Globus Online Future

Steve Tuecke

April 12, 2011

Identity provider

- **Turn Globus Online into a full-fledged, federated identity provider**
  - Shibboleth, OpenID, Oauth, MyProxy CA
  - For GO services & external service providers
- **Can authenticate with Globus Online using other federated identity providers**
  - Shibboleth (e.g. your campus credential)
  - OpenID (e.g. Google), OAuth (Facebook , TeraGrid)
  - MyProxy (e.g. NERSC)
- **REST and LDAP interfaces for provisioning**
- **Sync accounts with external identity provider**

Group management

- **Self-help creation and management of groups of Globus Online identities**
- **Admission workflow engine**
  - User can request admission to group
  - Group specifies profile attributes user must provide
  - Configurable admission policy (e.g. owner approval)
- **REST and LDAP interfaces for provisioning**
  - Groups of native Globus Online identities
  - Groups projected with only federated identities

Application Domestication

- **Other applications can use Globus Online as a platform to provide:**
  - Federated identity management
  - Group management
  - File transfer

Profile management

- **Extensible set of profile attributes can be associated with an account**
  - Notification addresses (email, SMS #, IM, etc)
  - Other attributes required for group admission
- **Eventually want to federate this information, so that it can come from another source**
  - E.g. Facebook, Shibboleth, etc.

Alerts

- **Actionable conditions of interest to users**
  - File transfer service has rich event stream
  - Must reduce that to actionable alerts
    - E.g. Credential expired, file not found, slow transfer, …
- **Flexible notification based on alerts**
  - Alerts classified by severity level
  - Each severity level has notification method
  - Reasonable defaults, and customizable by user
- **Flexible fault handling based on alerts**
  - Configure which faults are fatal errors, critical, informational, should be ignored

File transfer enhancements

- **Remove files**
- **Make directories**
- **Credential refresh**
- **Suspend / resume**
- **File verification**
- **Group support**
  - Group endpoints
  - Transfer monitoring
- **Transfer priorities**
- **Symlink handling**
- **Anonymous FTP**
- **HTTP support**
- **SRM support**
- **UDT support**
- **rsync support**
- **Optional encryption**

Tools for Site/Endpoint Administrators

- **For all transfer to/from my public endpoints:**
  - Monitor (status, details, events)
  - Get alerts
  - Suspend/resume/cancel
  - Periodic summary reports
- **More endpoint configuration options**
  - E.g. Maximum and recommended transfer settings

Globus Connect

- **Simple update**
- **Allow access to only specific directories**
- **Bandwidth limiting**

Globus Connect Multi-User

- **Easily connect your multi-user Linux server to Globus Online**
  - No host certs; service cert from Globus Online
  - MyProxy CA for local Linux users (NIS, LDAP)
  - GridFTP with automatic gridmap management
    - Optionally limit to select users
  - Optional firewall/NAT handling
    - Outbound connections only + control channel relay
  - Automatic endpoint management
  - Simple update

Web interface

- **Transfer files**
  - Handle large directory listings
  - Sync
  - Quick endpoint addition
- **View transfers**
  - Select columns, column order, etc.
- **Manage endpoints**
  - Table support CRUD on endpoints
  - Favorites
- **Skins (branded interface)**

The next big steps…

- **Cloud file storage**
  - File server tied to Globus Online identity and group management system
  - Amazon based storage vs local site storage?
- **File sharing**
  - Allow transfers / access between users
  - Tied to cloud storage vs peer-to-peer?

