Java and Python CoG Kits

Keith R. Jackson
krjackson@lbl.gov
Gregor von Laszewski
gregor@mcs.anl.gov

Objectives

- What are the CoG Kits, and why do I want to use them?
- How do I use the CoG Kits?
- How do I develop my own components with the CoG Kits?
- Future

Motivation for CoG Kits

- **Problem**
  - Many application developers desire to program the Grid in frameworks familiar to them to for example enable rapid prototyping.

- **Solution**
  - The CoG Kit project integrates Grid software based on the Globus Toolkit and a commodity framework such as Java or Python.
    - Easier development of advanced Grid services
    - Easier and more rapid application development
    - Easier deployment of Grid services
    - Code reuse and use of component repositories
    - Use of Web services as part of the Grids
    - Widespread use of the Grid

CoGs are more …

- CoGs are more than just an interface to the Globus Toolkit
- CoGs allow Grid programmers to use the *Commodity Technologies* AND the *Grids* advantages
  - Example: Event and exception model of Java
  - Example: SWIG wrappers in Python for rapid prototyping and legacy code support.
- Thus, CoGs are not just an API but provide access to the Commodity *Framework*

Reasons for using the Java CoG Kit

- Why use the Java CoG Kit?
  - keeping up with patches and protocol changes in Globus has been difficult in the past
  - The Java CoG Kit has so far provided the community with a "service" to hide the burden of changing your code.
    - Ask for a testimony by the XCAT group at Indiana University.
  - Many users need only the features provided in the Java CoG Kit
  - Little/no changes were so far involved to switch between different versions of the Globus Toolkit
  - Mostly client side programming
  - Java Platform:
    - High level framework better suited for Grid programming than C
- Why not use the Java CoG Kit
  - It does not provide all the latest features of the Globus Toolkit.
  - It uses Java, and some in the community … ;-)

Reasons for Using the Python CoG Kit

- Why use the Python CoG Kit?
  - Provides a full interface to the Globus Toolkit
  - Little/no changes are involved in switching between different versions of the Globus Toolkit
  - High level language allows for easier Grid programming
    - Supports rapid prototyping of Grid services/applications
  - Many automated tools exist for exposing legacy C/C++ or Fortran codes as Python objects
- Why not use the Python CoG Kit
  - Small performance penalty for using any interpreted language
    - Minimized because the Python CoG Kit is a thin wrapper over the native C code.
  - No static type checking
  - It uses Python, and some in the community … ;-)

Motivation: Java and Python CoG Kits

- Use and leverage existing technologies for Grid programming
  - The capabilities of the framework onto which Grid Services are mapped can be exploited:

    Objects, Events, Exceptions, JNDI, ...
  - Objects like jobs/tasks can be defined.
  - XML support is provided.
  - GUI's, ..., IDE's can be used (Forte, BOA Constructor…)
- Maximize software flexibility, extensibility, and reusability
- Provide foundations for application developer teams that are familiar to develop applications in this framework
  - Reduce development and maintenance cost
- Use as glue for many technologies
  - Python is well suited to tying together many different languages/technologies

What is the Java CoG Kit ?

- The Java CoG Kit provides a mapping between Java and the Globus Toolkit. It extends the use of Globus by enabling to access advanced Java features such as events and objects for Grid programming.
- The Java CoG Kit is implemented in pure Java. It speaks the Grid protocols.
- It is not a wrapper of the C Globus Toolkit
- This allows integration within applets.
- Mostly client side support

What is the Python CoG Kit?

- Similarly the Python CoG Kit provides a mapping between Python and the Globus Toolkit. It extends the use of Globus by enabling to access advanced Python features such as exceptions and objects for Grid programming.
- The Python CoG Kit is implemented as a series of Python extension modules that wrap the Globus C code.
- Uses SWIG (http://www.swig.org) to help generate the interfaces.

Status: Java CoG Kit

- Modified core Globus components (Protocols)
- Basic services are provided accessing:
  - Security (GSI)
  - Remote job submission and monitoring (GRAM)
  - Remote Data Access (GridFTP)
  - Information Service Access (MDS)
  - Certificate store (myProxy)
- Current 100% client side components includes
- Reusable Grid GUI components
- A variety of Grid Interfaces

Status: Python CoG Kit

- Basic services are provided accessing:
  - Security (security)
  - Remote job submission and monitoring (gramClient)
  - Secure high-performance network IO (io)
  - Protocol independent data transfers (gassCopy)
  - High performance Grid FTP transfers (ftpClient)
  - Support for building Grid FTP servers (ftpControl)
  - Remote file IO (gassFile)
  - Information Services access
- High level services for easier usage
- Task based services to encapsulate common usage patterns

Communication

- Web Page
  - www.globus.org/cog
  - www-itg.lbl.gov/gtg/projects/pyGlobus/
- Bugs
  - Java:
    - http://www-unix.globus.org/cog/contact/bugs/

      Is maintained by the Globus Project but does not contain some of the older comments
    - http://arbat.mcs.anl.gov:8080/bugzilla/

      Is our old bugzilla
  - Python:
    - http://www-itg.lbl.gov/bugzilla/
- Download
  - www.globus.org/cog
  - http://www-unix.globus.org/cog/java/
  - www-itg.lbl.gov/gtg/projects/download/download_info.html

Versions

- Java Versions
  - 0.9.13 is a release compatible with Globus 2.0
    - we recommend using Globus 2.0 if you like to use this release
  - 1.1a is compatible with Globus Toolkit 2.2 and in future Globus Toolkit 3.0, e.g. much of it may be distributed as part of Globus Toolkit 3.0
    - We recommend using this version for Globus 2.2
    - Still an alpha release
- Python Versions
  - 0.9.8
    - Two versions, one for GT2.2 and one for GT2.0

Download & Compile

Java CoG Setup Options

- Download Source or binary
  - http://www.globus.org/cog
- Binary
  - Read the Readme
  - setup
  - change the environment variables
- Source
  - Read the readme
  - setup
  - change the environment variables
- Manual will be available shortly

Python CoG Kit Setup Options

- Download source
  - Set Environment variables
  - Compile
  - Install
- Easy to create RPM's for Linux systems
- Binary installer coming for win32
  - As soon as Globus officially releases the win32 port
- Can build binary packages for any platform
- Uses the standard Python distutil module

Requirements

- Java CoG
  - JDK 1.3.1 or 1.4.1
    - http://java.sun.com
  - ant
    - http://jakarta.apache.org/ant/
- Python CoG
  - Python 2.0 +
    - http://www.python.org
  - Globus Toolkit Installation
  - GPT Installation

Java CoG Compilation

- cd work
- cd ogce
- ant dist
- cd ..
- <inspect work>
- work
  - ogce
  - jglobus
  - build
    - cog-1.1a

      FAQ.TXT

      bin // here are the commands

      lib // here are the jars

      etc

- we create a build directory in which the jar files and shell scripts are located that allow
  - (a) programming and
  - (b) use of client tools from the command line

Python CoG Kit Compilation

- Ensure that GPT_LOCATION and GLOBUS_LOCATION are set appropriately
- cd pyGlobus-{Version}
- python setup.py build
  - --prefix=/path/to/installation/directory
  - Will only build those packages that you have Globus installs of
- python setup.py install
  - To install in the site-extensions directory requires root privilege

