt(sys.argv[1:], "h:",
                               ["host="])
except getopt.GetoptError:
    print sys.argv[0] + ": error parsing command line: " +            string.join(sys.argv)
    usage()
    sys.exit(2)

schedhost = None
 for o, a in opts:
    if o in ("-h", "--host"):
        schedhost = a

###
# Define SGE_ROOT environment variable and SGE qstat comand
# Run the SGE qstat command and store the output in "out" variable
 os.environ['SGE_ROOT']='/usr/local/sge/V60u7_1'
cmd = '/usr/local/sge/V60u7_1/bin/lx26-x86/qstat -g c'
import commands
(stat, out) = commands.getstatusoutput(cmd)
if stat:
    print sys.argv[0] + ': SGE \'qstat -g c\' command' +            ' exited with nonzero status'
    sys.exit(1)

###
# Split the qstat output command in several lines. Gather information on VOs/Queues
# Fill the pcaps dictionary with the queue name and number of free processors and print it
 lines = out.split('\n')
fieldnames = lines[0].split()                      # Split column legend
 gpos = fieldnames.index('CLUSTER')   # column index number for queue name
 lpos = fieldnames.index('cdsuE')           # column index number for queue process cap
 pcaps = {}
for line in lines[2:]:                                  # Loop trough all lines
    f = line.split()                                       # Array with the line contents
    max_asstring = f[lpos-4]                     # Store the available processors
    pcaps[f[gpos]] = int( max_asstring )
print pcaps
```

Running the **/opt/lcg/libexec/vomaxjobs-sge** in a standalone mode, you should get as output the following python dictionary:

```
[root@ce02 ~]# /opt/lcg/libexec/vomaxjobs-sge
{'atlasgrid': 0, 'biomedgrid': 2, 'opsgrid': 1, 'imaingrid': 2, 'eelagrid': 2, 'cmsgrid': 0, 'swe
```

The lrmsinfo-sge python script

The lrmsinfo-sge python script substitutes the lrmsinfo-pbs one although it is basically the same script (LIP lrmsinfo-sge). The main difference is that the lrmsinfo-sge script imports classes from other python functions, such as

- /opt/lcg/lib/python/sgeServer.py
- /opt/lcg/lib/python/sge_utils.py

which must be changed to support **SGE** commands. If you run the **/opt/lcg/libexec/lrmsinfo-sge** in a standalone mode, you should get as output the following information:

```
[root@ce02 libexec]# /opt/lcg/libexec/lrmsinfo-sge
 nactive      6
 nfree        5
 now          1156522995
 schedCycle   26
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2208.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2214.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2212.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2210.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2158.ce02.lip.pt', 'queue': 'cmsgrid', 'state': '
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2211.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2159.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2179.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2163.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2209.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2187.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2213.ce02.lip.pt', 'queue': '', 'state': 'queued'
{'group': 'cmsgrid', 'name': 'STDIN', 'jobid': '2157.ce02.lip.pt', 'queue': 'cmsgrid', 'state': '
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2201.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2221.ce02.lip.pt', 'queue': '', 'state': 'queued',
{'group': 'atlas', 'name': 'STDIN', 'jobid': '2170.ce02.lip.pt', 'queue': '', 'state': 'queued',
```

The sgeServer.py script

The **SgeServer.py** is a complicated python scrypt where the only relevant code for our purposes is the **LiveServer class** (LIP sgeServer.py python script). This class is changed in such a way that it is possible to built a similar output as the one produced running the default **/opt/lcg/libexec/lrmsinfo-pbs** file. It runs the **qstat SGE** command, and parsing the output, it obtains quantities such as the job id, the job name, the job owner, the job state, the queue where the job is running and the maximum wallclock time defined for that queue.

```
class LiveServer(Server):

    def __init__(self,*arg,**kw):

        Server.__init__(self,*arg,**kw)
        cnow = time.ctime() # this hack is to get around time zone problems

        from sge_utils import sgenodes

        cpucount = 0
        jobcount = 0
        for node in sgenodes():
            if node.isUp():
                cpucount += node.numCpu
                for cpu in range(node.numCpu):
                    jobcount += len(node.jobs[cpu])
```

```
        self.slotsUp   = cpucount
        self.slotsFree = cpucount - jobcount

        nowtuple = time.strptime(cnow,"%c")
        self.__evtime__ = int(time.mktime(nowtuple))

        cmdstr = '/usr/local/sge/V60u7_1/bin/lx26-x86/qstat'
        (stat, out) = commands.getstatusoutput(cmdstr)

        qstatlines = out.split('\n')
        for qstatline in qstatlines[2:]:
            newj = Job()
    qstatlfds = qstatline.split()                              # Divide line fields
            jobids = qstatlfds[0]+'.ce02.lip.pt'               # Job id
            jname  = qstatlfds[2]                # Job name
            user   = qstatlfds[3]                # Job owner
            jstate = qstatlfds[4]                # Job state
            if jstate == 'r':
             queue = string.split(qstatlfds[7],'@')[0]              # Get queue
             cmd   = '/usr/local/sge/V60u7_1/bin/lx26-x86/qconf -sq '+queue+' | grep h_rt'    # G
     (stat, out) = commands.getstatusoutput(cmd)
             hms = out.split()[1]
             t = string.split(hms,':')
             mins = (int(t[1]) + 60*int(t[0]))      # MaxWallTime in m
            else:
  queue = ''
  mins = ''

    newj.set('jobid',jobids)
    newj.set('name',jname)
    newj.set('user',user)
            newj.set('queue',queue)
            newj.set('maxwalltime',mins)

            try:
                thisgroup=pwd.getpwnam(user)[3]
                groupname=grp.getgrgid(thisgroup)[0]
            except:
                thisgroup='unknown'
                groupname='unknown'

            newj.set('group',groupname)
            if jstate == 'qw':
               val = 'queued'
            elif jstate == 'r':
                val = 'running'
            else:
                val = 'unknown'
            newj.set('state',val)

            self.addjob(newj.get('jobid'),newj)
```

The sge_utils.py script

The **SgeServer.py** script imports the **sgenodes** function from the **sge_utils.py** python script (LIP sge_utils.py python script). The **sgenodes** function runs the **qstat SGE** command and parses the output searching and storing information regarding the execution machine name, the number of processors, the operation status and state of the machine.