Virtual Endpoints

12/04/2011

Gopi Kandaswamy

Information Sciences Institute

University of Southern California

Goals

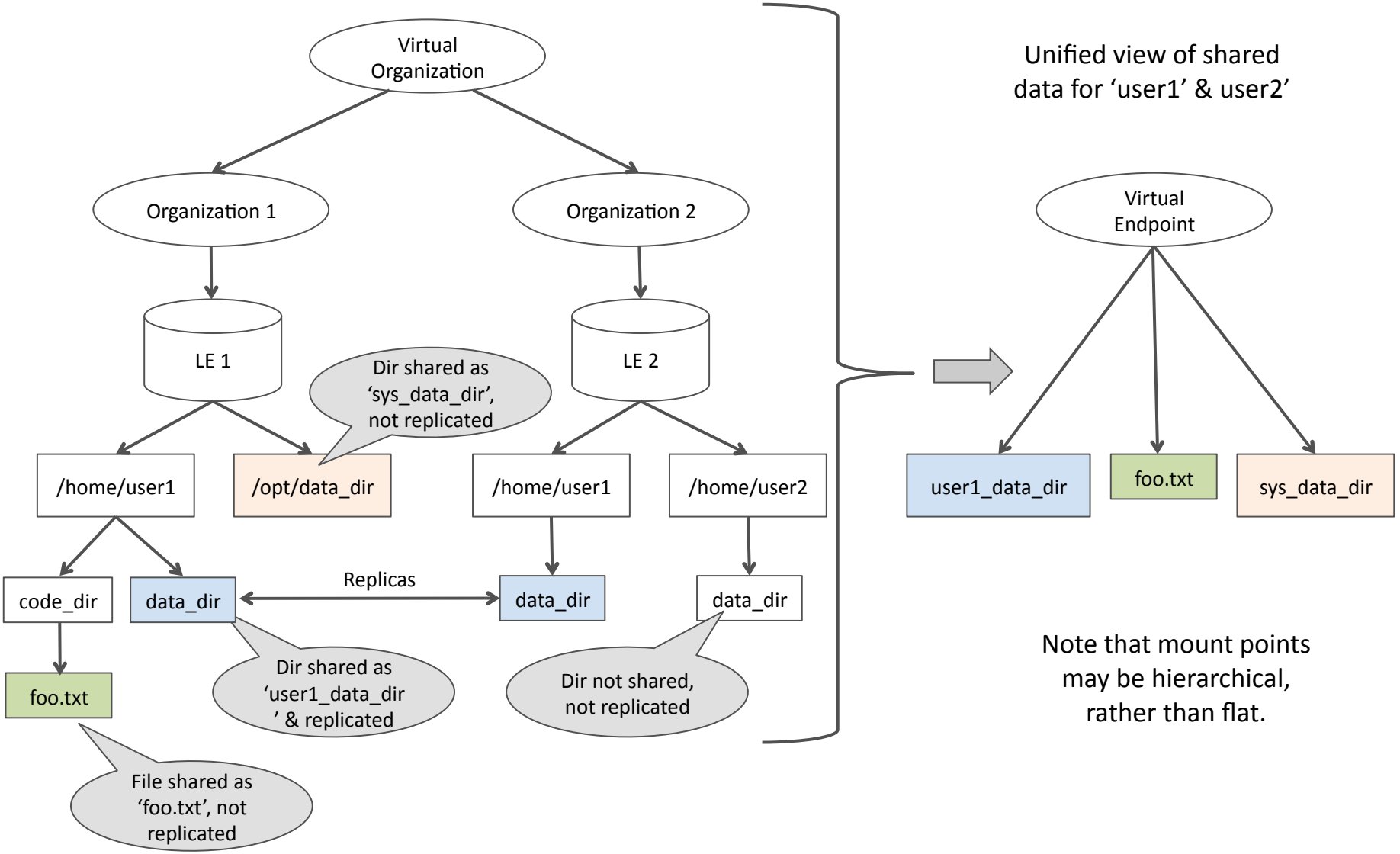
- Provide users with a unified view of the shared data in a virtual organization.
- Track multiple data instances.
- Improve fault tolerance and performance using automatic policy based mirroring.

Background

- A virtual endpoint (VE) is a collection of logical endpoints (LE).
  - Each LE may be administered by a different resource provider with a different set of policies.
  - Each VE has a VE-level policy apart from the policies of its constituent LEs.
  - Creator of a VE serves as its administrator, and sets policy for that VE.
- A LE may be part of zero or more VEs.

Background (contd.)

- Virtual Endpoint Server (VES)
  - Keeps track of, and implements VE-level policy.
  - Manages locks on shared resources.
  - Performs policy based synchronization.
- READ operation: Refers to transferring files from VE to LE where LE is not part of that VE.
- WRITE operation: Refers to transferring files from LE to VE where LE is not part of that VE.


Virtual Organization
Organization 1
Organization 2
LE 1
LE 2
Dir shared as 'sys_data_dir', not replicated
/home/user1
/opt/data_dir
/home/user1
/home/user2
code_dir
data_dir
Replicas
data_dir
data_dir
foo.txt
Dir shared as 'user1_data_dir ' & replicated
Dir not shared, not replicated
File shared as 'foo.txt', not replicated
Unified view of shared data for 'user1' & user2'
Virtual Endpoint
user1_data_dir
foo.txt
sys_data_dir
Note that mount points may be hierarchical, rather than flat.

VE Commands (1/7)

- Create a VE
  - *endpoint-add --virtual VE1 -e LE1://home/user1/data LE2://Users/user1/data --policy policy.txt*
- This creates a VE from 2 LEs with a policy.
- The file system path where data is mirrored on each LE is also specified.

VE Commands (2/7)

- Write files to a VE
  - *scp -r LE3://home/users/user1/foo VE1://user1_data*
- This copies the source dir to all mirrors in VE.
- User must have file system write permissions on all mirrors.
- If dir VE1://user1_data does not exist, it is created and contents of ‘foo’ are copied into it.
- If dir VE1://user1_data exists, then ‘foo’ becomes a sub-dir of VE1://user1_data.