Obtain Keys

- Obtain Globus credentials
  - User private key
  - User certificate
  - Trusted CA certificates
  - Take a look at http://www.globus.org/security
    - Obtain a certificate from your local organization CA
    - you can install a simple CA on your machine to create your own certificates.
    - you could install part of the Globus toolkit that would allow you to run

      globus-cert-request
- Place credentials in common place
  - Users home directory in .globus directory

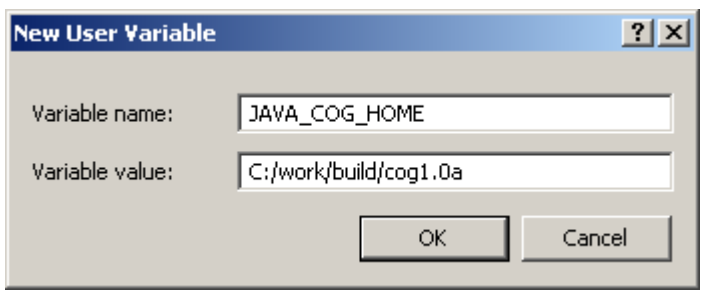
Setup of the Java CoG Kit

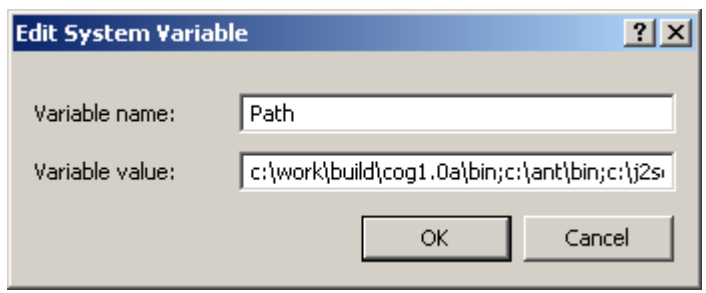
Setup has two purposes

- Enable access to command lines (optional)
  - path and other environment variables
- Enable access to certificates (mandatory)
  - convenient setup component

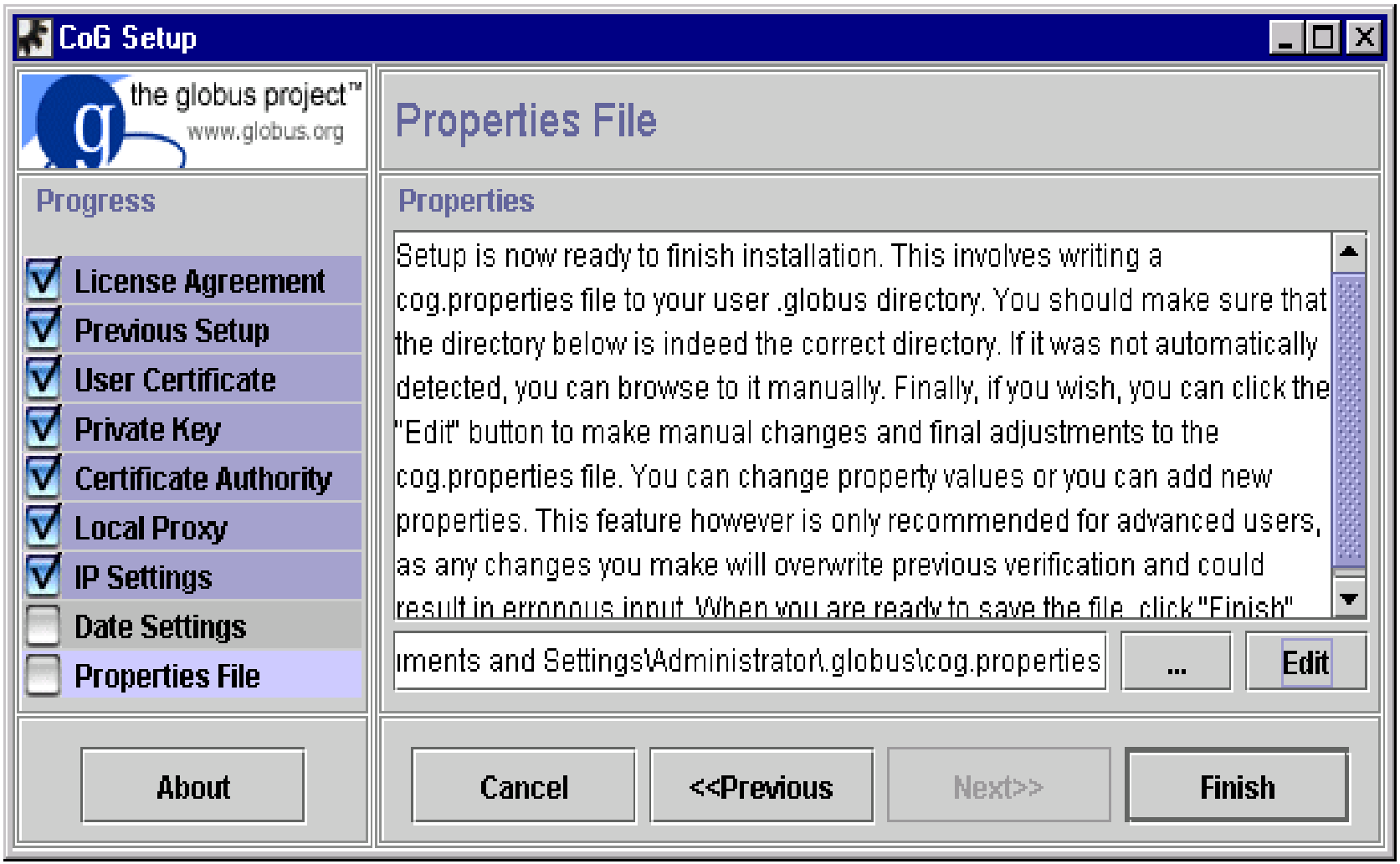
Using the command line tools (optional)

- add work/build/cog1.0a/bin/<OS> to your path
- Unix
  - export PATH=$PATH:/home/gregor/work/build/cog1.0a/bin
  - export JAVA_INSTALL_PATH=/home/gregor/work/build/cog1.0a
  - should be : ;-) JAVA_COG_HOME=/home/gregor/work/build/cog1.0a
- Windows
  - Registry
    - Control Panel->System->Advanced->Environment Variables
    - New Variable JAVA_COG_HOME
    - Edit the Path Variable





Setup Component (mandatory)



Setting up the Java COG Kit

- This involves writing a cog.properties file to your user .globus directory which contains information regarding the following:
  - The location of your **user certificate** file that identifies you as a trusted user and is usually named usercert.pem
  - The location of your **private key** file that usually takes the name userkey.pem and is used to encrypt information you send out, so that recipients with a public key could verify your identity
  - The location of the **Certificate Authority (CA)** file. The CA is a third party that is used to certify the link between the public key and the subject in the certificate.
  - The location of the **proxy file** that is used to temporarily identify you on the grid. The file usually takes the name x509up_u_username and can often be found in your temporary directory.
  - The system **IP address** and the **date** also needs to be checked.

It is advised that the certificates also be stored in the default .globus directory.

Example cog.properties file

- You can customize the cog.properties files by hand.
  - see the FAQ.TXT in the bin directory
- relevant Code
  - work/jglobus/src/org/globus/common/CoGProperties.java
  - work/ogce/src/org/globus/ogce/gui/cogsetup
- Common values set in cog.properties
  - usercert=/home/globoid/.globus/usercert.pem
  - userkey=/home/globoid/.globus/userkey.pem
  - proxy=/tmp/x509up_u999
  - cacert=/etc/grid-security/certificates/42864e48.0, /etc/grid-security/certificates/5aba75cb.0
  - ip=192.123.123.1

Visual components

One example of a
user interface to the Grid

Java Beans

- **Local System File Browsing**

  LocalTreeFrame Bean provides the capability to browse the files present in the local system.