```
###
# SGE Definitions
 os.environ['SGE_ROOT']='/usr/local/sge/V60u7_1'
SGENODES = "/usr/local/sge/V60u7_1/bin/lx26-x86/qhost"
QSTAT = "/usr/local/sge/V60u7_1/bin/lx26-x86/qstat"
```

```
###
# Define SGE execution nodes and build the same output as standard scripts
 def sgenodes(nodes="", ignoreError=False):
    if not _sgenodesOK:
        raise IOError, "'%s' not found" % SGENODES
    if not nodes:
        nodes = [""] # meaning all nodes
    else:
        nodes = nodes.split(',')

    #####
    # Run the SGE command qhost to know which are the execution nodes
    cmd = '/usr/local/sge/V60u7_1/bin/lx26-x86/qhost'
    import commands
    (stat, out) = commands.getstatusoutput(cmd)
    if stat:
        print sys.argv[0] + ': SGE \'qhost\' command' +                ' exited with nonzero stat
        sys.exit(1)
    qhostlines = out.split('\n')

    #####
    # Run the command qstat to know which are the execution nodes
    cmd = '/usr/local/sge/V60u7_1/bin/lx26-x86/qstat'
    (stat, out) = commands.getstatusoutput(cmd)
    if stat:
        print sys.argv[0] + ': SGE \'qstat\' command' +                ' exited with nonzero stat
        sys.exit(1)
    qstatlines = out.split('\n')

    fieldnames = qstatlines[0].split()
    if fieldnames != [] :                 # Chech if fieldnames is not empty (no running jobs)
     qpos = fieldnames.index('queue')  # output column for queue

    params = ['     state = ','     np = ','     ntype = ','     status = ']
    paragraph = []

    for qhostline in qhostlines[3:]:             # Loop through all the lines of the qhost output
 qhostlfds = qhostline.split()            # Define qhostlinefields
        execnd    = qhostlfds[0]+'.lip.pt'       # Define execution host
        jobs      = ''
 if fieldnames != [] :
            for qstatline in qstatlines[2:]:         # Loop through all the lines of the qstat ou
         qstatlfds = qstatline.split()        # Define the qstatlinefields
                field     = qstatlfds[qpos]          # Get the SGE queue from the qpos column
         n         = field.count(execnd)      # exec host appears n times in the string field
         if n>0 :
            jobid   = qstatlfds[0]
            jobuser = qstatlfds[3]
            jobs    = '     jobs = 0/'+jobid+'.ce02.lip.pt\n'

        execnode = execnd+'\n'
        state    = params[0]+'free\n'
        np       = params[1]+qhostlfds[2]+'\n'
        ntype    = params[2]+'cluster\n'
        status   = params[3]+'opsys=linux,uname=Linux '+qhostlfds[0]+'.lip.pt'
        entry    = execnode+state+np+ntype+jobs+status
        paragraph.append(entry)

    result = []
    for node in nodes:
        for entry in paragraph:
         result.append(SGENode(entry))
    return result
```

The final produced information

After substituting the very different perl and python scripts, you should run the **/opt/lcg/libexec/lcg-info-wrapper**.

```
[root@ce02 python]# more /opt/lcg/libexec/lcg-info-wrapper
#!/bin/sh
 export LANG=C
/opt/lcg/bin/lcg-info-generic /opt/lcg/etc/lcg-info-generic.conf
```

```
[root@ce02 python]# more /opt/lcg/etc/lcg-info-generic.conf
 temp_dir = /opt/lcg/var/gip/tmp
 plugin_dir  = /opt/lcg/var/gip/plugin
 static_dir =  /opt/lcg/var/gip/ldif
 provider_dir = /opt/lcg/var/gip/provider
 dynamic_script = /opt/lcg/libexec/lcg-info-dynamic-sge
 freshness = 20
 cache_ttl = 300
 response = 5
 timeout = 150
```

After running this wrapper script, the updated information becomes available using an **ldapsearch** command:

```
[root@ce02 python]# ldapsearch -x -h ce02.lip.pt -p2135 -b "mds-vo-name=local,o=grid"
```

CESGA SGE Implementation

SGE integration with LCG at CESGA

Download

The LCG job manager and information provider for SGE currently being used at CESGA with LCG 2.7.0 can be downloaded from: http://www.egee.cesga.es/lcgsge/releases/

You can get an idea of to install it in the following sections taken from the *README* files of the job manager and information provider.

SGE LCG Job Manager

It is based in the **PBS LCG job manager**. At the beginning some variables are defined for an easier development and reading of the code:

```
# This script is based on jobmanager-lcgpbs provided by LCG. It is modified to
# work with SGE by Javier Fontan (jfontan_AT_gmail.com) from CESGA (www.cesga.es)

# Modify by Pablo Rey (prey@cesga.es) to include a Configuration file

# Comments, support and bug reports should be sent to egee-admin_AT_cesga.es

# SGE queues should have an epilog script to delete STDOUT and STDERR files

package Globus::GRAM::JobManager::lcgsge;

@Globus::GRAM::JobManager::lcgsge::ISA = qw(Globus::GRAM::JobManager::Helper);

use Globus::GRAM::Helper;
use Globus::GRAM::Error;
use Globus::GRAM::JobState;
use Globus::GRAM::JobManager;
use Globus::Core::Paths;

use IO::File;
use Config;
use POSIX;

my ($sge_base_path, $sge_bin_path, $mpirun, $qsub, $qstat, $qdel, $qmsg, $cluster, $cpu_per_node,

BEGIN
{
 # [prey]
 # Including the config file
        my $confile;
        if (defined $ENV{'GLOBUS_LOCATION'}) {
                $confile = $ENV{'GLOBUS_LOCATION'} . "/lib/perl/Globus/GRAM/JobManager/lcgsge.con
        } else {
                $confile = "/opt/globus/lib/perl/Globus/GRAM/JobManager/lcgsge.conf";
        }
 do $confile;

 # [jfontan]
 # Change this to sge tools
 #$sge_base_path = '/opt/sge6';
 #$sge_bin_path = $sge_base_path.'/bin/lx24-x86';

 # [prey]
 # Getting the $sge_base_path and $sge_bin_path from the config file
```

```
 $sge_base_path = $SGE_BASE_PATH;
 # Removing the last / if it is included
 chop($sge_base_path) if ($sge_base_path =~ /\/$/);
 $sge_bin_path = $SGE_BIN_PATH;
 # Removing the last / if it is included
 chop($sge_bin_path) if ($sge_bin_path =~ /\/$/);


 $mpirun = $sge_bin_path.'/mpirun';
 $qsub =   $sge_bin_path.'/qsub';
 $qstat =  $sge_bin_path.'/qstat';
 $qdel = $sge_bin_path.'/qdel';
 $qmsg = $sge_bin_path.'/qmsg';
 $cluster = 0;
 $cpu_per_node = 0;
 $remote_shell = '/usr/bin/ssh';
}

sub new
{
 my $proto = shift;
 my $class = ref($proto) || $proto;
 my $self = $class->SUPER::new(@_);

 bless $self, $class;
 return $self;
}
```

* **subroutine submit**: it checks the RSL arguments. A globus error is obtained in case the arguments are not valid or there are no resources; otherwise a job_id is assigned.

