Integrating various Grid resource managers with GT4

the early experience

Dominik Lupinski
Pawel Plaszczak

www.gridwisetech.com
{pawel,dominik}@gridwisetech.com

gridwise technologies

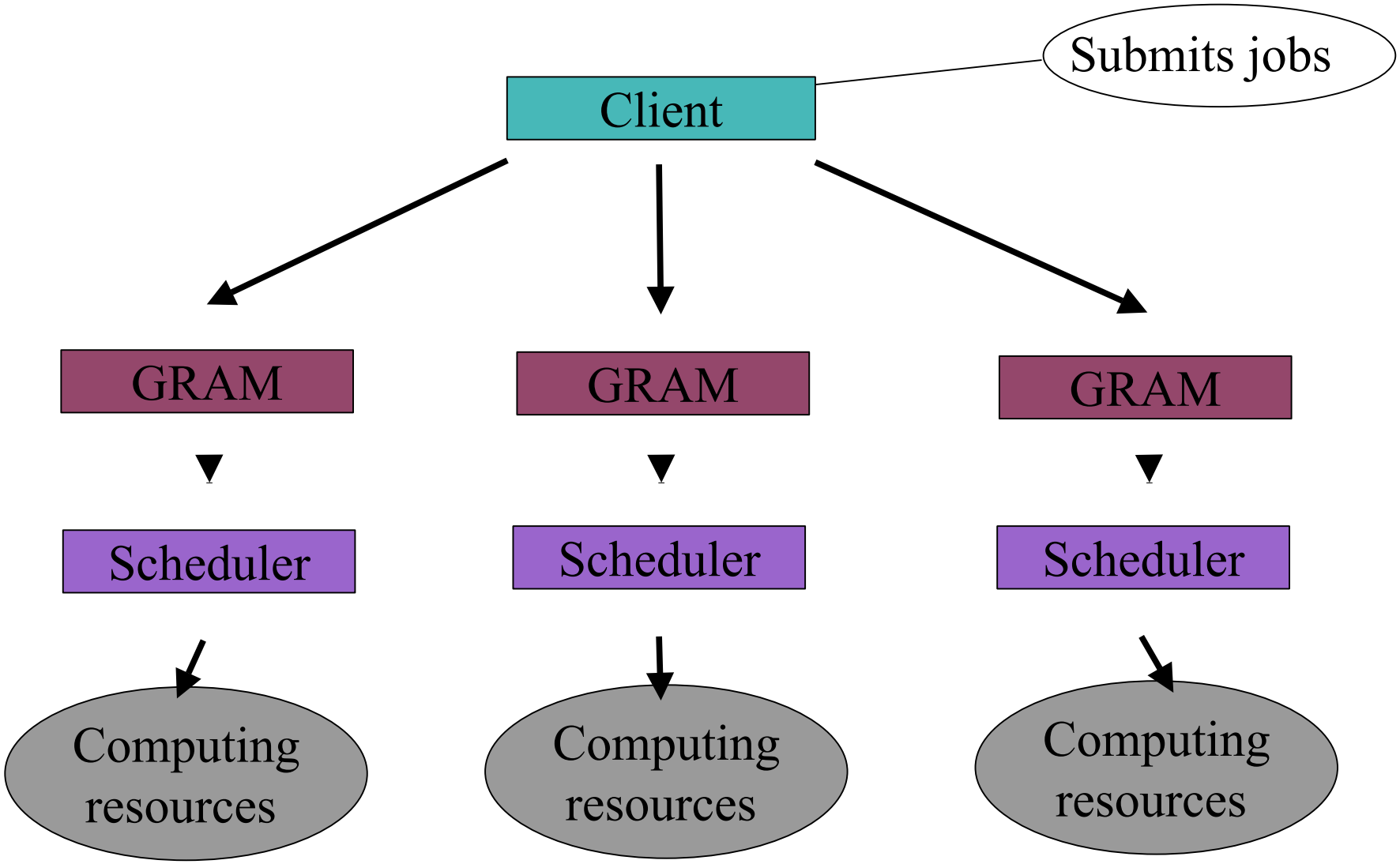
Note

This project is a work-in-progress.