- **Remote System file Browsing**

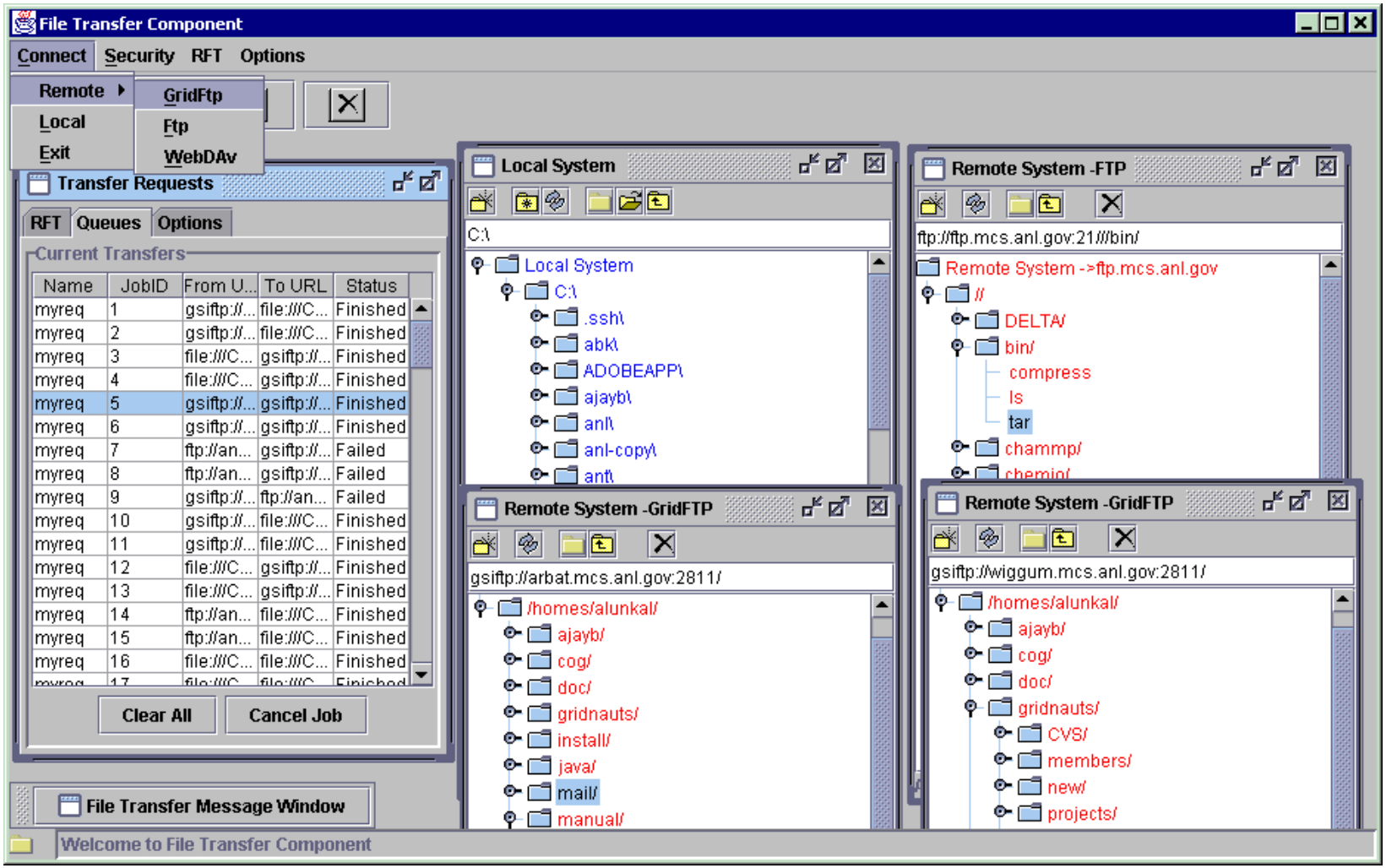
  RemoteTreeFrame Bean is the basic bean, which provides generic remote file browsing functionalities. Currently we provide GridClient and FtpClient beans that utilize the RemoteTreeFrame to provide implementation for ftp and gridftp servers respectively.

- **File Transfer**

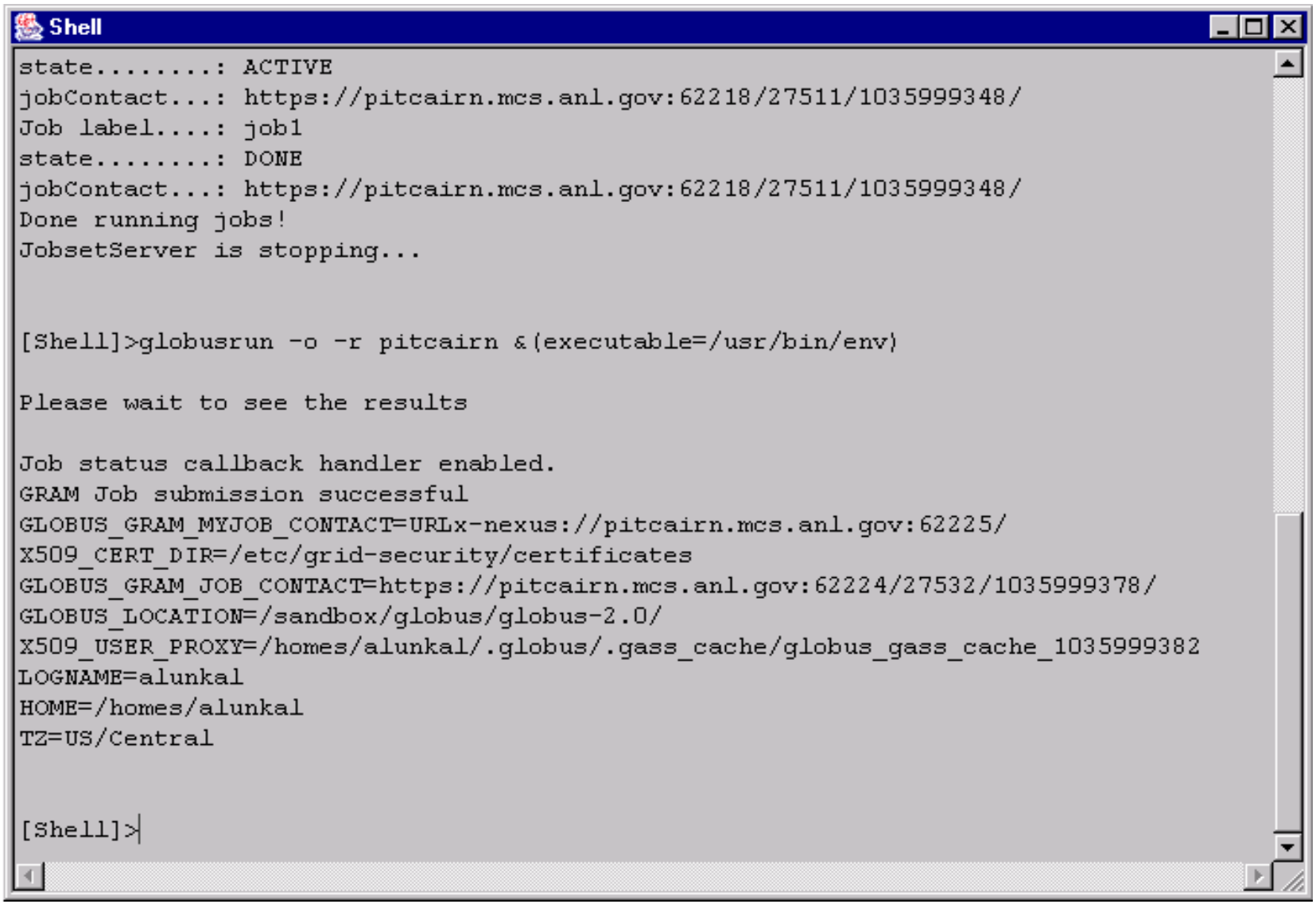
  FileTranfer Bean provides the file transfer capabilities.