```
sub submit
{
 my $self = shift;
 my $description = $self->{JobDescription};
 my ($cpu_time,$wall_time);

 $self->log("Entering sge submit");

 # check jobtype
 if(defined($description->jobtype()))
 {
  # if($description->jobtype !~ /^(mpi|single|multiple)$/)
  if($description->jobtype !~ /^(single|multiple)$/)
  {
   return Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED;
  }
 }
 else
 {
  return Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED;
 }

 if( !defined $description->directory() || $description->directory() eq "")
 {
  return Globus::GRAM::Error::RSL_DIRECTORY();
 }

 if( !defined $description->executable() || $description->executable() eq "")
 {
  return Globus::GRAM::Error::RSL_EXECUTABLE();
 }
 elsif( !defined $description->stdin() || $description->stdin() eq "")
 {
  return Globus::GRAM::Error::RSL_STDIN;
```

```
 }

 $self->log("Determining job max time cpu from job description");
 if(defined($description->max_cpu_time()))
 {
  $cpu_time = $description->max_cpu_time();
  $self->log("   using maxcputime of $cpu_time");
 }
 elsif(! $cluster && defined($description->max_time()))
 {
  $cpu_time = $description->max_time();
  $self->log("   using maxtime of $cpu_time");
 }
 else
 {
  $cpu_time = 0;
  $self->log('   using queue default');
 }

 $self->log("Determining job max wall time limit from job description");
 if(defined($description->max_wall_time()))
 {
  $wall_time = $description->max_wall_time();
  $self->log("   using maxwalltime of $wall_time");
 }
 elsif($cluster && defined($description->max_time()))
 {
  $wall_time = $description->max_time();
  $self->log("   using maxtime of $wall_time");
 }
 else
 {
  $wall_time = 0;
  $self->log('   using queue default');
 }

 foreach ($description->environment())
 {
  if(!ref($_) || scalar(@$_) != 2)
  {
   return Globus::GRAM::Error::RSL_ENVIRONMENT();
  }
 }

 foreach($description->arguments())
 {
  if(ref($_))
  {
   return Globus::GRAM::Error::RSL_ARGUMENTS;
  }
 }

 my $job_id = $self->queue_submit("lcgsge",$cpu_time."|".$wall_time);

 $self->log("Leaving sge submit");

 return
 {
  JOB_ID => $job_id,
  JOB_STATE => Globus::GRAM::JobState::PENDING
 } if defined $job_id;

 return Globus::GRAM::Error::NO_RESOURCES;
}
```

* **subroutine poll**: It links the present status of jobs running in the site cluster with the Globus appropriate message.

```
sub poll
{
 my $self = shift;
 my $description = $self->{JobDescription};
 my $state;

 my $internal_id = $description->jobid();
 my ($batch_id,$job_submit_time);

 $self->lookup_or_submit(\$batch_id,\$job_submit_time,\$state);
 $self->log("polling job $batch_id") if defined $batch_id;

 my @data;
 my $query_ret = $self->make_a_poll_query(".lcgjm","sgequeue.cache",$job_submit_time,\@data);

 if ( !defined $state )
 {
  my $exit_code = 153;
  my $status_line;

  foreach my $line (@data)
  {
   my $jid;

   if ($line =~ /^(\S+)\s+(\S+)$/)
   {
    ($jid,$status_line) = ($1,$2);
    if ($jid eq $batch_id)
    {
     $self->log("  Job found: " . $jid . ' | ' . $status_line );
     $exit_code = 0;
     last;
    }
   }
  }

  # return code 153 = "Unknown Job Id".
  # verifying that the job is no longer there.
  if ($exit_code == 153)
  {
   if ($query_ret)
   {
    $self->log("qstat rc is 153 == Unknown Job ID == DONE");
    $state = Globus::GRAM::JobState::DONE;
   }
   else
   {
    $self->log("Job not found, assuming it is PENDING");
    $state = Globus::GRAM::JobState::PENDING;
   }
  }
  else
  {

   # Get 3rd field (after = )
   $_ = $status_line;

   if(/E|q|w|t/)
   {
    if (/E/)
    {
     $self->cancel();
     $state = Globus::GRAM::JobState::FAILED;
```

```
    }
    else
    {
     $state = Globus::GRAM::JobState::PENDING;
    }
   }
   elsif(/s/)
   {
    $state = Globus::GRAM::JobState::SUSPENDED
   }
   elsif(/r/)
   {
    $state = Globus::GRAM::JobState::ACTIVE;
   }
   else
   {
    # This else is reached by an unknown response from pbs.
    # It could be that PBS was temporarily unavailable, but that it
    # can recover and the submitted job is fine.
    # So, we want the JM to ignore this poll and keep the same state
    # as the previous state.  Returning an empty hash below will tell
    # the JM to ignore the respose.
    $self->log("qstat returned an unknown response.  Telling JM to ignore this poll");
    return {};
   }
  }
 }

 #
 # Not using pbsmsg for now.
 #
 #    if ($state == Globus::GRAM::JobState::ACTIVE)
 #    {
 #        $self->l_send_pbsmsg($batch_id);
 #    }

 #
 # Dont append the batch.err, as we're using PBSMSG
 # Not using PBSMSG for now, so allow appending of batch output
 #

 $self->helper_cache_import($internal_id,$batch_id,$job_submit_time,\$state,"batch.out","batch.er

 return {
  JOB_STATE     => $state
 };
}
```

* **subroutine poll_batch_system**: Allows to know the status of jobs running in the site cluster parsing the output of the qstat SGE command.