VE Commands (3/7)

- Read files from a VE
  - *scp –r VE1://user1_data LE3://tmp*
- This selects an optimal mirror from VE1 as source.
- The user must have file system read permissions on at least one mirror in VE.

VE Commands (4/7)

- Edit files on a VE
  - *checkout VE1://user1_data LE1*
- Need write permissions on checked out dir.
- No user can write to mirrors of checked out dir.
- Any authorized user can read from mirrors of checked out dir.

VE Commands (5/7)

- Commit edited files to a VE
  - *checkin VE1://user1_data LE1*
- Need write permissions on all mirrors of the dir to checkin.
- Locks all mirrors, copies files from LE1 mirror to all mirrors

VE Commands (6/7)

- Undo edits
  - *rollback VE1://user1_data LE1*
- This will undo all changes made after '*checkout*' of data on LE1.

VE Commands (7/7)

- Remove a VE
  - *endpoint-remove VE1*
- This removes the VE; no data is deleted

Questions?

Implementing globus clients

...using Grisu in one way or other

Grisu is...

...basically a wrapper around grid (mostly Globus) technologies:

- Job submission (GT4/GT5)
- File transfer (gridftp)
- Resource information (mds/bdii)
- Authorization/Authentication (x509/SLCS/MyProxy/VOMS)

Fair enough, but why?

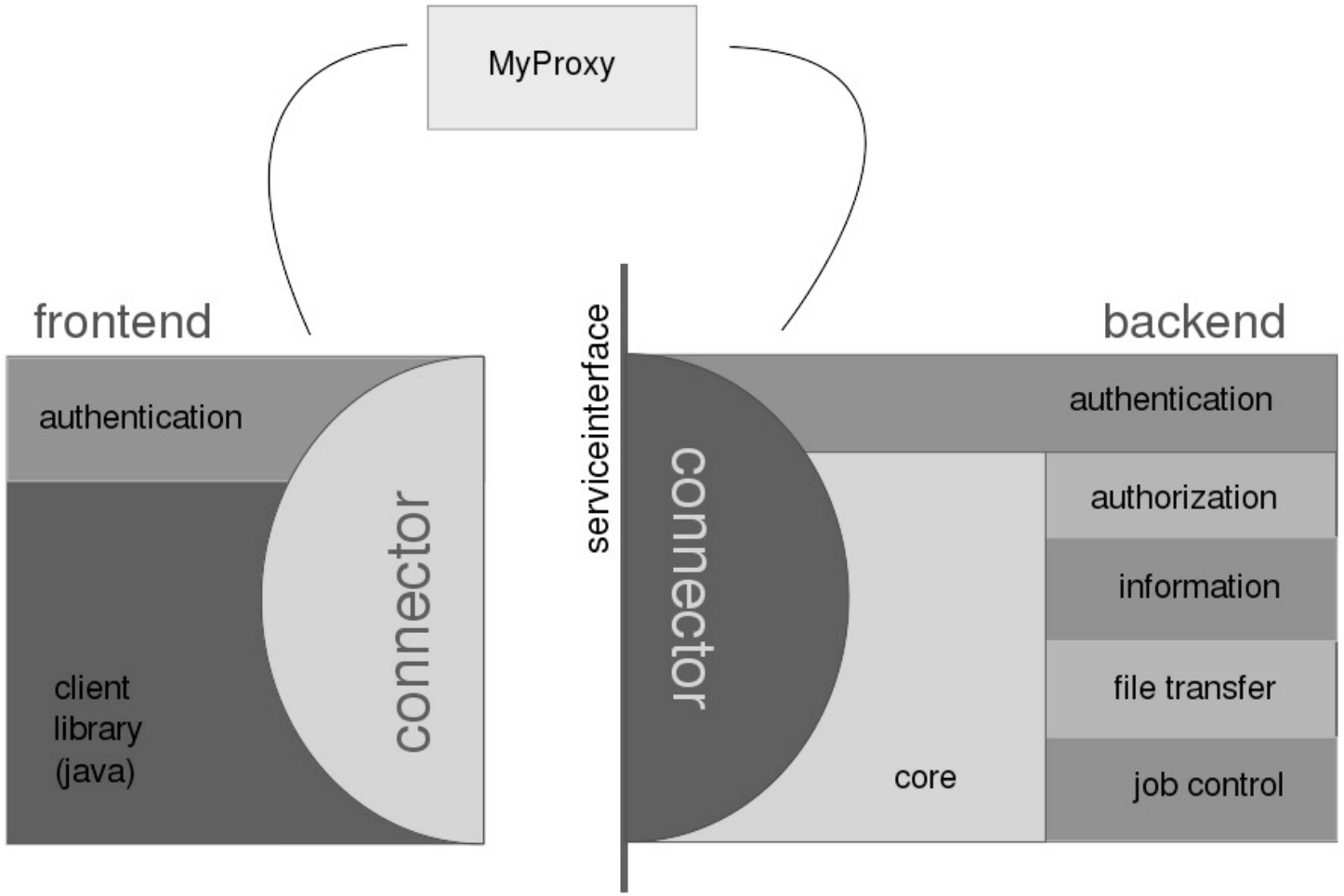
Grid in New Zealand / Australia:

- (very) small development teams
- many individual users with basic job submission needs (NZ)
- many heterogeneous small/medium sized research projects
  - different data needs
  - different compute needs
  - different level of IT literacy
- many heterogeneous sites with very different network policies

Thus, Grisu:

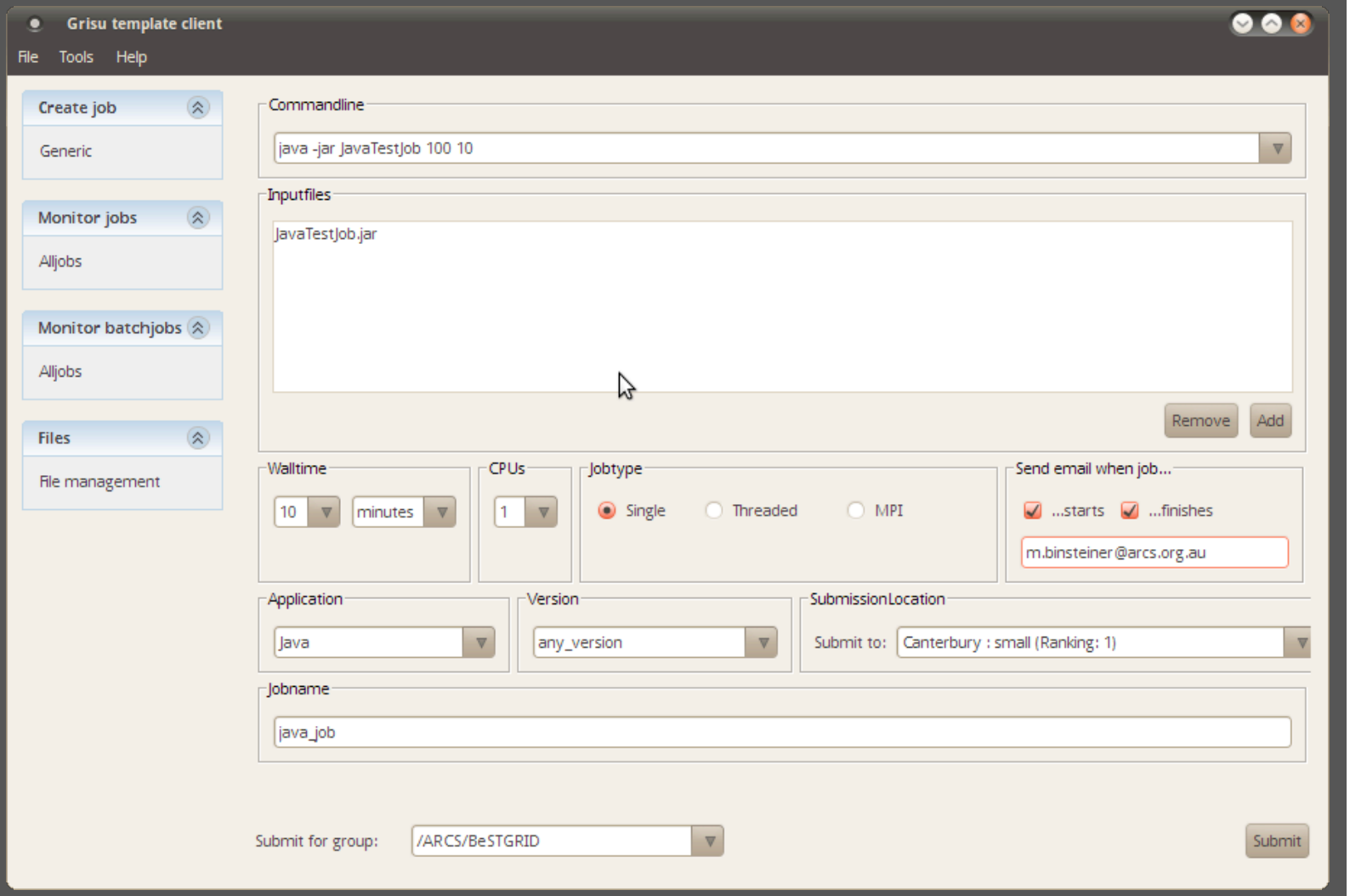
- main goals:
  - easy to use for (non-grid) developers
  - hide grid technologies
  - hide grid terminology from users (still struggling...)
  - try to identify and cover the more basic, generic usecases -- ignoring more specialized ones if need be (for now at least)

