LGI

Leiden Grid Infrastructure

Dr. M. F. Somers
Theoretical Chemistry
Leiden Institute of Chemistry
University of Leiden
m.somers@chem.leidenuniv.nl

http://fwnc7003.leidenuniv.nl/LGI

July 2010.

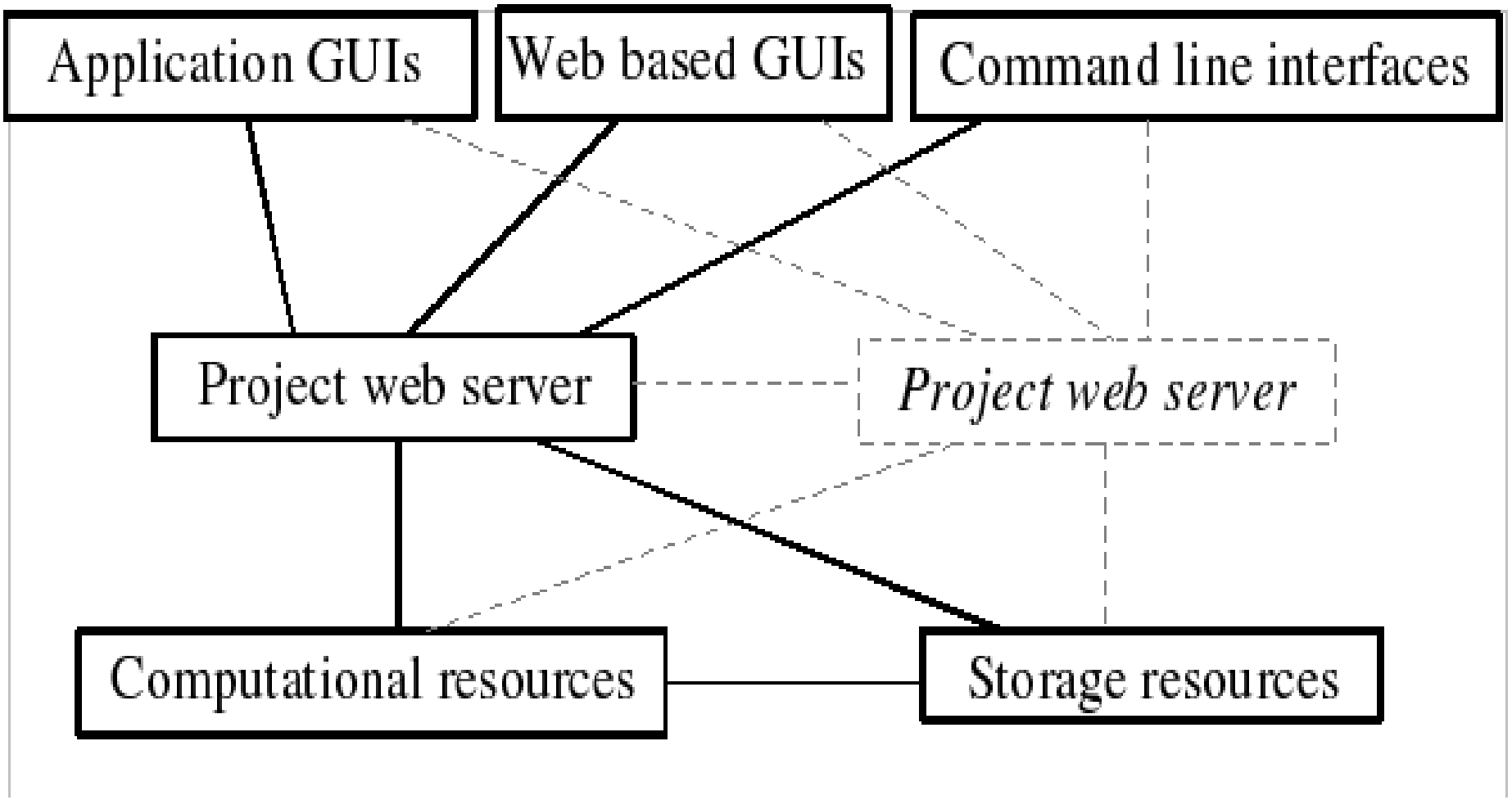
Outline

- Why LGI?
- Design of LGI.
- Current setup & example session.
- Scalability & performance.
- LGI as pilot job framework.
- Possible storage applications.

Why LGI?

- Bundle clusters, supercomputers, grids, BOINC project & workstations behind firewalls, routers & outside our administrative domain.
- Only few (high-performance) applications hard to install (some licensed, binary only or very system specific).
- Easy to setup, administrate, manage & scale.
- Easy interface for users.

Design of LGI



Design of LGI

- Use x509 certificates for XML over https & LAMP stack as project server. NO PROXY's!
- Project is nothing more than a database & single server can run multiple projects.
- User management on project server & on resources: a global LGI user namespace per project & in x509 certs.
- Scheduling & QoS possible on project server. Currently FIFO & FCFS & heartbeat.
- Three general interfaces: web, CLI & Python. Other specific interfaces use API.

Design of LGI

- Each job in DB has 'input' & 'output' blobs with application specific content.
- Each job has repository (URL on a server) to upload & download big files.
- Each job has application specific XML field 'JobSpecifics' for references to repository & possible restart data in other repositories.
- Each job can be scheduled / submitted to specific / 'any' resource that reported back support for application.