Available : org.globus.ogce.beans.filetransfer.*

File Transfer Component



Shell Component



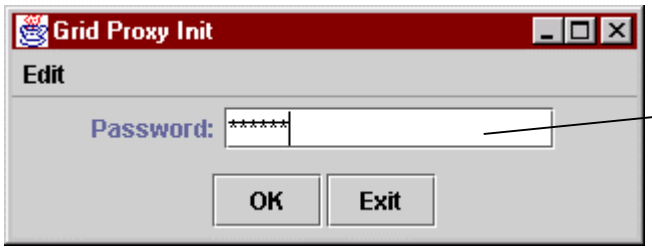
Shell Component (commands)

- Help
  - Prints out all the commands supported by the shell
  - help <command> gives specific syntax help
- Basic shell commands
  - mkdir, cd, ls, pwd, rm, rmdir, cp, cls, exit …
- File Transfer Consoles to perform all file transfer operations.
  - ftp to open a connection to FTP site
  - gridftp to open a connection to gridFTP site

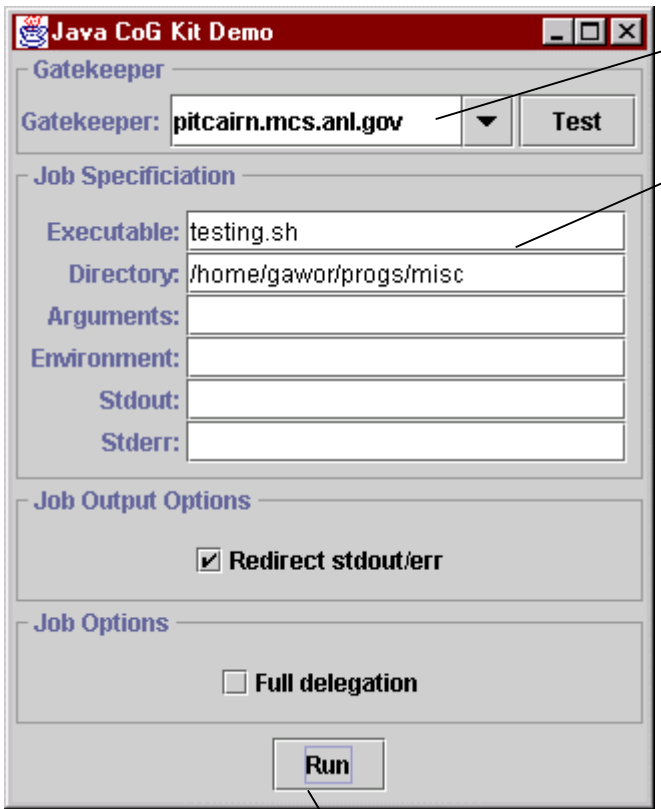
Shell Component (commands)

- Grid commands
  - globusrun — run jobs on remote machine
  - globus-url-copy — copy files
  - globus-jobset — submit set of jobs with dependencies to a remote machine
- Background Thread option (&) commands
  - ps — view all the running processes
  - resume — resume a process
  - suspend — suspend a process
  - kill — kill a process thread
  - result — see the output of the process
- *Batch files*
  - *ant –f grant.xml*
  - *run filename*

Visual Job Submission



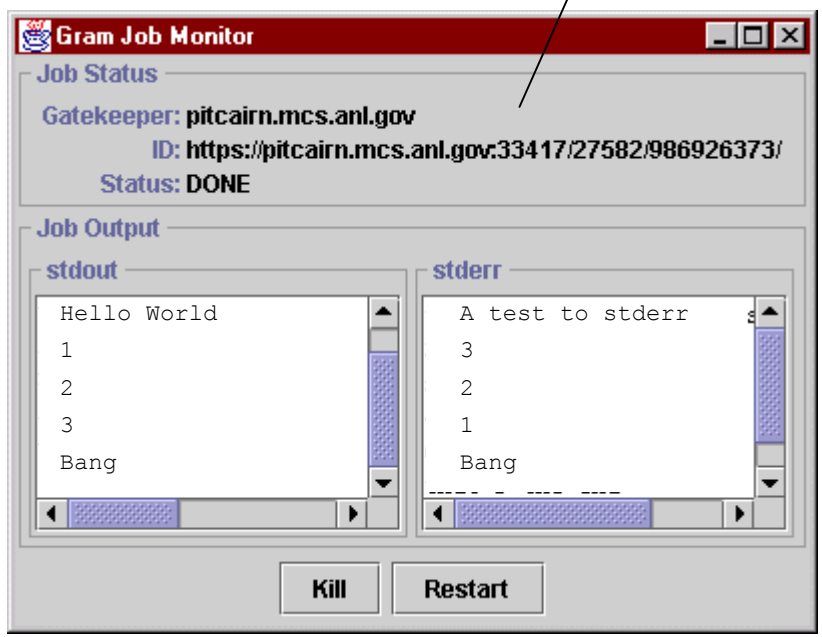
Authentication
(Enter Passphrase)