- design objectives:
  - object oriented
  - backend - frontend architecture
  - one AMAP interface per grid technology


MyProxy
frontend
authentication
client
library
(java)
connector
serviceinterface
backend
connector
authentication
authorization
information
file transfer
job control
core

Several ways to create something that submits something to the grid (using Grisu):

Template for Swing client



...using the new gui template editor that comes with the next version of Grisu (or a plain text editor).

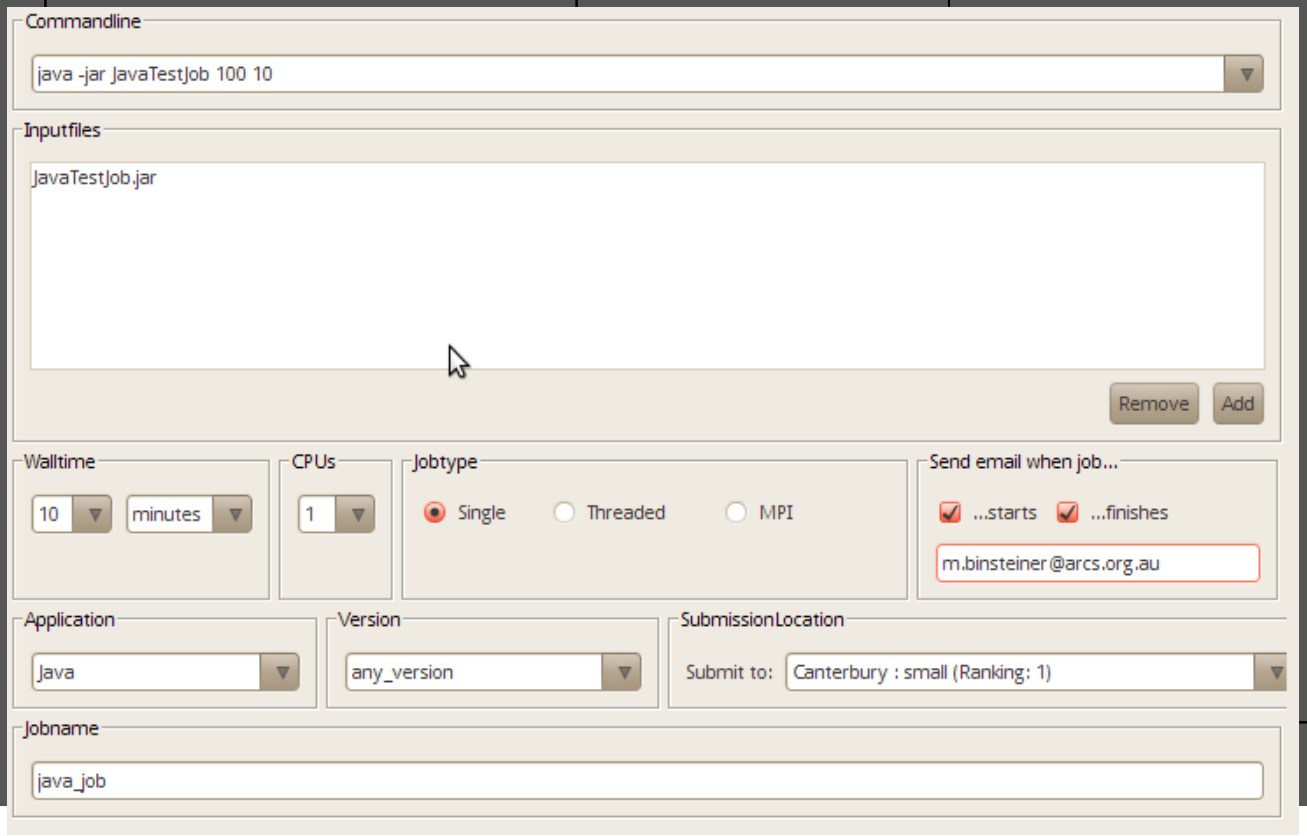
Generic template

```
commandline = ${cmdlWidget}
= Generic =
-------
[cmdlWidget]
type = SimpleCommandline
title = Commandline
size = 2000x70
-------
```

