The Integration of AJAX, Interactive X Windows Applications and Application Input Generation into the UCLA Grid Portal

Joan Slottow
Academic Technology Services
University of California Los Angeles
Los Angeles, CA 90095
1-310-825-7418
joan@ucla.edu

Prakashan Korambath
Academic Technology Services
University of California Los Angeles
Los Angeles, CA 90095
1-310-825-7422
ppk@ucla.edu

Kejian Jin
Academic Technology Services
University of California Los Angeles
Los Angeles, CA 90095
1-310-206-9605
kjin@cs.ucla.edu

ABSTRACT

The University of California (UC) has adopted the UCLA Grid Portal (UGP), a Globus Incubator project, to create grids of computational clusters [1] at its campuses. Campus grids can also be accessed system wide via the UC Grid Portal. Developed as a set of portlets for the GridSphere Portal Framework, parts of UGP, such as those for data management and job submission, have been rewritten using AJAX technologies. In this paper we will discuss the latest additions to UGP. These include: the various UGP services that were programmed using AJAX technologies, the ability to run interactive applications such as Matlab™ and Mathematica™ through the portal web interface, and appForm, our interface that allows portal administrators to easily create application input forms for users. appForm makes extensive use of Web 2.0 technologies.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services

General Terms

Design

Keywords

Grid, Grid Portal, Web Portal, AJAX, Web 2.0, VNC, Interactive, Batch, Data Management, Job Submission, Port Forwarding

1. INTRODUCTION

UGP provides a common way of working with computational clusters in a grid. UGP enables secure, unified, access to computational resources. Because it handles all grid related operations for users, UGP eliminates any requirement that users install and use client-side grid software or certificates and lets the users concentrate on their research.

Slottow J., Korambath P., Jin K. The Integration of AJAX, Interactive X Windows Applications and Application Input Generation into the UCLA Grid Portal, *International Workshop on Grid Computing Environments 2007, Nov., Day, 2007, Reno, NV, USA. An electronic version of this document will be available at http://casci.rit.edu/proceedings/gce2007*

A grid portal powered by UGP is currently deployed system wide at the University of California [2] and at three University of California campuses. Several additional campuses are in various stages of grid implementation. The system-wide UC Grid Portal allows users to access resources at the different campuses from a single grid portal. Two UGP Web Services, following the Web Services Resource Framework (WSRF), are run at the UC Grid Portal. These web services are used for user registration and data synchronization among the grid portals.

UGP is implemented as a set of Java portlets based on the GridSphere Portlet Framework [3]. In addition, appropriate Asynchronous JavaScript and XML (AJAX) toolkits are used to provide client side interactivity. UGP makes use entirely of open source software components. UGP is itself open source.

Services provided by UGP include: resource discovery, data management, batch job submittal/management, the ability for users to use interactive GUI applications, and a Grid Development Environment (GDE) for programmers. Two Data Managers are currently available: an HTML style Web 1.0 Data Manager and an AJAX Data Manager. A new tool named appForm, to be discussed in depth in this paper, makes it easy to build input forms for popular applications so that users no longer need to know how to create application specific input files. appForm forms are easy to define via XML and appForm uses XSLT [4] to generate both the HTML input forms for the web interface and the application input file for the program.

2. ARCHITECTURE

2.1 Single-Campus Architecture

The grid portal at a single campus provides a single interface to all of the clusters at that campus that are participating in the campus grid. UGP provides a single login for users and hides user interface and scheduler differences among clusters making it easy to work with multiple clusters at once. UGP runs a certificate authority (SimpleCA) to issue user certificates. Certificates issued by UGP are only for use by the grid portal: user portal sign on, job submission, cluster access, etc. and have no other campus purpose.

Individual university professors, groups and departments, own and maintain control of the clusters, and local administrative procedures and security solutions take precedence. Each of the participating clusters is attached to a grid portal via a grid appliance. The addition of a grid appliance to a cluster in no way modifies policy decisions at the cluster level. Any participating cluster can always also be used directly by users who login to the cluster head node, without going through the grid portal.

The UCLA grid architecture for a single institution is depicted in Figure 1.