Please download the most recent version of the presentation from:

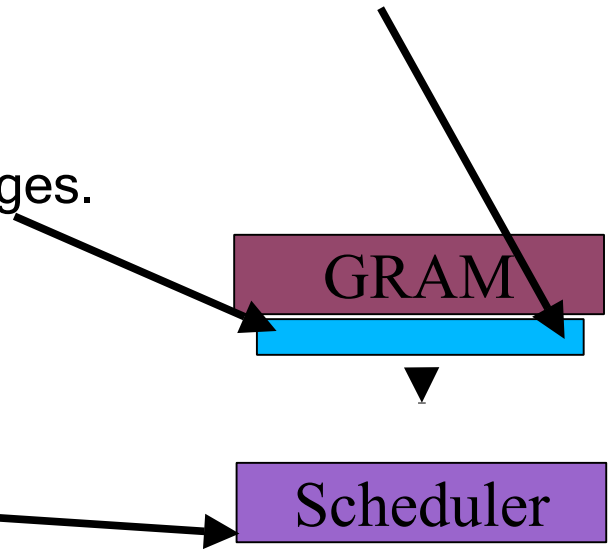
http://gridwisetech.com/resources

GRAM Overview



What do we need to cope with?

- Job Manager Scheduler Interface.
  - Set of Perl modules that implement scheduler-specific interfaces.
- Scheduler Event Generator.
  - C module that monitors job state changes.
- Local job schedulers.
  - PBS/TORQUE.
  - SUN N1 Grid Engine.
  - etc.