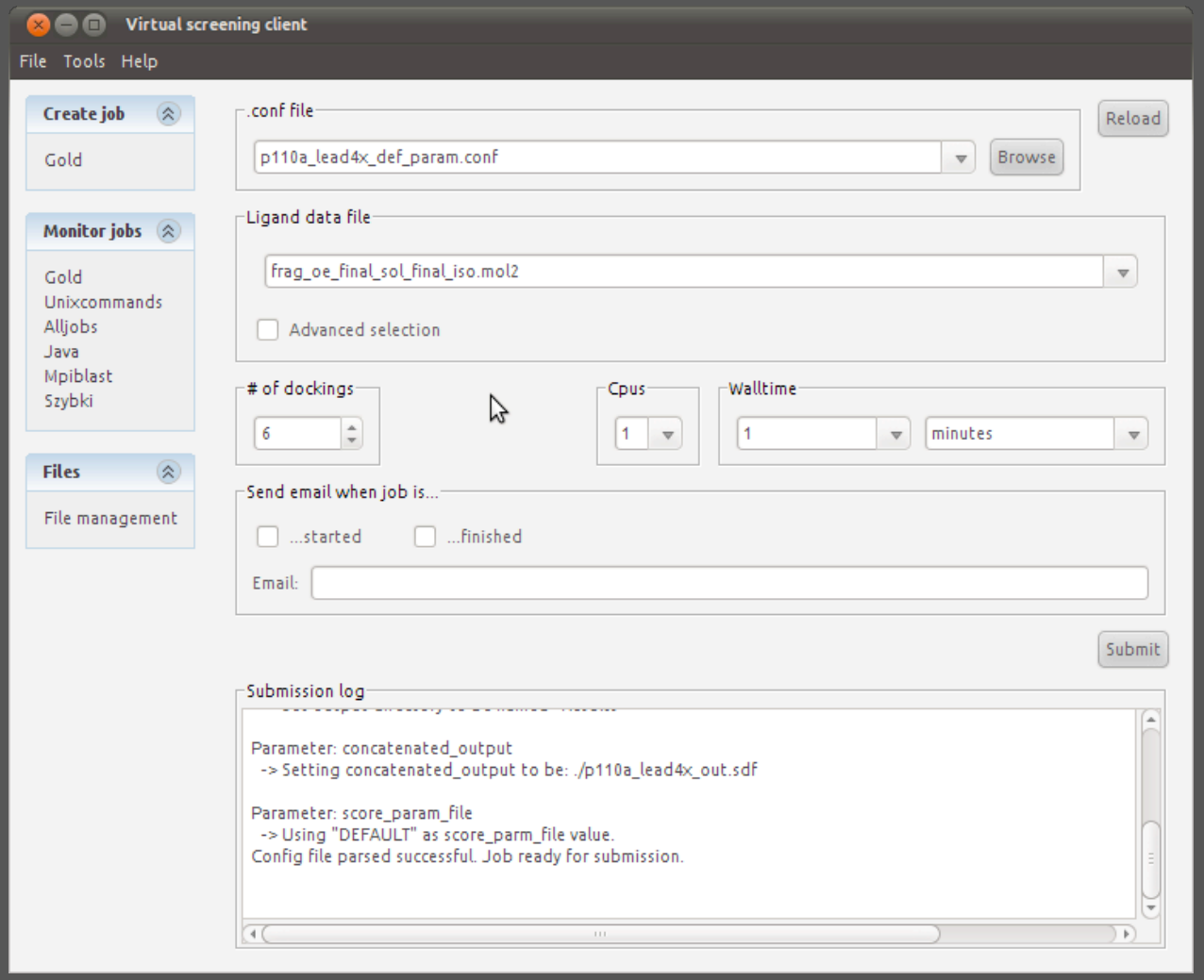
```
[inputFiles]
type = MultipleInputFiles
size = 2000x200
title = Inputfiles
--------
[walltime]
type = Walltime
title = Walltime
defaultAmount = 10
defaultUnit = minutes
size = 200x100
```

```
[cpus]
type = Cpus
title = CPUs
size = 100x100
defaultValue = 1
[jobtype]
type = JobType
size = 2000x100
```

```
[email]
type = Email
size = 2000x100
--------
[appInfo]
type = ApplicationSelector
size = 2000x70
title = Application
[appVersion]
type = ApplicationVersionSelector
size = 2000x70
title = Version
[queueSel]
type = QueueSelector
title = SubmissionLocation
size = 2000x70
-------
[jobname]
type = Jobname
defaultValue = templatetest
title = Jobname
size = 2000x100
```