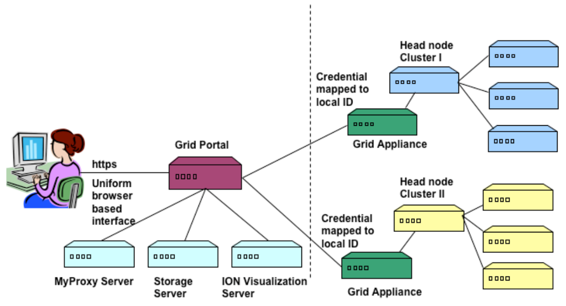


Figure 1. Single Campus Architecture

The grid portal is accessed from a web browser. Each grid appliance connects a cluster to the grid portal and acts as alternative job submission host for that cluster. To facilitate data management, the file system containing the user's home directories on a cluster is cross-mounted on its grid appliance. For security, each grid appliance runs a firewall (depicted by the dashed line) and is open only to the grid portal machine. All transactions between the Portal and the Appliances use public key cryptography conforming to the X-509 certificate standard [5]. Additional servers that can be part of the grid portal include a MyProxy server for certificates, a storage server to provide disk storage for those users who do not have their own cluster accounts, and an ION Visualization server if the commercial product IDL-ION™ is used to provide data visualization.

2.2 University of California Architecture

At the University of California, all computational resources reside at the campus level. Consequently, there is a campus grid at each campus and grid portal at each campus to provide access to the campus grid. An additional grid portal for the UC provides access to all computational resources system wide.

Figure 2 depicts the hierarchical architecture deployed for the University of California. It shows the campus grids for three campuses and their relationship to the UC Grid Portal.

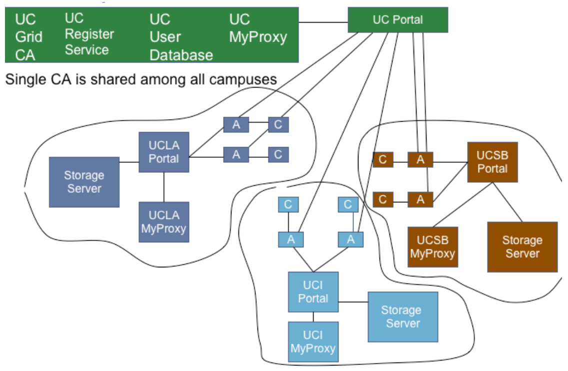


Figure 2. Multi-Campus Architecture ("A" stands for grid appliance; "C" stands for cluster)

The campus grids used in the multi-campus architecture are different from grids constructed when there is only single campus grid in that a certificate authority (CA) is not run at the campus level. A single CA for all the grids is included as part of the UC Grid portal and a special web service, the UC Register Service, is also run at the UC grid portal, with clients at each campus, to make sure the certificates and usernames are unique across the campus grids. This design provides for a streamlined new user application process. Each new user of a campus grid portal is automatically given access to the UC Grid Portal as part of the workflow that creates the user's grid certificate.

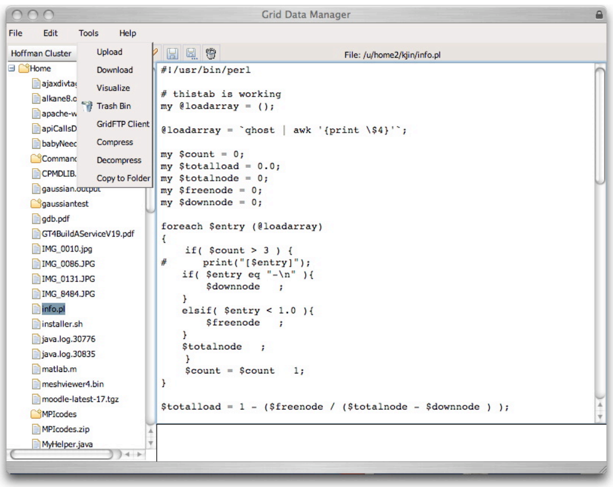
In addition to the UC Register Service, the UC Sync Service, also run at the UC Grid Portal, updates the information in the UC Grid Portal's databases about clusters and their applications whenever a change is made to this same information on one of the campus grid portals.

3. AJAX USER INTERFACES IN UGP

AJAX is a technique used for creating web applications that makes them more interactive. It makes use of today's JavaScript-enabled web browsers that support asynchronous XMLHttpRequests and JavaScript. AJAX enabled websites normally make use of JavaScript libraries or toolkits, that include widget sets and APIs that handle events and asynchronous requests on the client side. Java code, run on the server, processes the requests. Users experience real performance benefits because only a small part of the user interface is updated with each request.