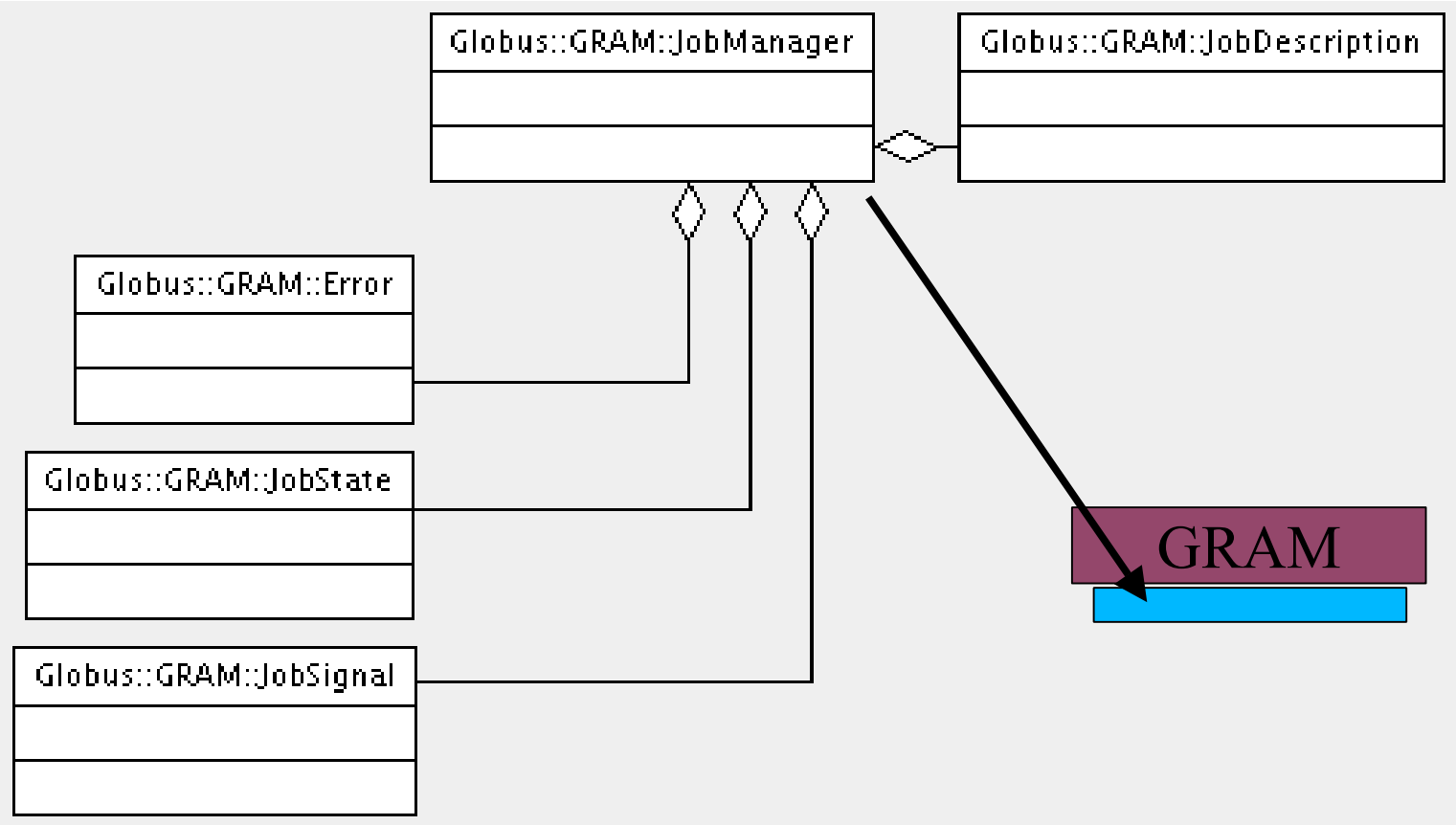
Good news

- Job Manager Scheduler Interfaces are compatible with both existing versions of GRAM.
  - Pre-WS GRAM
    - Uses the whole implementation of the interface.
  - WS GRAM
    - Uses a subset of the Pre-WS GRAM methods.

Job Manager Scheduler Interface

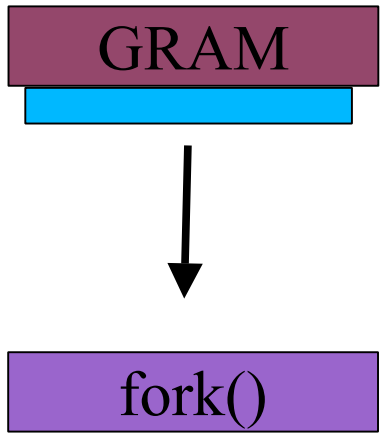
- There are a few files containing Perl modules needed to setup Job Manager Scheduler Interface (files are always named as the modules with .pm extension).
  - Globus::GRAM::JobManager – *Base class for all JobManager scripts.*
  - Globus::GRAM::Error – *GRAM Protocol Error Constants.*
  - Globus::GRAM::JobState – *GRAM Protocol JobState Constants.*
  - Globus::GRAM::JobSignal – *GRAM Protocol JobSignal Constants.*
  - Globus::GRAM::JobDescription – *GRAM Job Description.*

Schedulers' interface class diagram



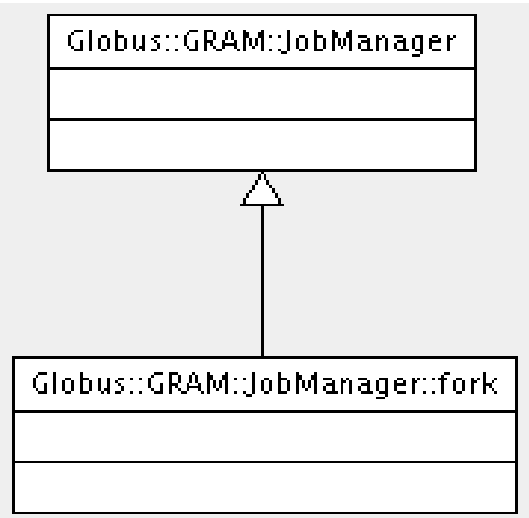
Simplifying the model

- Globus Toolkit contains basic interface to job scheduler – Fork.
- Not really a job manager scheduler interface, just the ability to spawn new jobs using fork() function.
- Fork is the only preinstalled interface and is a default one.
- Helps to test the environment and will help us to understand integrating GRAM with schedulers.



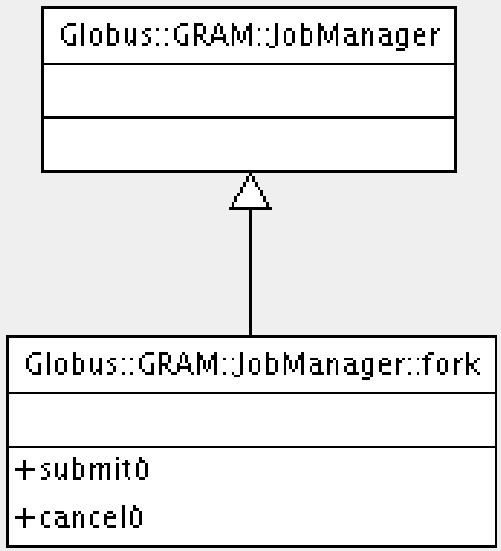
Specific job scheduler interface

- In order to use/write an interface for the specific job scheduler we need to care only for one module.
  - Globus::GRAM::JobManager::name_of_the_scheduler
- There is one such subclass of JobManager for each job scheduler. In case of Fork there is:
  - Globus::GRAM::JobManager::fork
  - Contained in fork.pm file.