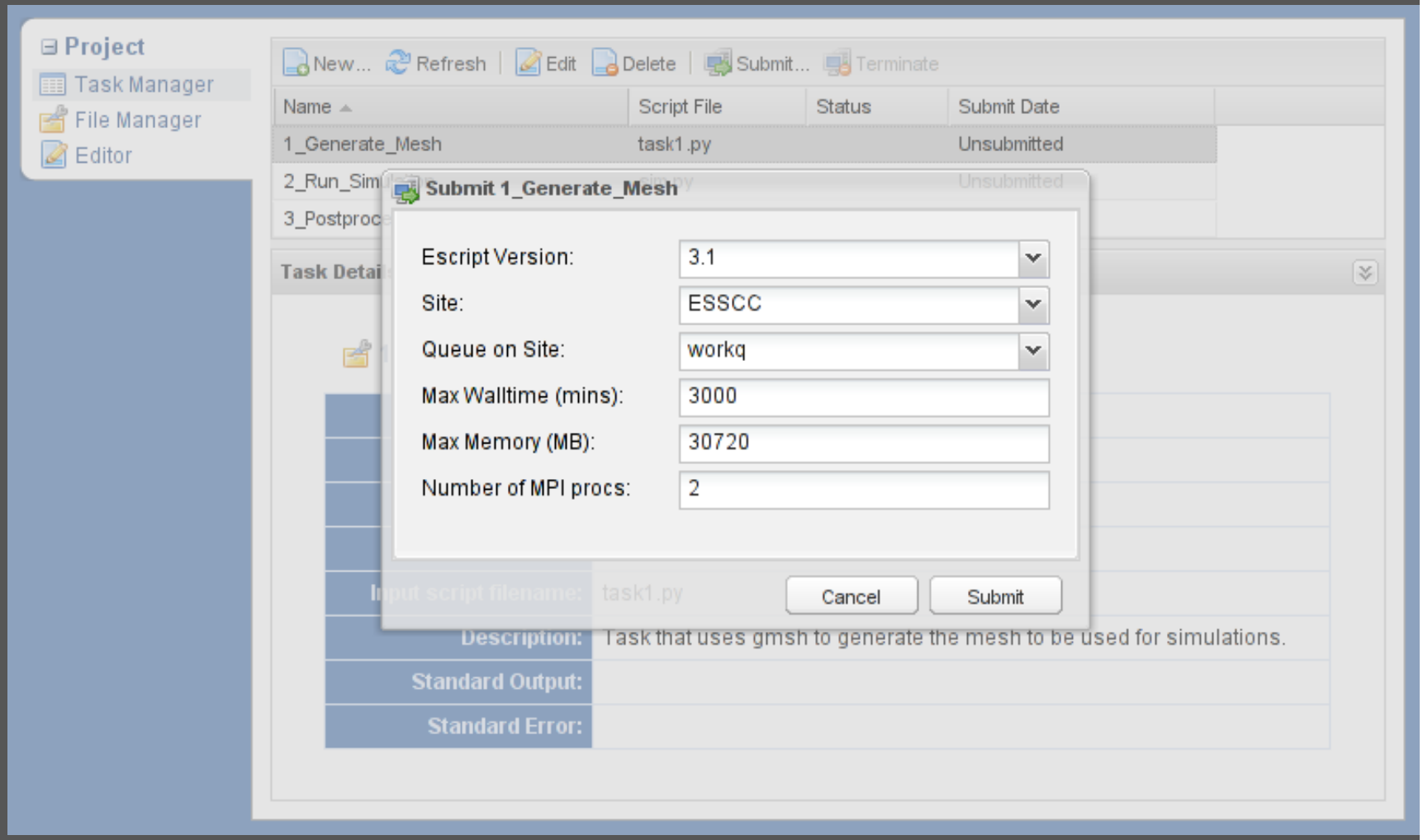


App-specific job creation panel



...using the Grisu client library & ready-made widgets

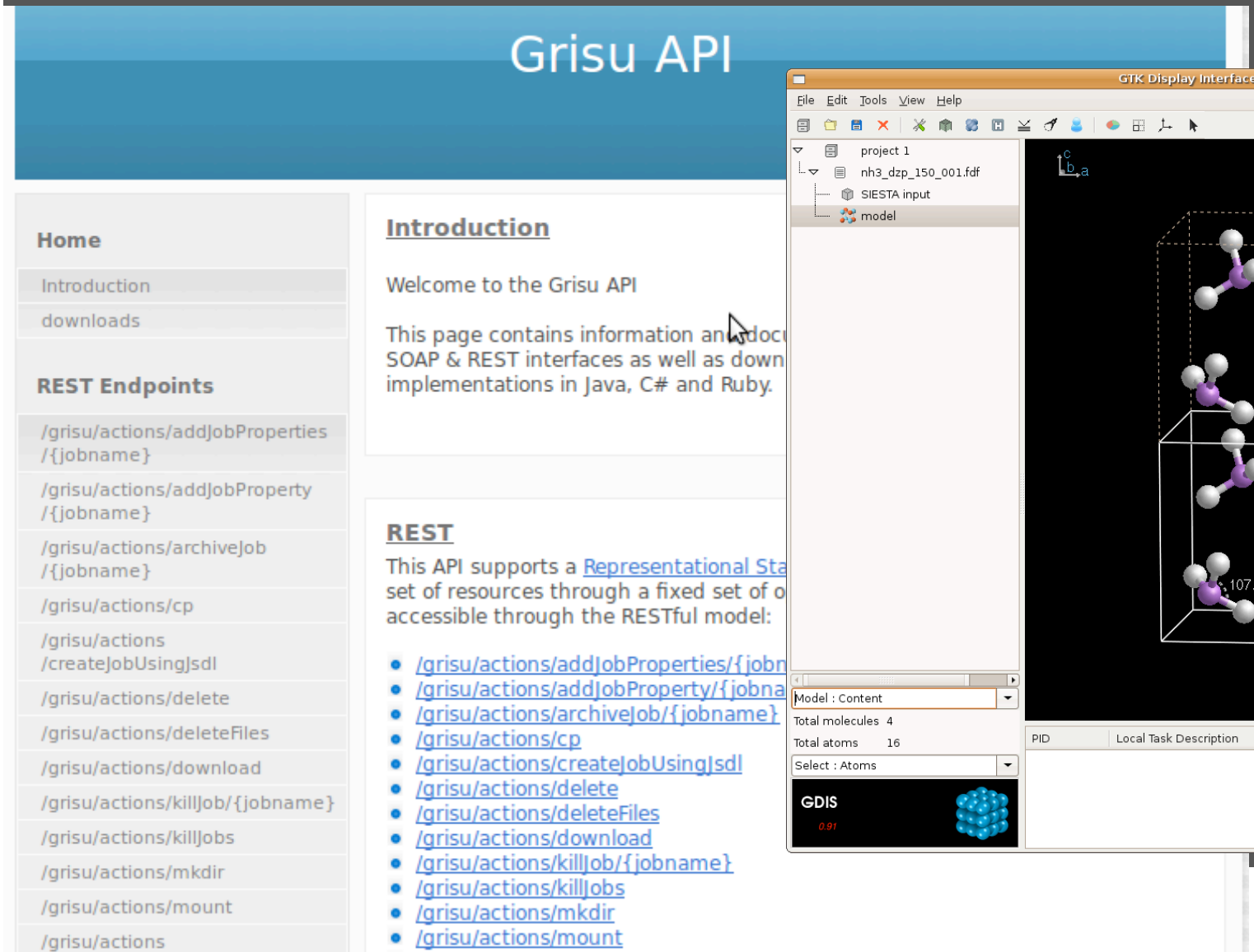
Use the Grisu client library



... to create a grid client in the java technology of your choice.

Use the Grisu ReST/SOAP API

...in order to write a grid client in the language of your choice.



Jython scripts

```
from grisu.frontend.control.login import LoginManager
from grisu.frontend.model.job import JobObject

si = LoginManager.loginCommandline()

job = JobObject(si)
job.setUniqueJobname('cat_job')
job.setCommandline('cat singleJobFile_0.txt')
job.addInputFileUrl('/home/markus/test/singleJobFile_0.txt')

job.createJob("/nz/UoA")
job.submitJob()

job.waitForJobToFinish(10)

print 'Job finished. Status: '+job.getStatusString(False)
print 'Stdout: ' + job.getStdOutContent()
print 'Stderr: ' + job.getStdErrContent()

job.kill(True)
```

gricli (currently in development):

- Commandline shell, can be used interactively (with tab-completion) or scripted

```
set vo /nz/UoA
set application UnixCommands
attach /home/markus/localFile.txt
attach grid://groups/nz/NeSI/gridftpFile.txt
attach grid://jobs/archived/javajob/result.txt
attach http://www.gnu.org/licenses/gpl.html
submit cmd "cat localFile.txt gridftpFile.txt result.txt gpl.html"
   >> job name is gricli_1285812002395

print jobs
   >> gricli_1285812002395 : Done

download job gricli_1285812002395
   >> download of job gricli_1285812002395 complete...
```

GO / GC & Grisu

Common problems:

- Firewalls (we've got a few very strict ones here in NZ)
- Transferring files from/to the desktop is slow
- Code that handles transfer is not very smart

Possible solution:

- GlobusOnline/GlobusConnect!
  - package up GlobusConnect and install it alongside Grisu
  - replace gridftp-gridftp filetransfer plugin with GO
  - client-side, test whether GC can, well, ...connect
  - if not, use current (slow but firewall-friendly) method, otherwise do filestaging via GC/REST
  - bonus: transfer database independent of Grisu backend

Grisu is on GitHub:

http://github.com/grisu/grisu