```
sub poll_batch_system
{
 my $self = shift;
 my ($data_ref,$time_ref) = @_;

 my $good_query = 0;
 do
 {
  @$data_ref = ();
  $$time_ref = time();
  local(*JQ);
  if (open(JQ,"export SGE_ROOT=".$sge_base_path." ; $qstat 2>/dev/null |"))
  {
   my $jid;
```

```
   while(<JQ>)
   {
    chomp(my $line = $_);
    # [jfontan]
    # Changed regexp to match SGE qstat output
    if ($line =~ /^ *(\d+) +\d\.\d+ +[^ ]* +[^ ]* +(\w+)/)
    {
     my $st = $2;
     $jid = $1;
     push(@$data_ref,$jid." ".$st);
    }
   }
   close(JQ);
   $good_query = 1 if $? == 0;
   sleep 30 if !$good_query;
  }
  else
  {
   $good_query = -1;
  }
 } while($good_query != 1);
}
```

* **subroutine cancel_in_batch_system**: It cancels jobs in the local cluster by using fork.

```
sub cancel_in_batch_system
{
 my $self = shift;
 my ($batch_id) = @_;
 my $description = $self->{JobDescription};

 $self->log("cancel job $batch_id");

 $self->cd_fork_and_exec_cmd(undef,0,0,$qdel,$batch_id);

 if($? == 0 || $? == 153)
 {
  return { JOB_STATE => Globus::GRAM::JobState::FAILED }
 }

 return Globus::GRAM::Error::JOB_CANCEL_FAILED();
}
```

* **subroutine submit_to_batch_system**: It submits jobs to the batch system, after building the script to be submitted to the local SGE batch system, by getting the neccesary information from the variables filled with the RSL arguments.

```
sub submit_to_batch_system
{
 my $self = shift;
 my ($submit_arg) = @_;
 my ($cpu_time,$wall_time) = split('\|',$submit_arg);
 my $description = $self->{JobDescription};
 my $tag = $description->cache_tag() || $ENV{GLOBUS_GRAM_JOB_CONTACT};
 my $cache_pgm = "$Globus::Core::Paths::bindir/globus-gass-cache";

 # [jfontan]
 # Real and fake stdout/stderr
 my $real_stdout;
 my $real_stderr;
 my $fake_stdout;
 my $fake_stderr;
```

```
 my $script_url = "$tag/sge_job_script";
 $self->cd_fork_and_exec_cmd(undef,0,0,$cache_pgm,"-add","-t",$tag,"-n",$script_url,"file:/dev/nu
 my $sge_job_script_name = `$cache_pgm -query -t $tag $script_url`;
 chomp($sge_job_script_name);
 if($sge_job_script_name eq "")
 {
  return "FAILED";
 }

 my $sge_job_script = new IO::File($sge_job_script_name, '>');

 # [jfontan]
 # Changed EOF style by "" and added h_fsize (required by cesga queues)
 $sge_job_script->print("
#! /bin/sh
# SGE batch job script built by Globus job manager
#
#\$ -S /bin/sh
");

 # [prey]
 # Adding the extra parameters if it has been set in the Config file to a non-null value.
 if(defined $EXTRA && $EXTRA) {
  $sge_job_script->print("#\$ -l ".$EXTRA."\n");
 }

 if(defined $description->email_address() && $description->email_address() ne '')
 {
  # [jfontan]
  # Mail wont be used with SGE
  #$sge_job_script->print("#PBS -M " . $description->email_address() . "\n");
 }
 if(defined $description->emailonabort() && $description->emailonabort() eq 'yes')
 {
  $email_when .= 'a';
 }
 if(defined $description->emailonexecution() && $description->emailonexecution() eq 'yes')
 {
  $email_when .= 'b';
 }
 if(defined $description->emailontermination() && $description->emailontermination() eq 'yes')
 {
  $email_when .= 'e';
 }
 if($email_when eq '')
 {
  $email_when = 'n';
 }

 # [jfontan]
 # Mail wont be used with SGE
 #$sge_job_script->print("#PBS -m $email_when\n");

 if(defined $description->queue() && $description->queue() ne '')
 {
  $sge_job_script->print("#\$ -q ". $description->queue() . "\n");
 }
 if(defined $description->project() && $description->project() ne '')
 {
  # [jfontan]
  # Project wont be used with SGE
  #$sge_job_script->print("#PBS -A " . $description->project() . "\n");
 }

 if($cpu_time != 0)
 {
  my $total_cpu_time;
```

```
 if($description->jobtype() eq 'multiple')
 {
  $total_cpu_time = $cpu_time * $description->count();
 }
 else
 {
  $total_cpu_time = $cpu_time;
 }
 #$sge_job_script->print("#PBS -l pcput=${cpu_time}:00\n");
 $sge_job_script->print("#\$ -l s_rt=${total_cpu_time}:00\n");
} else {
 # [prey]
 # Adding s_rt if it has been set in the Config file to a non-null value.
 if(defined $S_RT && $S_RT) {
  $sge_job_script->print("#\$ -l s_rt=".$S_RT."\n");
 }
 #$sge_job_script->print("#\$ -l s_rt=10:00:00\n");
}

if($wall_time != 0)
{
 # [jfontan]
 # Walltime wont be used with SGE
 #$sge_job_script->print("#PBS -l walltime=${wall_time}:00\n");
}

if(defined $description->max_memory() && $description->max_memory() != 0)
{
 my $max_memory;

 if($description->jobtype() eq 'multiple')
 {
  $max_memory = $description->max_memory() * $description->count;
 }
 else
 {
  $max_memory = $description->max_memory();
 }
 $sge_job_script->print("#\$ -l s_vmem=${max_memory}M\n");
} else {
               # [prey]
               # Adding s_vmem if it has been set in the Config file to a non-null value.
               if(defined $S_VMEM && $S_VMEM) {
                       $sge_job_script->print("#\$ -l s_vmem=".$S_VMEM."\n");
               }
 #$sge_job_script->print("#\$ -l s_vmem=512M\n");
}

chomp(my $my_hostname = `hostname -f`);
mkdir '.lcgjm', 0700;
chomp(my $pwd=`pwd`);
chomp(my $cache_export_dir = `mktemp -d $pwd/.lcgjm/globus-cache-export.XXXXXX`);