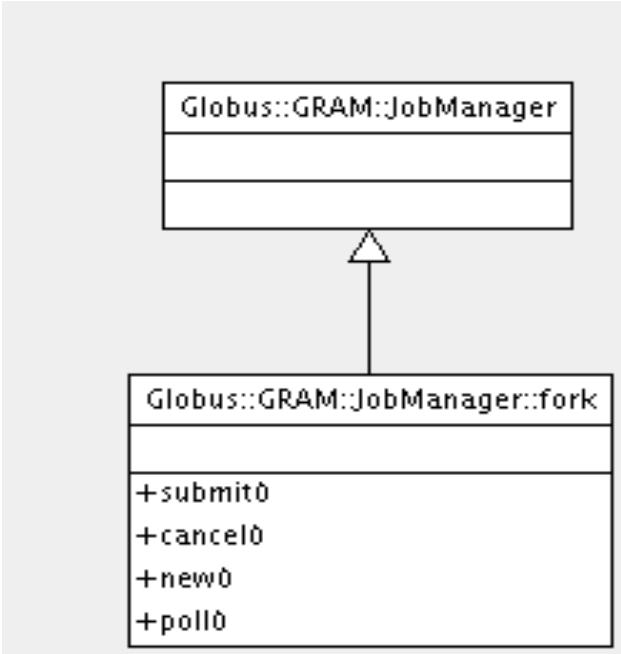
Module internals overview - fork.pm

- Each Job Schedulers' interface is implemented as a subclass of the Globus::GRAM::JobManager module.
- The most important methods that must be implemented are:
  - “submit”
    - This method is called when job manager submits the job to the scheduler.
    - “submit” method recieves the information of the original job request through the JobDescription data member.
  - “cancel”
    - This method allows to cancel a scheduled job while it's running or waiting in a queue.



Module internals overview - fork.pm

- Our sample Job Scheduler's interface additionally consists of:
  - A constructor.
    - "new" method acts as a constructor
    - If there is nothing specific to setup the default Globus::GRAM::JobManager::new will do the job.
    - Otherwise we can overload "new" method as fork does.
  - "poll" method.
    - "poll" method is used only by Pre-WS GRAM implementation.
    - The purpose of this method is to check for updates of the job's status.



Scheduler Event Generator (SEG)

- New, WS GRAM, job manager uses Scheduler Event Generator module for receiving events from schedulers.
- It is used instead of constantly polling schedulers with “poll” method (less overhead – performance improved).
- SEG module is implemented as C shared library.
- At the time of this presentation SEG module parses schedulers' logs to generate new events with job state changes.

SEG in a bigger context

- All SEG modules uses *Globus Scheduler Event Generator API* to send messages about current state of the submitted jobs.
- Job states are sent to the *JobStateMonitor (JSM)* – a scheduler-independent object that provides notifications about job state changes to the *ManagedJobService (MJS).*
- It provides an interface between the process monitoring of the SEG module and the Managed Job Resources which contain the state of a job created by the ManagedJobService (MJS).

seg_fork_module.c - example

- The core of the SEG module is described in the globus_module_descriptor structure:

```
globus_module_descriptor_t
globus_scheduler_event_module_ptr =
{
  "globus_scheduler_event_generator_fork",
    globus_l_fork_module_activate,
    globus_l_fork_module_deactivate,
    NULL,
    NULL,
    &local_version,
    NULL
};
```

- The module's activate (*globus_l_fork_module_activate)* and deactivate (*globus_l_fork_module_deactivate)* functions are the only functions called by the *globus scheduler event generator* program.

- After the activation, the module starts issuing events based on the states of the submitted jobs.

seg_fork_module.c - example

- SEG module's task is to parse log files to obtain states of the jobs. Fork's log file is fairly simple:

```
...
001;1105565642;885;2;0
001;1105565645;883;8;0
001;1105565645;881;8;0
...
```

- Log's inputs are partitioned through *sscanf(3)* function:

```
rc = sscanf(p, "%d;%ld;%n%*[^;]%n;%d;%d",
            &protocol_msg_type,
            &stamp,
            &jobid_start,
            &jobid_end,
            &job_state,
            &exit_code);
```

seg_fork_module.c - example

- There are various states that the job can be in after submission to the scheduler.
- When new job state is discovered *SEG Fork module* uses *Globus Scheduler Event Generator* API to issue an event:

```
switch(job_state)
{
case GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING:
    globus_scheduler_event_pending(stamp, jobid);
    break;
case GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE:
    globus_scheduler_event_active(stamp, jobid);
    break;
case GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE:
    globus_scheduler_event_done(stamp, jobid, exit_code);
    break;
case GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED:
    globus_scheduler_event_failed(stamp, jobid, exit_code);
    break;
default:
    goto bad_line;
}
```