3.1 AJAX Data Manager

The AJAX Data Manager is written using the Zimbra AjaxTK[6] which has a very rich widget set. In writing the AJAX Data Manager our design principle was to make data management as intuitive as possible. With traditional web technology data management is cumbersome, as the user has to click on something and wait for a page to be refreshed numerous times just to accomplish a simple task. With the AJAX Data Manager the interface is extremely interactive and is very similar to a desktop data manager in appearance. As shown in Figure 3, at the left side, there is a directory tree. A menu appears at the top. There is even an icon bar with icons for the most commonly used activities. The AJAX Data manager's GridFTP client, Figure 4, is formatted with a column for each side of the FTP transaction, just like a GUI desktop FTP client. A progress bar, Figure 5, is displayed when a file is uploaded.



```
#!/usr/bin/perl

# thistab is working
my @loadarray = ();

@loadarray = `qhost | awk '{print \$4}'`;

my $count = 0;
my $totalload = 0.0;
my $totalnode = 0;
my $freenode = 0;
my $downnode = 0;

foreach $entry (@loadarray)
{
    if( $count > 3 ) {
#       print("[$entry]");
   if( $entry eq "-\n" ){
       $downnode   ;
   }
   elsif( $entry < 1.0 ){
       $freenode   ;
   }
   $totalnode   ;
    }
    $count = $count   1;
}

$totalload = 1 - ($freenode / ($totalnode - $downnode ) );
```

Figure 3. AJAX Data Manager

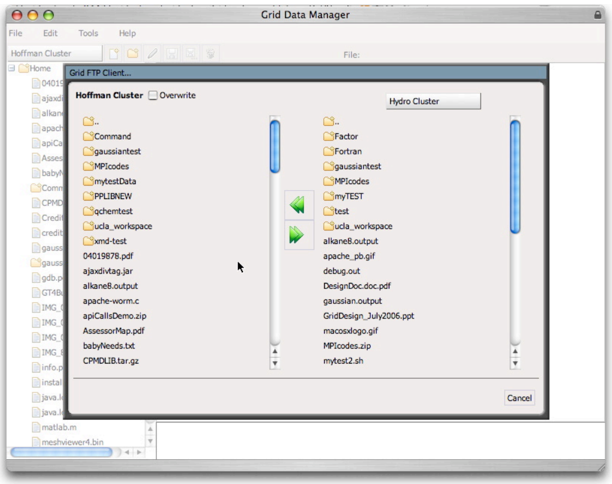
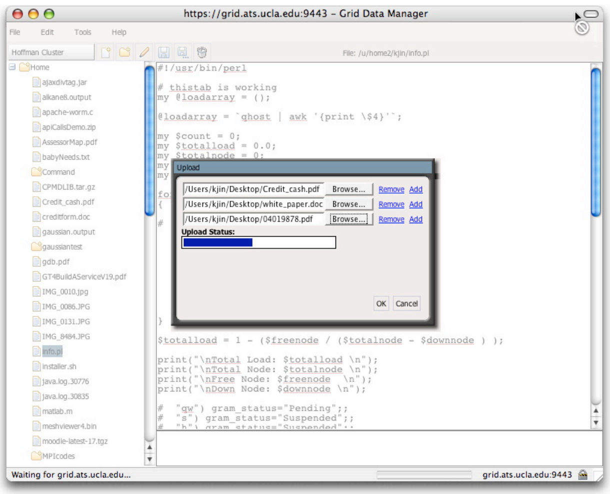


Figure 4. AJAX Data Manager – GridFTP Client


https://grid.ats.ucla.edu:9443 – Grid Data Manager
File
Edit
Tools
Help
Hoffman Cluster
File: /u/home2/kjin/info.pl
Upload
Browse...
Remove
Add
Upload Status:
OK
Cancel
Waiting for grid.ats.ucla.edu...
grid.ats.ucla.edu:9443

/Users/kjin/Desktop/Credit_cash.pdf
/Users/kjin/Desktop/white_paper.doc
/Users/kjin/Desktop/04019878.pdf