$self->helper_init_cache_export_url($cache_export_dir);

# [jfontan]
# Deal with stdout/stderr. SGE can not run jobs if it can not write to
# stdout/stderr before starting the job, and $cache_export_dir does not
# exist in worker nodes
my $r_number=int(rand(65536));
$real_stdout=$cache_export_dir."/batch.out";
$real_stderr=$cache_export_dir."/batch.err";
$fake_stdout=$pwd."/.out.".$r_number;
$fake_stderr=$pwd."/.err.".$r_number;

# OLD LCGPBS LINES
```

```
#    $sge_job_script->print("#\$ -o " . $cache_export_dir."/batch.out" . "\n");
#    $sge_job_script->print("#\$ -e " . $cache_export_dir."/batch.err" . "\n");

 $sge_job_script->print("#\$ -o " . $fake_stdout . "\n");
 $sge_job_script->print("#\$ -e " . $fake_stderr . "\n");


 $sge_job_script->print("#\$ -r n\n");

 my @tmp_list = split("/",$cache_export_dir);
 my $gpg_file = pop(@tmp_list);
 $gpg_file .= ".gpg";

 # [jfontan]
 # stagein is not implemented in SGE, have to deal with that other way (X509SELF)
 #$sge_job_script->print("#PBS -W stagein=".$gpg_file."@".$my_hostname.":".$cache_export_dir."/".

 if(defined $description->host_count() && $description->host_count() != 0)
 {
  $sge_job_script->print("#\$ -l num_proc=" .
     $description->host_count().
     "\n");
 }
 elsif($cluster && $cpu_per_node != 0)
 {
  $sge_job_script->print("#\$ -l num_proc=" .
     POSIX::ceil($description->count /
     $cpu_per_node).
     "\n");
 }
 else
 {
                # [prey]
                # Adding num_proc if it has been set in the Config file to a non-null value.
                if(defined $NUM_PROC && $NUM_PROC) {
                        $sge_job_script->print("#\$ -l num_proc=".$NUM_PROC."\n");
                }
  #$sge_job_script->print("#\$ -l num_proc=1\n");
 }

 my @library_vars=('LD_LIBRARY_PATH');
 if($Config::Config{osname} eq 'irix')
 {
  push(@library_vars,'LD_LIBRARYN32_PATH');
  push(@library_vars,'LD_LIBRARY64_PATH');
 }

 my $rsh_env = "";
 my $local_x509 = '-';
 my @new_env;

 foreach my $tuple ($description->environment())
 {
  $tuple->[0] =~ s/"/\\\"/g;
  $tuple->[1] =~ s/"/\\\"/g;
  $self->helper_armour(\$tuple->[0]);
  $self->helper_armour(\$tuple->[1]);

  push(@new_env, $tuple->[0] . "=" . '"' . $tuple->[1] . '"');
  $local_x509 = $tuple->[1] if $tuple->[0] eq 'X509_USER_PROXY';

  $rsh_env .= $tuple->[0] . "=\"" . $tuple->[1] . "\"\n"
      .  "export " . $tuple->[0] . "\n";
 }

 # [jfontan]
 # Add environment directly in the script.
```

```
 # Changed the method to exporting the environment variables directly in the
 # script as SGE had some troubles using -v parameter with too much variables

 #$sge_job_script->print("#\$ -v " .  join(',', @new_env) . "\n");

 foreach my $env_line (@new_env)
 {
  $sge_job_script->print("export " . $env_line . "\n");
 }


 if (defined $description->library_path() && $description->library_path() ne '')
 {
  my @library_path;
  if (ref $description->library_path())
  {
   foreach my $tuple ($description->library_path())
   {
    push(@library_path,@$tuple);
   }
  }
  else
  {
   @library_path = ($description->library_path());
  }
  my $library_path = join(":",@library_path);
  foreach my $lib_var (@library_vars)
  {
   # [jfontan]
   # Changed EOF by ""
   $sge_job_script->print("
            if test \"X\${$lib_var}\" != \"X\"; then
                $lib_var=\"\${$lib_var}:$library_path\"
            else
                $lib_var=\"$library_path\"
            fi
            export $lib_var
");
   $rsh_env .= "if test \"X\${$lib_var}\" != \"X\"; then\n";
   $rsh_env .= "$lib_var=\"\${$lib_var}:$library_path\"\n";
   $rsh_env .= "else\n";
   $rsh_env .= "$lib_var=\"$library_path\"\n";
   $rsh_env .= "fi\n";
   $rsh_env .= "export $lib_var\n";
  }
 }

 # [jfontan]
 # Here we can have a problem if the first parameter is "0"
 # [TODO] fix it
 my $args="";
 my @arguments = $description->arguments();
 if($arguments[0])
 {
  foreach my $arg (@arguments)
  {
   $self->log("Transforming argument \"$arg\"\n");
   $self->helper_armour(\$arg);
   $self->log("Transformed to \"$arg\"\n");

   $args .= '"' . $arg . '" ';
  }
 }

#    if($description->jobtype() eq "mpi")
#    {
#        $sge_job_script->print("\n#Change to directory requested by user\n");
```

```
#        $sge_job_script->print('cd ' . $description->directory() . "\n");
#
# $sge_job_script->print("$mpirun -np " . $description->count() . " ");
#
# if($cluster)
# {
#     $sge_job_script->print(" -machinefile \$PBS_NODEFILE ");
# }
# $sge_job_script->print($description->executable()
#                        . " $args < "
#          . $description->stdin() . "\n");
#    }
#    elsif...