Select Machine

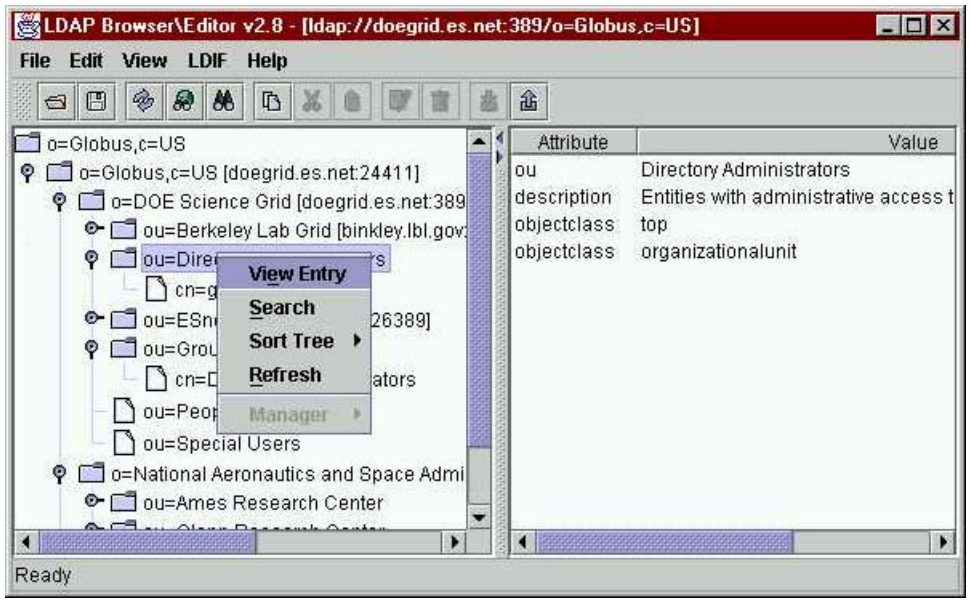
Specify Job

Observe output



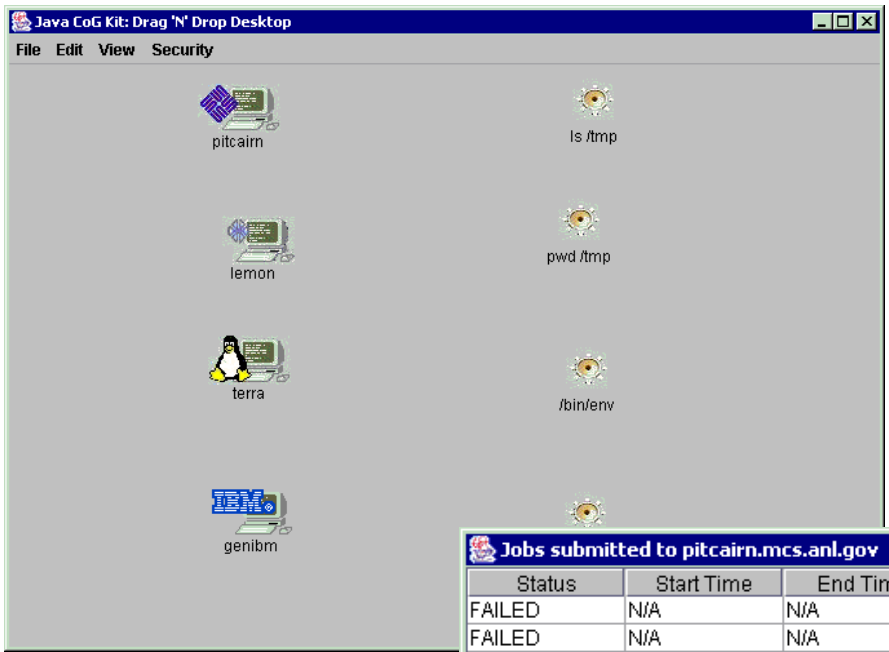
Run the Job

Grid Information/LDAP Browser



- Download
  - http://www-unix.mcs.anl.gov/~gawor/ldap/
- Separately distributed
  - Historical reasons, one of the first Java CoG Kit components
  - thousands of users
    - also outside of Grid community

Desktop

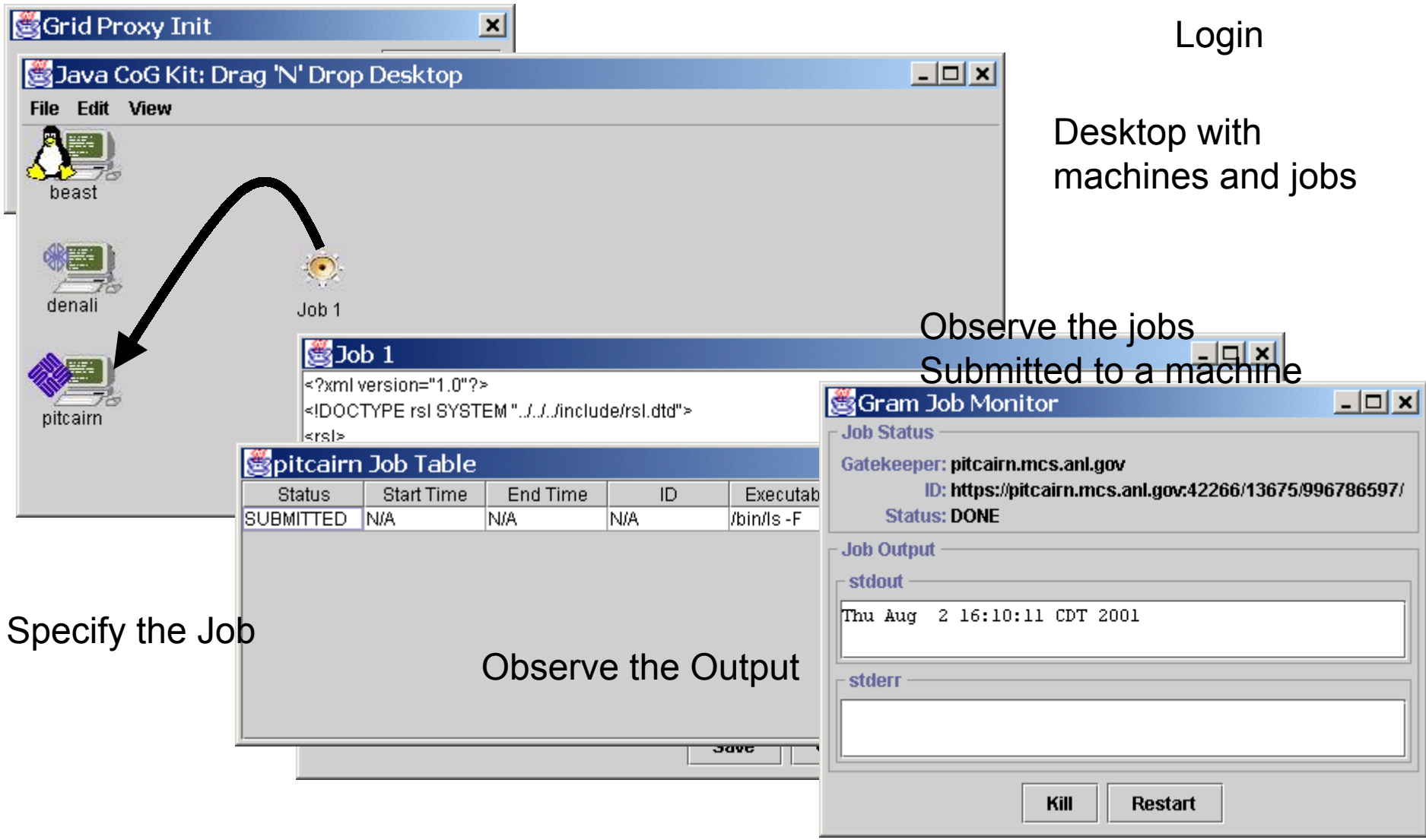


- Familiar look and feel
- prototype for what you can do

Jobs submitted to pitcairn.mcs.anl.gov

Status	Start Time	End Time	Directory	Executable	stdout	stderr
FAILED	N/A	N/A			N/A	N/A
FAILED	N/A	N/A			N/A	N/A
DONE	Fri Jul 19 20:4...	Fri Jul 19 20:4...	/tmp	/bin/pwd -F	gsiftp://pitcairn...	gsiftp://pitcairn...
DONE	Fri Jul 19 20:4...	Fri Jul 19 20:4...	/tmp	/bin/env	gsiftp://pitcairn...	gsiftp://pitcairn...
DONE	Fri Jul 19 20:4...	Fri Jul 19 20:4...		/bin/ls	N/A	N/A
DONE	Fri Jul 19 20:5...	Fri Jul 19 20:5...		/bin/ls	N/A	N/A
DONE	Fri Jul 19 20:5...	Fri Jul 19 20:5...	/tmp	/bin/ps -F	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A	/tmp	grid-proxy-init -F	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		grid-proxy-init -F	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		ps -ef	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		ps	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		date	gsiftp://pitcairn...	gsiftp://pitcairn...
DONE	Fri Jul 19 21:0...	Fri Jul 19 21:0...	/tmp	/bin/env	gsiftp://pitcairn...	gsiftp://pitcairn...
DONE	Fri Jul 19 21:0...	Fri Jul 19 21:0...	/tmp	/bin/pwd -F	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		cat	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		whot	gsiftp://pitcairn...	gsiftp://pitcairn...
FAILED	N/A	N/A		whoami	gsiftp://pitcairn...	gsiftp://pitcairn...
DONE	Fri Jul 19 21:0...	Fri Jul 19 21:0...	/tmp	/bin/ls -F	gsiftp://pitcairn...	gsiftp://pitcairn...