SEG modules for other schedulers

- Implementations of SEG modules for real-life schedulers are not that trivial as in Fork example.
- Log files of the scheduler can have less deterministic structure and be more difficult to parse than the Fork's logs.
- Logs are usually rotated in some manner and SEG module has to be aware of this to prevent data loss.
- Currently, all SEG module implementations use log files to collect job state changes. However, if a scheduler provides an API to receive events it could be used instead.

Integration procedure

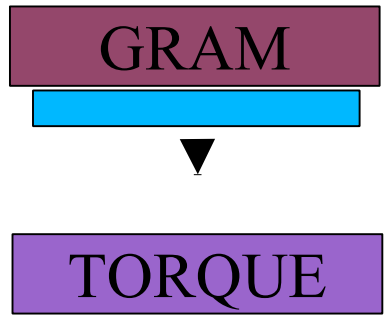
gridwise technologies

Job Schedulers' Adapters

- Job Schedulers' interface implementations come in prepared packages in the form of tarballs.
- Packages are prepared using Grid Packaging Toolkit (GPT) used in Globus Toolkit.
- Implementations available in GT4 includes:
  - Portable Batch System interface (gt4-gram-pbs)
  - Platform LSF interface (gt4-gram-lsf)
  - Condor interface (gt4-gram-condor)

TORQUE example

- TORQUE *(Tera-scale Open-source Resource and QUEue manager) is a resource manager.*
- It is based on the Portable Batch System (PBS) implementations such as OpenPBS.
- It is also a job scheduler (basic but with possibility to alter it by other, specialized schedulers).
- The fact that it is based on *PBS products makes it a good choice for integrating with Globus Toolkit's PBS implementation of Job Scheduler's Adapter.



Installation process

- Installation process using GT4 PBS adapter is straightforward.
  - First, we need to have complete installation of Torque with scheduler's commands in the path.
  - Next, in the GT sources directory the following commands need to be issued:
    - *make gt4-gram-pbs*
    - *gpt-postinstall*
  - This will install the adapter and register the new functionality in Globus Toolkit installation.

Configuration process

- Configuration tool for the interface is located in: $GLOBUS_LOCATION/setup/globus/setup-globus-job-manager-pbs.

- There are parameters to this tool that can be used to change default settings, ie:

  - *--service-name jobmanager* changes default jobmanager interface in the Globus Toolkit installation to this one.

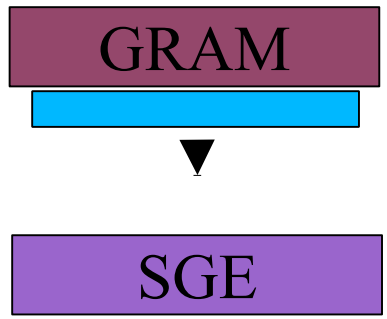
  - *--remote-shell=ssh* specifies how interface will try to access remote shell.

Configuration process

- The only thing left is to associate local resource managers with GridFTP servers. This is done by mapping file system paths to enable staging of files.
- It is done by editing *$GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml*
- The complete example of the file for PBS is located in *WS GRAM : Public Interface Guide* in the Globus Toolkit documentation.

SUN N1 SGE example

- SUN N1 Grid Engine is a complete solution for resource management and scheduling.
- It is a commercial Sun Microsystems product that started as a community project.
- Globus Toolkit does not contain implementation of Job Scheduler's Adapter for SGE.
- There exists some unofficial implementations of the adapter for older versions of the Globus Toolkit.
- We are going to look into details of this existing solutions and/or come up with our own implementation.



Conclusions

- There are variety of resource managers used to create homogeneous environments.
- All those environments can be used for grid-wide computations, but we always need to know how to talk to them.
- By the use of GRAM we can create heterogeneous grid environment with one, uniform interface to various resources located and governed at different sites.

Thank you!

- Most recent version of this presentation:
  http://gridwisetech.com/resources

- Other resources:
  http://www.globus.org/toolkit/

  http://www.clusterresources.com/products/torque/

  http://wwws.sun.com/software/gridware/

- Send feedback to:
  pawel@gridwisetech.com