 # [jfontan]
 # Link fake stdout/stderr to the real one so the script can send back the results
 $sge_job_script->print("mkdir -p $cache_export_dir\n");
 $sge_job_script->print("touch $fake_stdout\n");
 $sge_job_script->print("ln $fake_stdout $real_stdout\n");
 $sge_job_script->print("touch $fake_stderr\n");
 $sge_job_script->print("ln $fake_stderr $real_stderr\n");

 # [jfontan]
 # Add grid environment. This is needed in our cluster because WN are not
 # installed the standard way. This is not needed for standard LCG WN.
# $sge_job_script->print(". /opt/cesga/lcg/etc/profile.d/grid_env.sh\n");
 # [prey]
 # Adding the grid enviroment if it has been set in the Config file to a non-null value.
 if(defined $GRID_ENV && $GRID_ENV) {
  $sge_job_script->print(". ".$GRID_ENV."\n");
 }

 @tmp_list=();
 $self->helper_get_from_tmp_file("scratch",\@tmp_list);
 my $gram_scratch_dir = (scalar(@tmp_list)>0) ? $tmp_list[0] : '-';

 # [jfontan]
 # Changed method to send the proxy from GPG to SELF (embeded in the script)

 if(($description->jobtype() eq 'multiple' && !$cluster) || $description->jobtype() eq 'single')
 {
#        $self->helper_write_non_mpi_script($sge_job_script,$my_hostname,$local_x509,
#                                           "X509GPG",$cache_export_dir,$gram_scratch_dir,"NONE",
  $self->helper_write_non_mpi_script($sge_job_script,$my_hostname,$local_x509,
      "X509SELF",$cache_export_dir,$gram_scratch_dir,"NONE",$args);
 }
 elsif($description->jobtype() eq 'multiple')
 {
#        $self->helper_write_fake_mpi_script($sge_job_script,$rsh_env,$my_hostname,$local_x509,
#                                           "X509GPG",$cache_export_dir,$gram_scratch_dir,"NONE",
  $self->helper_write_fake_mpi_script($sge_job_script,$rsh_env,$my_hostname,$local_x509,
      "X509SELF",$cache_export_dir,$gram_scratch_dir,"NONE",$args);

  my $count = $description->count;

  # [jfontan]
  # Changed EOF by ""
  # [TODO] This wont work but multiple jobs are not a priority
  $sge_job_script->print("
 hosts=\`cat \$PBS_NODEFILE\`;
 counter=0
 while test \$counter -lt $count; do
     for host in \$hosts; do
  if test \$counter -lt $count; then
                    rsh_bootstrap=\`$remote_shell \$host mktemp /tmp/rsh_bootstrap.XXXXXX\`
                    $remote_shell \$host \"cat > \$rsh_bootstrap; chmod 700 \$rsh_bootstrap;\
      counter=\`expr \$counter + 1\`
```

```
  else
      break
  fi
     done
 done
 wait
        rm \$local_bootstrap
");
 }

 # [jfontan]
 # Delete cache dir and stdout/stderr
 # After the job completes stdout/stderr files are not deleted so $cache_export_dir
 # will remain in the home diretory.
 $sge_job_script->print("rm -rf ".$cache_export_dir."\n");

 # [jfontan]
 # Could not delete this files until the job finishes :(
 # ...this is a job for... epilog!
 # This files are referenced by SGE_STDOUT_PATH and SGE_STDERR_PATH
 #$sge_job_script->print("rm -f ".$fake_stdout."\n");
 #$sge_job_script->print("rm -f ".$fake_stderr."\n");

 $sge_job_script->close();

 my $errfile;
 if($description->logfile() ne "")
 {
  $errfile = "2>>" . $description->logfile();
 }

 $self->helper_cache_export();

 $self->helper_remove_tmp_file();

 return "FAILED" if ! -e $local_x509;

 symlink $local_x509,$cache_export_dir."/.emergency-x509";
 symlink $sge_job_script_name,$cache_export_dir."/.submit-script";
 $self->cd_fork_and_exec_cmd(undef,0,0,"/bin/cp",$local_x509,$cache_export_dir."/.emergency-x509.

 my $batch_id;
 my $submit_status;
 my $attempts=0;
 do
 {
  #chomp($batch_id = `$qsub < $sge_job_script_name 2>&1 > $errfile`);

  # [jfontan]
  # DEBUG: copy script to /tmp
  #system("cp -f $sge_job_script_name /tmp/sge_job_script");
  chomp($tmp_batch_id = `export SGE_ROOT=$sge_base_path ;$qsub < $sge_job_script_name`);
  $submit_status = $?;
  # [jfontan]
  # The way SGE qsub tells the job number is different to PBS
  if($tmp_batch_id =~ /^Your job (\d*)/) {
   $batch_id = $1;
  }

  $self->log("Batch ID: " . $batch_id);

  if ($submit_status != 0)
  {
   $batch_id = undef;
   sleep 5;
  }
  $attempts++;
```

```
 } while($submit_status != 0 && $attempts<6);

 if (!defined $batch_id)
 {
  $batch_id = "FAILED";
 }

 return $batch_id;
}


1;
```

PREREQUISITES

- The **SGE** client tools (*qsub*, *qstat*, etc) must be available in the CE
- The home directories should have the same path in the CE and in the WN
- The Job Manager use the **XML::Simple.pm** perl module. It can be installed using the following command:

```
 perl -MCPAN -e 'install XML::Simple'
```

INSTALLATION

- Copy **JobManager/lcgsge.pm** and **JobManager/lcgsge.conf** to **/opt/globus/lib/perl/Globus/GRAM/JobManager/**
- Edit **lcgsge.conf** to best fit your needs. This is the content of lcgsge.conf:

```
# SGE paths
$SGE_BASE_PATH   = '/opt/sge6';
$SGE_BIN_PATH    = '/opt/sge6/bin/lx24-x86';

# Default values for SGE required complex values
# In case SGE forces to specify a given set of parameters you can
# specify the default values in this section
# If you don't need to define some parameter you should to set it
# to an empty string or to 0. For example:
#$S_VMEM         = '';
#$NUM_PROC       = 0';

# Parameters that could be availables in the jdl file
# These values are only use in case a value is not specified
# in the jdl file
#max cpu time
#$S_RT           = '10:00:00';
$S_RT            = '';
#max memory
#$S_VMEM         = '512M';
$S_VMEM          = '';
#number of processors
#$NUM_PROC       = 1;
$NUM_PROC        = 0;

# Extra parameters not available in the jdl file.
#$EXTRA          = 'h_fsize=5G,s_cpu=0:10:0';
$EXTRA          = '';

# Add grid environment. This is needed if the middleware is not
# installed in the standard location in the WNs.
# This is not needed for standard WN.
#$GRID_ENV       = '/opt/cesga/lcg/etc/profile.d/grid_env.sh';
```

```
$GRID_ENV        = '';
```

- Edit **/etc/globus.conf** and add the following lines:

```
[gatekeeper/lcgsge]
type=lcgsge
```

And restart **globus-gatekeeper**:

```
service globus-gatekeeper restart
```

In this way globus will add automatically the **SGE** jobmanager service to **/opt/globus/etc/grid-services**

- Copy the file **conf/lcgsge.rvf** to **/opt/globus/share/globus_gram_job_manager/lcgsge.rvf**. This file contains the queues where the jobmanager can submit.
- Edit **lcgsge.rvf** and adjust the queue values (*CHANGEME*) to your current **SGE** configuration
- Configure the **SGE** queues to use the epilog file available in **conf/epilog.sh** so that home directories are cleaned after the job finishes

TESTING IT

Check the installation with:

```
globus-job-run &lt;CE_HOST&gt;:2119/jobmanager-lcgsge -q &lt;QUEUE&gt; /bin/hostname
edg-job-submit -r &lt;CE_HOST&gt;:2119/jobmanager-lcgsge-&lt;SGE_QUEUE&gt; job.jdl
```

SUPPORT

For further details about 'SGE LCG Job Manager' check: SGE_integration_with_LCG_at_CESGA

SGE LCG Information Provider

This is the **SGE** LCG information provider currently being used at ***CESGA*** with LCG 2.7.0. It is based in the **SGE** LCG information provider developed by ***David McBride*** from ***Imperial College***.

PREREQUISITES

- The **SGE** client tools (*qsub*, *qstat*, etc) must be available in the CE

INSTALLATION

- Copy **gip/plugin/lcg-info-dynamic-sge** to **/opt/lcg/var/gip/plugin/**
- Copy **gip/ldif/static-file-sge.ldif** to **/opt/lcg/var/gip/ldif/**
- Edit **/opt/lcg/var/gip/ldif/static-file-sge.ldif** and replace <CE_HOST>, <SE_HOST> and <SGE_QUEUE> with the corresponding values for your site
- Copy the directory **gip/sge/** to **/opt/lcg/var/gip/**
- Copy **lcg-info-dynamic-sge** to **/opt/lcg/libexec/**
- Edit **/opt/lcg/libexec/lcg-info-dynamic-sge** to best fit your needs:

- Adjust the paths in the beginning of the file
- Replace each occurrence of "GRID" with the name of the SGE queue that will used to run grid jobs
- In the line `$jobentry->{"JB_owner"} !~ /^cesga\d\d\d$/ ) {` replace *cesga* with the name of the pool accounts you are using to run grid jobs (if you are using several pool accounts you can use *(pool1|pool2|...)*)

- Create an empty **sge-jobmanager.conf** file

```
touch /etc/sge-jobmanager.conf
```

TESTING IT

Check the output of:

```
ldapsearch -x -H ldap://&lt;CE_HOST&gt;:2135 -b mds-vo-name=local,o=grid
```

SUPPORT

In case of problems contact egee-admin@cesgaNOSPAMPLEASE.es

SGE and Apel

Introduction

The Apel software is composed of two components: The *Log Parser* and *Publisher*.

The *Log Parser* interprets log files to extract job information and publishes it using **R-GMA**. Specifically, it processes the **LCG gatekeeper logs**, the **system message logs** and the **batch system (PBS, LSF, SGE, ...) event logs**. Extracted data is then stored within a *MySQL* database. The Apel *Log Parser* also makes *LDAP* queries of the Computing Element to obtain the CPU performance figures for the worker node clusters and sub-clusters.

The *Publisher* is used to generate accounting records derived from the parsed logging data. The accounting records are then published into **R-GMA** where they are then collected by a central accounting server which, aggregates records from all sites.

You can get more information about Apel in the **Apel User Guide** and the **Apel Schema** which can be download from http://goc.grid-support.ac.uk/gridsite/accounting/index.html.

Publisher

The *Publisher* is used to piece together accounting records derived from data parsed from gate keeper, system message and event log files. The generated data is stored locally within a *MySQL* database and is also published into **R-GMA**. The published data will then be stored by **R-GMA** onto the central accounting server.

The *Publisher* is deployed on the same host that runs the *MySQL* server, normally the *MON Box*.

The *Publisher* is independent of the *Log Parser* used. You can get more information about the *Publisher* in the **Apel User Guide**.

Log Parser

The *Log Parser* is used to parse **gate keeper, system message and event logs** produced by a site running a batch processing system. Currently, the *Log Parser* comes in several different flavours: *PBS, LSF, SGE, Condor, ...* The *Log Parser* you choose will depend upon the batch processing system used by the site. However, the deployment and configuration are both equivalent. In this case we will speak about the **SGE** flavour.

The *Log Parser* can be deployed in a variety of ways depending upon the setup of your site. An overview of some of the typical deployments are described in the **Apel User Guide**. We use the Combined GK/CE option. In this setup the *Log Parser* is configured to process both the gate keeper logs and event logs but on the same host.

At this moment the *Log Parser* for **PBS** and **LSF** comes with its own *rpm* package included in the installation of the CE. It requires the *Apel Core* module to be installed (included also in the installation of the CE). The *Log Parser* for **SGE** has to be installed manually. You can get the corresponding rpm from http://goc.grid-support.ac.uk/gridsite/accounting/latestBuild.html.

The *rpm* installs the following components (using the **SGE** *Log Parser* as an example):

- **/opt/glite/bin/apel-sge-log-parser** the script used to run the *Log Parser*.
- **/opt/glite/etc/glite-apel-sge/parser-config.xml** contains an example config file used by the *Log Parser*.

The *Log Parser* should be run daily. To schedule the *Log Parser* to run at 6:25 am every day, setup a cron entry as follows:

```
25 06 * * * root env RGMA_HOME=/opt/glite APEL_HOME=/opt/glite /opt/glite/bin/apel-sge-log-parser
```

The *Log Parser* is setup using a configuration file encoded in an *XML* format. The configuration file is composed of a list of processors. Each processor carries out a unit of functionality. When the *Log Parser* is started, the program will find any processors defined within the config file and will attempt to schedule each one consecutively.

This is the config file used in our site:

```
<?xml version="1.0" encoding="UTF-8"?>

<ApelConfiguration enableDebugLogging="yes">

    <SiteName>CESGA-EGEE</SiteName>

    <DBURL>jdbc:mysql://mon.egee.cesga.es:3306/accounting</DBURL>
    <DBUsername>***********</DBUsername>
    <DBPassword>***********</DBPassword>