Grid Desktop



Login

Desktop with machines and jobs

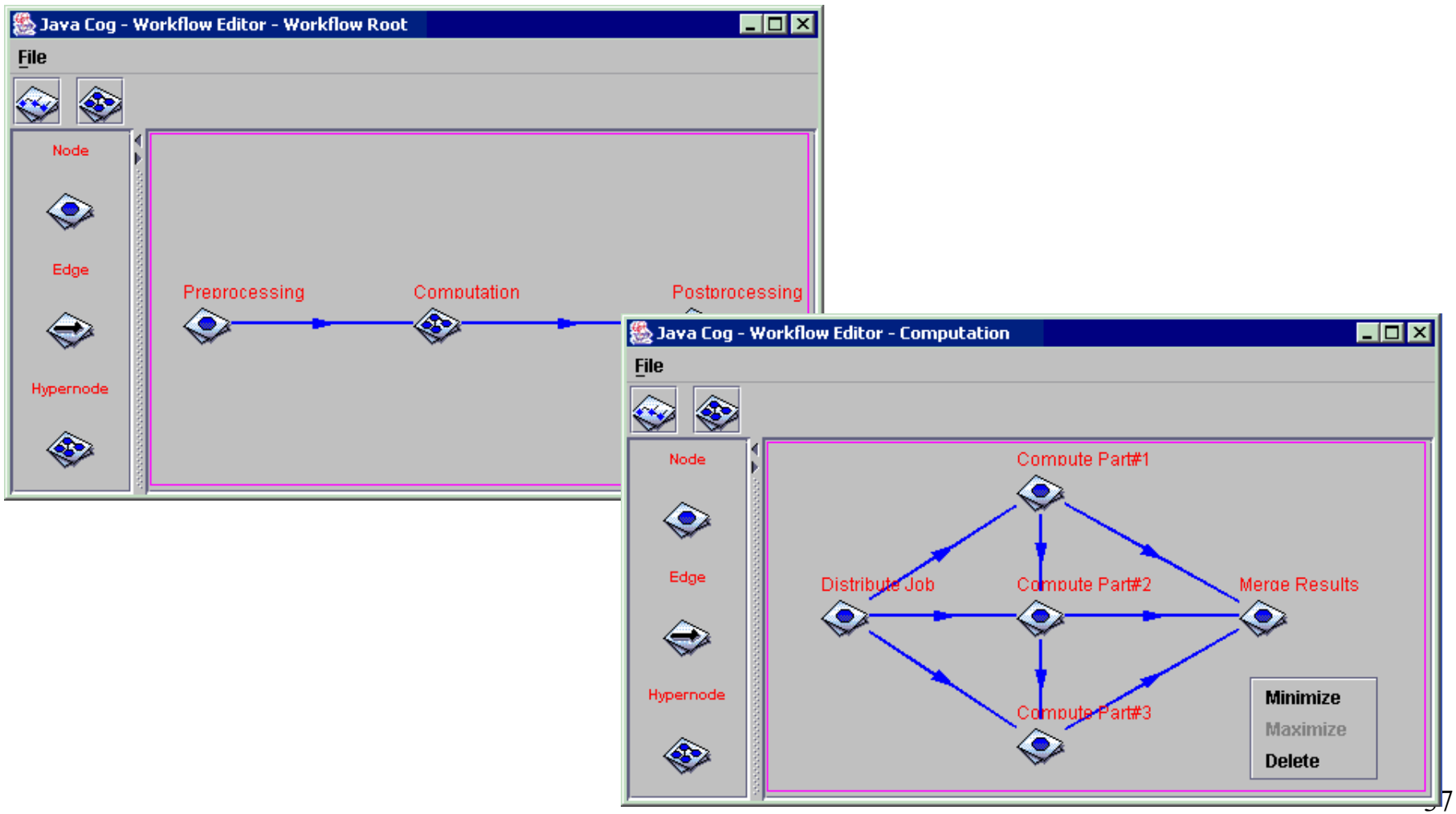
Observe the jobs Submitted to a machine

Specify the Job

Observe the Output

Workflow Component

- under development



Java Command Line Tools

Java Command Line Tools

- Security
  - grid-proxy-init — authenticate to grid
  - grid-cert-info — gives cert information
  - grid-proxy-destroy — destroy the proxy
  - myproxy — starts a myproxy-server
- File Transfer
  - globus-url-copy — simple URL-to-URL copy
- Job execution
  - globusrun — run both batch and interactive jobs
- Personal Gatekeeper
  - globus-personal-gatekeeper — run a simple server
- Gass Server for tranfering files for execution
  - globus-gass-server — starts the gass server
  - globus-gass-server-shutdown — shuts down the server

Setup for Running Tools

- COG_INSTALL_PATH and PATH are the variables you need to setup in order to run the tools.

- Procedure :
  - Build and install the Java CoG Kit package.
  - Set the COG_INSTALL_PATH to the place where the cog is built.
  - Include the cog bin directory into the path.

- Example (sh)
  - set COG_INSTALL_PATH=C:\build\cog-1.0a
  - set PATH=% COG_INSTALL_PATH%\bin

How to Program with the Java and Python CoG Kits

Examples

- Java examples:
  - several basic examples are in
    - jglobus/org/globus/examples
  - several advanced examples are in the appropriate subdirectory
    - e.g. security examples
- Python examples:
  - Basic examples are in
    - pyGlobus/examples
  - Test directories contain the unittest code that provide more advanced examples

Resource Specification Language

- RSL is a common interchange language to describe resources, irrespective of the scheduler or batch system used.

  Supports attributes like

  - executable, working directory, arguments list
  - stdin, stdout, stderr (local files or Gass/FTP URLs)
  - min/max memory, max cpu time, no. of processes
  - Etc.

- Class org.globus.rsl.RslAttributes
  convenient methods for RSL expression creation and manipulation

  RslAttributes rsl = new RslAttributes();

  rsl.add("executable", "/bin/ls");

  String rslStr = rsl.toRSL(); // returned RSL: &(executable=/bin/ls)

  rsl.remove("executable");

Java Job Submission

- Check whether you can submit a job to a particular gatekeeper.

```
Gram.ping( proxy, “hot.mcs.anl.gov”);
```

- Create a job

```
GramJob job = new GramJob(proxy, rsl.toRSL());
```

- Add a status change listener

```
class GramJobListenerImpl implements GramJobListener {
    public void statusChanged(GramJob job) {
            String status = job.getStatusAsString();
    }

job.addListener(new GramJobListenerImpl());
```

- Submit the job to a GRAM resource manager

```
job.request(“hot.mcs.anl.gov”); // default IANA port 2119
```

- Cancel the job, if need be.

```
job.cancel()
```

Java Output Redirection

- Create a GASS server to receive output/error

  For a GASS server on client machine:

  GassServer gass = new GassServer(proxy, port);

  A GASS server can also be started on a remote machine through the gatekeeper.