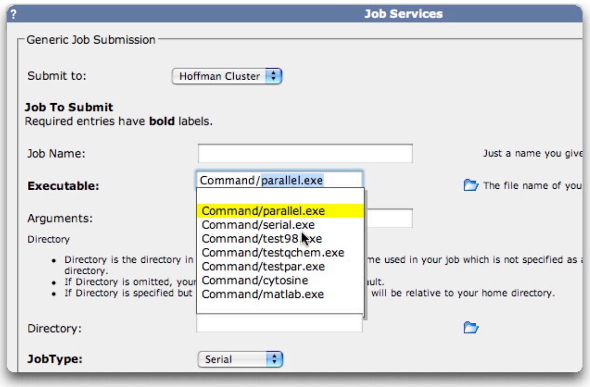
```
$totalload = 1 - ($freenode / ($totalnode - $downnode ) );

print("\nTotal Load: $totalload \n");
print("\nTotal Node: $totalnode \n");
print("\nFree Node: $freenode  \n");
print("\nDown Node: $downnode \n");

#  "qw") gram_status="Pending";;
#  "s") gram_status="Suspended";;
```

Figure 5. AJAX Data Manager – Upload Progress Bar

3.2 AJAX in Job Services

UGP Job Services enable users to submit batch jobs and view job status and results. To submit a job, a user must fill out a form specifying job requirements, input files, etc. UGP knows how to run commonly used applications so that the details of how to run them don't have to be known by the user. For a user-provided program the user must provide a path to the executable.

As shown in Figure 6, we have augmented the UGP Generic Job Submission page with the auto-complete feature from Yahoo User Interface (YUI)[7]. In addition to providing a convenient file name completion for the user, this allows UGP to check whether the executable exists as the user enters the executable's path. Eliminating errors in job submission has a large impact because of the wait time in the queues before job startup; finding a problem immediately literally saves hours or days.



Submit to: Hoffman Cluster

Job To Submit
Required entries have **bold** labels.

Job Name: Just a name you give

Executable: Command/parallel.exe The file name of you

Command/parallel.exe
Command/serial.exe
Command/test98.exe
Command/testqchem.exe
Command/testpar.exe
Command/cytosine
Command/matlab.exe

Arguments:

Directory

- Directory is the directory in … me used in your job which is not specified as … directory.
- If Directory is omitted, your … ult.
- If Directory is specified but … will be relative to your home directory.

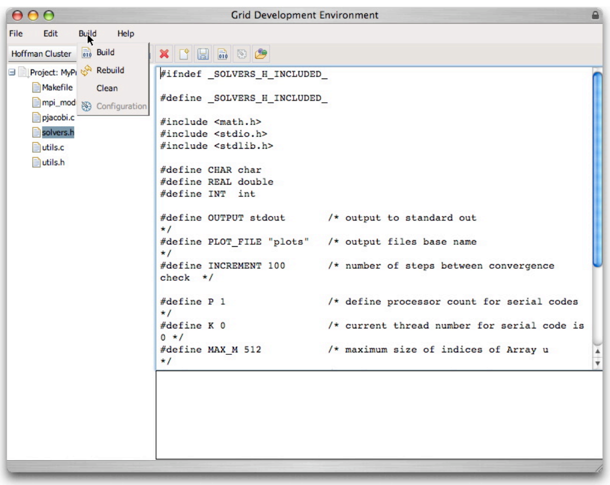
Directory:

JobType: Serial

Figure 6. Auto-completion

3.3 AJAX Grid Development Environment

The Grid Development Environment (GDE) was also written using the Zimbra AjaxTK and benefits from its rich widget set. Our goal in writing GDE was to enable code compilation from the grid portal in the heterogeneous environment of the grid. Code compilation is normally done interactively rather than via batch, and a program has to be compiled on a machine of the same architecture as the one on which it will be run. The UCLA Grid currently has clusters of diverse architectures and the architecture of the grid appliance normally differs from that of the cluster to which it is attached. Therefore, the GDE submits code to an instantaneous queue on the target cluster for compilation on a compute node of the correct architecture. An expert user can alternatively compile code on the cluster head node by getting an xterm or by ssh, both interactive services provided by UGP.



```
#ifndef _SOLVERS_H_INCLUDED_

#define _SOLVERS_H_INCLUDED_

#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define CHAR char
#define REAL double
#define INT  int

#define OUTPUT stdout       /* output to standard out
*/
#define PLOT_FILE "plots"   /* output files base name
*/
#define INCREMENT 100       /* number of steps between convergence
check  */

#define P 1                 /* define processor count for serial codes
*/
#define K 0                 /* current thread number for serial code is
0 */
#define MAX_M 512           /* maximum size of indices of Array u
*/
```

Figure 7. Grid Development Environment

4. Providing Interactive X Windows-based GUI Applications on the Web Through the use of VNC

The use of Virtual Network Computing (VNC) for interactive applications was spearheaded at other grid portals most notably: Purdue University's nanoHUB [8], University of Florida's In-VIGO [9] and University of Texas Advanced Computer Center (TACC) [10] prior to its adoption by UGP. Figure 8 shows the design used by UGP, which enables the VNC Viewer applet to communicate with a VNC Server running on a compute node.