    <CPUProcessor>
        <GIIS>mon.egee.cesga.es</GIIS>
    </CPUProcessor>

    <EventLogProcessor>
        <Logs searchSubDirs="yes" reprocess="no">
            <Dir>/var/log/sge</Dir>
        </Logs>
        <Timezone>UTC</Timezone>
    </EventLogProcessor>

    <GKLogProcessor>
        <SubmitHost>ce2.egee.cesga.es</SubmitHost>
```

```
        <Logs searchSubDirs="yes" reprocess="no">
            <GKLogs>
              <Dir>/var/log</Dir>
            </GKLogs>
            <MessageLogs>
              <Dir>/var/log</Dir>
            </MessageLogs>
        </Logs>
    </GKLogProcessor>

</ApelConfiguration>
```

The SGE accounting log file

The **SGE** batch log file has a simple format using ***:*** as a field separator. These are the fields that you can find in the accounting log file:

```
qname:hostname:group:owner:jobname:jobnumber:account:priority:qsub_time:start_time:
end_time:failed:exit_status:ru_wallclock:ru_utime:ru_stime:ru_maxrss:ru_ixrss:ru_ismrss:
ru_idrss:ru_isrss:ru_minflt:ru_majflt:ru_nswap:ru_inblock:ru_oublock:ru_msgsnd:ru_msgrcv:
ru_nsignals:ru_nvcsw:ru_nivcsw:project:department:granted_pe:slots:UNKNOWN:cpu:mem:
UNKNOWN:command_line_arguments:UNKNOWN:UNKNOWN:maxvmem_bytes
```

Below you can find an real example:

```
GRID:compute-1-25.local:cesga:cesga013:STDIN:865666:sge:0:1149045197:1149045325:
1149045327:0:0:2:0:0:0.000000:0:0:0:0:18439:0:0:0.000000:0:0:0:0:2935:240:NONE:
defaultdepartment:NONE:1:0:0.010000:0.000020:0.000000:-U cesgaGRID_solocesga,
cesgaGRID -q GRID -l h_fsize=5G,num_proc=1,s_rt=36000,s_vmem=512M:0.000000:
NONE:10375168.000000
```

The accounting file format is documented in the **SGE** man 5 page "accounting" (http://gridengine.sunsource.net/nonav/source/browse/~checkout~/gridengine/doc/htmlman/htmlman5/accounting.html).

Normally, the **SGE** server (***qmaster***) is located in the CE machine but in our case it is located in other machine. So we have to copy the **SGE** batch log files from this machine to the CE.

In the **SGE** server we have setup a *cron* entry which is in charge of copy daily the **SGE** batch log file from this machine to the CE. This is the script used to do this job:

```
#!/bin/sh

# To take into account the change in month
 TZ=MET+24
 export TZ

DIA=`date "+%d"`
MES=`date "+%m"`
ANHO=`date "+%Y"`

grep ^GRID /home/cesga/casperez/accounting/$ANHO$MES$DIA | ssh  sge@ce2.egee.cesga.es 'cat - >> /
```

This script search the grid jobs in the **SGE** batch log file corresponding to the day before the current date and copy the output to a file in the CE.

We need to configure the access without password from the **SGE** server to the CE using the ***ssh-keygen*** command and the ***authorized_keys*** file.

The Gatekeeper log file

Apel used the the ***JMA records*** listed in the Gatekeeper log file. All the records in the GK log contain the **GATEKEEPER_JM_ID** field. The **GRAM_SCRIPT_JOB_ID** and the **DN** are in different GK log records and this field permits them to be matched up. It serves no other purpose and does not propagate further.

This is an example:

```
JMA 2006/05/31 04:43:25 GATEKEEPER_JM_ID 2006-05-31.04:42:03.0000002096.0000052290 for
/C=ES/O=DATAGRID-ES/O=CESGA/CN=Jose Carlos Mourino Gallego on 193.144.34.236

JMA 2006/05/31 04:43:31 GATEKEEPER_JM_ID 2006-05-31.04:42:03.0000002096.0000052290 has
 GRAM_SCRIPT_JOB_ID 1149043405:lcgsge:internal_1053670411:4710.1149043396 manager type lcgsge
```

The System Message log file

The ***MessageRecords*** table is built from data written to the system messages log files when the job manager is launched and when the batch job has completed.

This is an example:

```
May 31 05:13:18 ce2 gridinfo: [31767-11715] Submitted job 1149043405:lcgsge:internal_1053670411:
4710.1149043396 to batch system lcgsge with ID 865666

May 31 05:33:29 ce2 gridinfo: [31767-31767] Job 1149043405:lcgsge:internal_1053670411:
4710.1149043396 (ID 865666) has finished
```

Example of Accounting Record (LcgRecords)

Using the previous examples of event, gatekeeper and system message logs this the final accounting record which is stored in the ***LcgRecords*** table:

Field	Value
RecordIdentity	2006-05-31 03:15:27 865666 ce2.egee.cesga.es CESGA-EGEE
ExecutingSite	CESGA-EGEE
LocalJobID	865666
LCGJobID	NULL
LocalUserID	cesga013
LCGUserID	/C=ES/O=DATAGRID-ES/O=CESGA/CN=Jose Carlos Mourino Gallego
LCGUserVO	cesga
ElapsedTime	P2S
BaseCpuTime	P0S
ElapsedTimeSeconds	2
BaseCpuTimeSeconds	0
StartTime	2006-05-31T03:15:25Z
StopTime	2006-05-31T03:15:27Z
StartTimeUTC	2006-05-31T03:15:25Z
StopTimeUTC	2006-05-31T03:15:27Z
StartTimeEpoch	1149045325
StopTimeEpoch	1149045327
ExecutingCE	ce2.egee.cesga.es
MemoryReal	NULL
MemoryVirtual	NULL
SpecInt2000	381

SpecFloat2000	0
EventDate	2006-05-31
EventTime	03:15:27
MeasurementDate	2006-06-02
MeasurementTime	05:17:06

Imperial SGE implementation

For all the details see http://www.gridpp.ac.uk/wiki/LCG-on-SGE#Explanation:_The_Information_Reporter

SA3 SGE TESTBED

SGE Testbed and Stress tests

This topic: LCG > ImplementationOfSGE
Topic revision: r33 - 2011-06-21 - AndresAeschlimann