- Set the GASS URL as stdout/stderr parameter in job RSL. This will stream the job output/error to the GASS server.

  rsl.add("stdout", gass.getURL() + "/dev/stdout ");

  rsl.add("stderr", gass.getURL() + "/dev/stderr");

- Redirect the output/error from GASS server to a JobOutputListener.

```
class JobOutputListenerImpl implements JobOutputListener {
      public void outputChanged(String s) { /* process the output here */}
      public void outputClosed() {}
     };
JobOutputListenerImpl outListener =  new JobOutputListenerImpl();
JobOutputStream outStream = new JobOutputStream( outListener );
gass.registerJobOutputStream( "out", outStream )
gass.registerJobOutputStream( "err", outStream )
```

- Output/errors from different jobs can be handled by a single GASS server.

Python Job Submission Example

- Creating a job.

```
try:
    gramClient = GramClient.GramClient()
    callbackContact = gramClient.set_callback(func, condV)
    jobContact =
    gramClient.submit_request("clipper.lbl.gov",

    "&(executable=/bin/sleep)(argument=15)",
                        GramClient.JOB_STATE_ALL)
except GramClient.GramClientException, ex:
    print ex.msg
```

- Callback for state changes.

```
def func(cv, contact, state, error):
    if state == GramClient.JOB_STATE_PENDING:
        print "Job is pending"
    elif state == GramClient.JOB_STATE_ACTIVE:
        print "Job is active"
```

Redirecting stdout with gramClient

```
from pyGlobus import gassServerEZ
opts = gassServerEZ.STDOUT_ENABLE
server = gassServerEZ.GassServerEZ(opts)
url = server.getURL()
rsl = "&(executable=/bin/sleep)(arguments=15)
                    (stdout=%s/dev/stdout)“ % url
```

Compare: C Job Submission Example

```
callback_func(void *user_arg, char *job_contact,
              int state, int errorcode)
{
    globus_i_globusrun_gram_monitor_t *monitor;
    monitor = (globus_i_globusrun_gram_monitor_t *) user_arg;
    globus_mutex_lock(&monitor->mutex);
    monitor->job_state = state;
    switch(state)
    {
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING:
{
    globus_i_globusrun_gram_monitor_t *monitor;
    monitor = (globus_i_globusrun_gram_monitor_t *) user_arg;
    globus_mutex_lock(&monitor->mutex);
    monitor->job_state = state;
    switch(state)
    {
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED:
        if(monitor->verbose)
        {   globus_libc_printf("GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED\n");
        }
        monitor->done = GLOBUS_TRUE;
        break;
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE:
        if(monitor->verbose)
        {      globus_libc_printf("GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE\n");
        }
        monitor->done = GLOBUS_TRUE;
        break;
 }
  globus_cond_signal(&monitor->cond);
  globus_mutex_unlock(&monitor->mutex);
}
```

Compare: C Job Submission Example (cont.)

```
globus_l_globusrun_gramrun(char * request_string, unsigned long options, char *rm_contact){
    char *callback_contact = GLOBUS_NULL;
    char *job_contact = GLOBUS_NULL;
    globus_i_globusrun_gram_monitor_t monitor;
    int err;
    monitor.done = GLOBUS_FALSE;
    monitor.verbose=verbose;
    globus_mutex_init(&monitor.mutex, GLOBUS_NULL);
    globus_cond_init(&monitor.cond, GLOBUS_NULL);

    err = globus_module_activate(GLOBUS_GRAM_CLIENT_MODULE);
    if(err != GLOBUS_SUCCESS)
    { … }
    err = globus_gram_client_callback_allow(
            globus_l_globusrun_gram_callback_func,
            (void *) &monitor,
            &callback_contact);
        if(err != GLOBUS_SUCCESS)
        { … }
   err = globus_gram_client_job_request(rm_contact,
            request_string, GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL,
             callback_contact, &job_contact);
   if(err != GLOBUS_SUCCESS)
        { … }
 globus_mutex_lock(&monitor.mutex);
    while(!monitor.done)  {
       globus_cond_wait(&monitor.cond, &monitor.mutex);
    }
 globus_mutex_unlock(&monitor.mutex);
    globus_gram_client_callback_disallow(callback_contact);
    globus_free(callback_contact);

    globus_mutex_destroy(&monitor.mutex);
    globus_cond_destroy(&monitor.cond);
```

Personal Gatekeeper

Eg : org.globus.gatekeeper.Gatekeeper

- Running the gatekeeper
  - –Ant : ant -f demos.xml server
  - –Command line :globus-personal-gatekeeper

  (Remember to set the environment variables)

  Use the GRAM contact returned.
- Submitting jobs to this gatekeeper
  - –globusrun -o -r "localhost:2119:/O=Grid/O=Globus/OU=mcs.anl.gov/CN=Gregor von Laszewski" "&(executable=/c:/cygwin/bin/ls.exe)"

Remote File Transfer

- Based on the interface we can move the files in the following ways.
  - GUI Based: File Transfer Component for Grids
  - Non-GUI Based:Command Line tools
  - Using API: For Programmers
- Based on the protocols it supports, we can classify different ways as follow
  - Vanilla FTP: FTP
  - Grid File Transfer Protocol : GridFTP
  - UrlCopy : HTTP, HTTPS, FTP, and FILE
  - Global Access to Secondary Storage-GASS: HTTP, HTTPS

Java UrlCopy: copy from ->to

Eg : org.globus.io.urlcopy.UrlCopyTest

```
import org.globus.io.urlcopy.*
UrlCopy c = new UrlCopy();
```

Setting the urls

```
c.setSourceUrl(from);
c.setDestinationUrl(to);
c.setUseThirdPartyCopy(true);
```

Registering with the listener

```
c. addUrlCopyListener(new UrlCopyListener() {
      public void transfer(int total, int current) {
                  System.out.println(total + " " + current);  }
      public void transferError(Exception e) {
        System.out.println("transfer failed: " + e.getMessage());
      }
     public void transferCompleted() {
        System.out.println("Transfer completed successfully");}
  });
```

Transfering the file

```
c.run();
```

Java GridFTP Client

Basic Functions :

Eg:org.globus.ftp.test.GridFTPClient2PartyTest

```
import org.globus.ftp.*;
GridFTPClient client = new GridFTPClient(host, port);
String fullLocalFile = localDir + "/" + localFile;
String fullRemoteFile = remoteDestDir + "/" + localFile;
client.authenticate(GlobusProxy.getDefaultUserProxy());
client.setProtectionBufferSize(16384);
client.setType(GridFTPSession.TYPE_IMAGE);
client.setMode(GridFTPSession.MODE_EBLOCK);
client.setDataChannelAuthentication(dcau);
client.setDataChannelProtection(prot);
DataSource source= new DataSourceStream(new
FileInputStream(fullLocalFile));
client.put(fullRemoteFile,source,  null);
DataSink  sink = new DataSinkStream(new
FileOutputStream(fullLocalFile));
client.get(fullRemoteFile, sink, null);
```