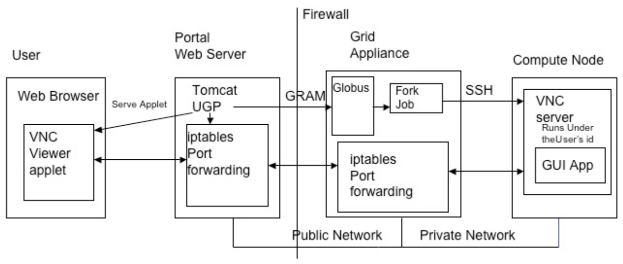


Figure 8. VNC Implementation

Figure 8 is a one-to-many diagram. Figure 9 shows an example of this. Many grid appliances are connected to a single grid portal. Each cluster can have multiple compute nodes that run interactive applications and each compute node can run multiple simultaneous interactive applications. UGP serves the VNC Viewer applet to the user's web browser and because the VNC Viewer is an applet, it can only communicate with the Portal. (For security purposes, an applet can only communicate with the machine that served it.) Therefore, iptables port forwarding has to be used, both on the grid portal machine and on the grid appliances to pass the communication back and forth between the VNC Viewer applet and the VNC server running on a compute node in a private subnet. We give complete instructions on our UGP Wiki [11] for setting up iptables.

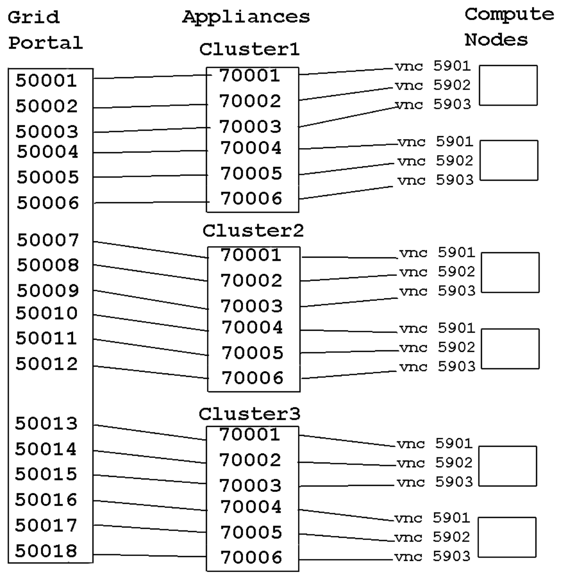


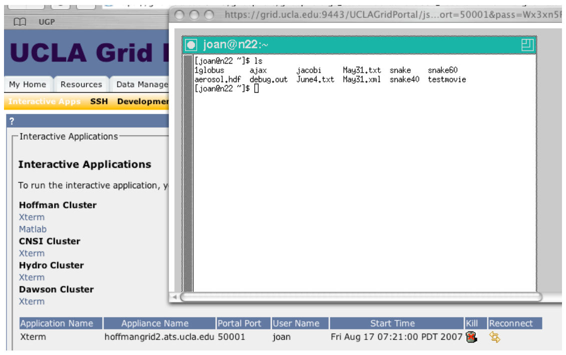
Figure 9. Port Map for VNC

A table inside of UGP keeps track of this port mapping. According to Figure 9, the grid portal's port 50001 maps to port 70001 on the grid appliance named cluserl and port 5901 on the VNC server running on the first compute node connected to the appliance. Other VNC servers can also use the same port 5901 but these are running on different compute nodes on the same or different clusters.

When the user wants to run an interactive application UGP picks a compute node and runs the VNC server there as the user, generating a random VNC password and creating all necessary VNC startup files. UGP starts the VNC client applet in the user's web browser, passing it the VNC password it just generated and the grid portal address and port that it is to use to communicate with the VNC Server. The user is unaware that all this is happening in the background and just sees a window opening in the browser containing the X Windows application.

Note that the VNC Server does not come with the OpenGL X-Windows extension (GLX) enabled in versions of Fedora prior to 7.0. GLX enabling is required to run OpenGL applications through VNC.