Design of LGI

- Daemon in C++ with libcURL & STL runs as non-root. If run as root: automatic sand boxing.
- Daemon caches on disk & several daemons can run concurrently on resource.
- Daemon monitors project server for jobs & 'abort' signals / rescheduling of jobs due to lost heartbeats / errors. Reports capabilities.
- Implement applications through back-end scripts: Torque/PBS, LoadLeveler, BOINC, gLite, forking, SGE ...

Example daemon config

```
<LGI>
        <ca_certificate_file> /home/mark/LGI/certificates/LGI+CA.crt </ca_certificate_file>
        <resource>
                <resource_certificate_file> /home/mark/LGI/certificates/mark@fwnc7003.crt </resource_certificate_file>
                <resource_key_file> /home/mark/LGI/certificates/mark@fwnc7003.key </resource_key_file>
                <run_directory> /home/mark/LGI/daemon/runhere </run_directory>
                <owner_allow> </owner_allow>
                <owner_deny> </owner_deny>
                <job_limit> 20 </job_limit>

                <number_of_projects> 1 </number_of_projects>

                <project number='1'>
                        <project_name> LGI </project_name>
                        <project_master_server> https://fwnc7003.leidenuniv.nl/LGI </project_master_server>

                        <owner_allow> </owner_allow>
                        <owner_deny> </owner_deny>
                        <job_limit> 10 </job_limit>

                        <number_of_applications> 1 </number_of_applications>

                        <application number='1'>
                                <application_name> hello_world </application_name>

                                <owner_allow> <any> 2 </any> </owner_allow>
                                <owner_deny> </owner_deny>
                                <job_limit> 4 </job_limit>
                                <max_output_size> 4096 </max_output_size>
                                <capabilities_file> /proc/cpuinfo </capabilities_file>

                                <check_system_limits_script> /home/mark/LGI/daemon/hello_world_scripts/check_system_limits_script </check_system_limits_script>
                                <job_check_limits_script> /home/mark/LGI/daemon/hello_world_scripts/job_check_limits_script </job_check_limits_script>
                                <job_check_running_script> /home/mark/LGI/daemon/hello_world_scripts/job_check_running_script </job_check_running_script>
                                <job_check_finished_script> /home/mark/LGI/daemon/hello_world_scripts/job_check_finished_script </job_check_finished_script>
                                <job_prologue_script> /home/mark/LGI/daemon/hello_world_scripts/job_prologue_script </job_prologue_script>
                                <job_run_script> /home/mark/LGI/daemon/hello_world_scripts/job_run_script </job_run_script>
                                <job_epilogue_script> /home/mark/LGI/daemon/hello_world_scripts/job_epilogue_script </job_epilogue_script>
                                <job_abort_script> /home/mark/LGI/daemon/hello_world_scripts/job_abort_script </job_abort_script>
                        </application>

                </project>

        </resource>
</LGI>
```

Current setup in Leiden

- Applications: SA-DVR, SPO-DVR, CPMD, Gaussian, MolPro, ADF, VASP, nonmem, Classical & hello_world.

- Resources: Huygens, Lisa, Boeddha, Nocona, Woodcrest, Harpertown, Nehalem, Gainestown, DAS3, gLite/globus through gridui & gliteui & our own BOINC-project.

Example session

Demonstration of CLI & WEB interface:

...students run Gaussian on grid within minutes...

Performance

2 CPU 3.0 GHz Xeon EM64T Nocona with HT, 4 GB DDRII 400MHz RAM, 74 GB 10k rpm HD, 100Mb Ethernet on SL 4.3:

100k jobs in database, 100 concurrent resources using '-ft 5' & 4 KB RSA keys.

load of 30 (84% user, 15% system), MySQL 15% to 50% CPU with 300 MB RAM & 2.6 GB virtual, DB 250 MB on disk, no swapping, <0.01% queries >1s, no queries >5s.

Possible

Should be possible to have >1M jobs & ~10k resources on ~3k euro hardware in a single server setup under normal conditions.

Have not tested this... **yet !**

Scalability

Several project servers as slaves with each a separate MySQL & storage back-end.

or

Load balance a single 'project server' with firewall / DNS round robin to several Apache+PHP servers, run MySQL cluster across servers & use GlusterFS to scale storage for repository.

LGI as pilot job framework

- Use a resource daemon as pilot job manager to schedule pilot jobs on grid.
- Each pilot job on grid is a resource daemon with application binaries & configuration polling a project server.
- If no work for some time pilot jobs stop automatically but some pilot jobs are kept alive for fast response on grid.
- Under development by Willem v. Engen & JanJust Keijser at NIKHEF.
- Testing & reporting fase.

Possible storage applications

1) Resource accepts 'file_storage' job.
2) Resource stores data from repository or other URL part of input any way it likes.
3) Resource creates link / reference to new storage location. Reference could be web interface using x509 & https.
4) Resource finished job with reference as job output.
5) Resource prunes storage / keeps track of storage job by checking job repository existence.

Support

LGI is supported by NWO/NCF & NIKHEF...

Thanks...

http://fwnc7003.leidenuniv.nl/LGI