Python GridFTP Example

```
from pyGlobus import ftpClient
from pyGlobus.util import Buffer
handleAttr = ftpClient.HandleAttr()
opAttr = ftpClient.OperationAttr()
marker = ftpClient.RestartMarker()
ftpClnt = ftpClient.FtpClient(handleAttr)
ftpClnt.get(url, opAttr, marker, done_func, condV)
buf = Buffer(64*1024)
handle = ftpClnt.register_read(buf, data_func, 0)

def data_func(cv, handle, buffer, bufHandle, bufLen, offset,
    eof, error):
    g_dest.write(buffer)
    if not eof:
        try:
            handle = g_ftpClient.register_read(g_buffer,
    data_func, 0)
        except Exception, e:
```

Performance Options for GridFTP

- Setting tcpbuffer size

```
from pyGlobus import ftpControl
battr = ftpControl.TcpBuffer()
battr.set_fixed(64*1024)
```

Or

```
battr.set_automatic(16*1024, 8*1024, 64*1024)
opAttr.set_tcp_buffer(battr)
```

- Setting parallelism

```
para = ftpControl.Parallelism()
para.set_mode(ftpControl.PARALLELISM_FIXED)
para.set_size(3)
opAttr.set_parallelism(para)
```

Python GassCopy

- Provides a protocol independent API to transfer remote files.

```
srcAttr        = GassCopyAttr()
handleAttr     = GassCopyHandleAttr()
destAttr       = GassCopyAttr()
ftpSrcAttr     = FtpOperationAttr()
ftpDestAttr    = FtpOperationAttr()
srcAttr.set_ftp(ftpSrcAttr)
destAttr.set_ftp(ftpDestAttr)
copy = GassCopy(handleAttr)
copy.copy_url_to_url(srcUrl, srcAttr, destUrl,  destAttr)
```

Java Web Based Portal

Features

- No installation needed on user's side. Only a browser is required.
- On the portal server, simple installation provided for Grid job portlets using Apache Ant (http://jakarta.apache.org/ant).
- Portlets allow the user to submit interactive/batch jobs to Globus gatekeepers, monitor their status (active, pending etc.) and view the output/errors. Access to the Grid is based on GSI security.

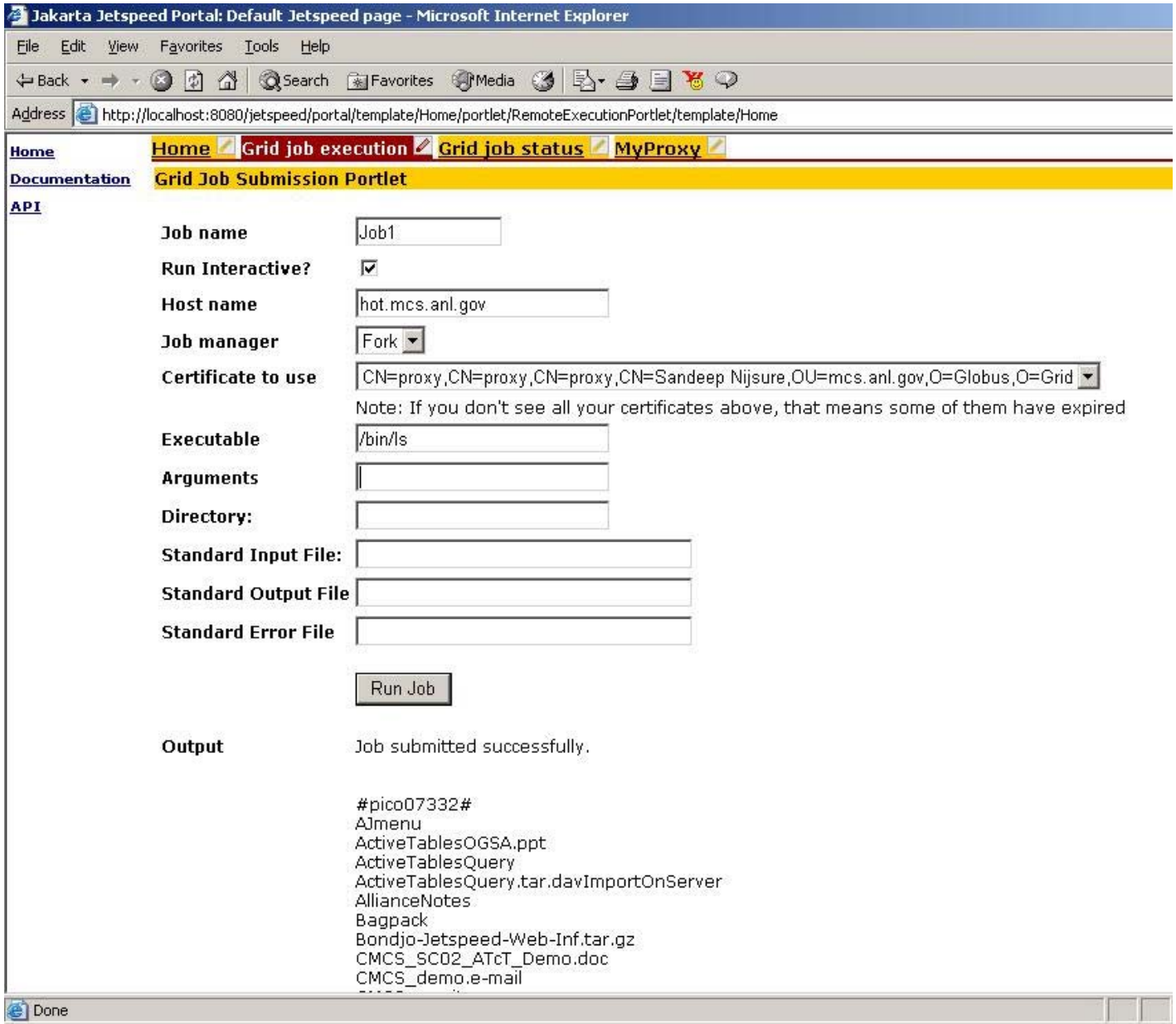
Software needed to run a Jetspeed portal server:

- A Java Servlet Engine/Container. Must be compatible with the Servlet 2.2 or Servlet 2.3 API. For example, Tomcat (http://jakarta.apache.org/tomcat)
- JDK 1.3 or higher Java Virtual Machine.

Requirements for Grid job portlets:

- Java CoG
- A Myproxy server on which delegated user credentials are stored. This is needed for GSI authentication.
- MyProxy credential access portlet developed by Indiana University Extreme! Computing Lab.

Jetspeed Portlet for Job Submission



Future Plans

- Java CoG Kit
  - is being heavily used in GT3
  - forms the basis of the new IBM Grid Application Framework for Java (GAF4J)
    - http://www.alphaworks.ibm.com/tech/gaf4j?Open&ca=daw-hp-pr
  - We are developing an apache Jetspeed based Portal toolkit for Grids.
  - Will be distributed as part of GT3 and separately
- Python Cog Kit
  - Will provide the underlying security implementation for the Python OGSI environment
  - continue to support our high-level interfaces for GT3

Acknowledgement

- The Java CoG Effort is part of the Globus Project
- The Java and Python CoG Kits are funded by the U.S. Department of Energy Office of Science
- Some Java CoG work is supported by NSF/Alliance

- More information can be found at
  - http://www.cogkits.org
  - http://www.globus.org/cog
  - http://www-itg.lbl.gov/gtg/projects/pyGlobus/
- Email:
  - krjackson@lbl.gov
  - gregor@mcs.anl.gov