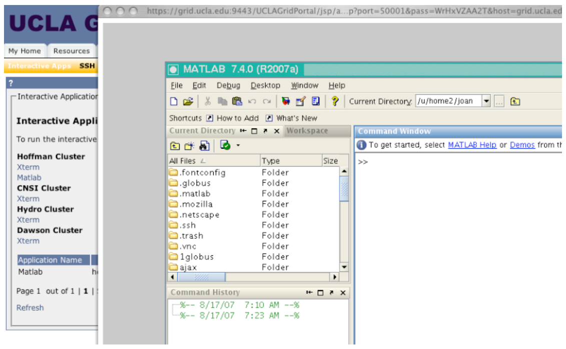
Figure 10 shows UGP's Interactive Apps page. A list of all the interactive applications available on each cluster in the user's cluster access list appears at left. Here the user has opened an xterm, which opens in a separate pop-up browser window. The information about the VNC session running the xterm appears in the bottom line with purge and reconnect icons. UGP allows each user to run an administrator configurable number of simultaneous interactive applications and to close the web browser, for example at work, and connect later from the same or a different browser to continue working. This enables long-running "interactive" problems to continue unattended. Figure 11 shows Matlab™ running through VNC.



```
[joan@n22 ~]$ ls
1globus     ajax       jacobi     May31.txt  snake    snake60
aerosol.hdf debug.out  June4.txt  May31.xml  snake40  testmovie
[joan@n22 ~]$ 
```

Application Name	Appliance Name	Portal Port	User Name	Start Time	Kill	Reconnect
Xterm	hoffmangrid2.ats.ucla.edu	50001	joan	Fri Aug 17 07:21:00 PDT 2007		

Figure 10. Running xterm through VNC on UGP



All Files	Type	Size
.fontconfig	Folder	
.globus	Folder	
.matlab	Folder	
.mozilla	Folder	
.netscape	Folder	
.ssh	Folder	
.trash	Folder	
.vnc	Folder	
1globus	Folder	
ajax	Folder	

```
>>
```

```
%-- 8/17/07   7:10 AM --%
%-- 8/17/07   7:23 AM --%
```

Figure 11. Running Matlab™ through VNC on UGP

5. appForm: Interactive Application Input File Generation Forms with a Web Interface

appForm is an application that generates both a web browser input form and a program input file for an application. The form is described with an easy-to-use set of XML elements. appForm makes heavy use of XSLT, JavaScript, and YUI.

Commercial application programs such as Q-Chem™, and Gaussian™ that are commonly run on the clusters as batch jobs have complex, application-specific input files that require an expert user to create. Many custom university discipline-specific applications are similar in this respect. It would be easier for casual users of these applications to be able to fill in forms on the web and have the input files generated for them. In writing appForm our goal was to do just that in the grid portal environment of UGP.

5.1 Previous Work

In developing appForm we were inspired by Purdue University's Rappture [12] and the San Diego Supercomputer Center (SDSC) and National Biomedical Computation Resource (NBCR) Project Gemstone [13].

Rappture is a toolkit that can create a GUI interface for any application. It includes an XML description language for a set of Rappture widgets and it creates a GUI for the application using TK/TCL. A set of language bindings for C/C++, Fortran, Matlab™, Perl, Python, etc. allow the application to be programmed to read the data placed into the GUI by the user and to communicate the application output back to the GUI for display to the user. A Perl, Python or TCL wrapper can be written for any binary-only or legacy application to create an application input file from the data entered in the GUI and to parse the application output file for transmission back to the GUI. On the nanoHUB, Rappture is used to turn many non-interactive applications into interactive applications, which the nanoHUB runs under VNC. Though we liked the XML widget descriptions defined by Rappture, our goal was to have GUI input forms solely for purpose of creating program input for batch jobs that are to be submitted and queued and which might run for days or weeks. Furthermore, we wanted the application input forms to be HTML so that they could be integrated with UGP.

Gemstone requires that the user download a Gemstone client to run on his/her local machine. A Gemstone application developer has to write JavaScript code using the Gemstone API for each application to be supported. The Gemstone client interacts with a Web Service such as Opal that can interact with a scheduler to launch jobs, query job status, retrieve job output, etc. We liked the Gemstone user interface but 1) didn't want the users to have to download and run a client, 2) already had the features of the Web Service built into UGP, and 3) wanted it to be easy to add forms for additional applications. We therefore created appForm based loosely on the Rappture widget XML.

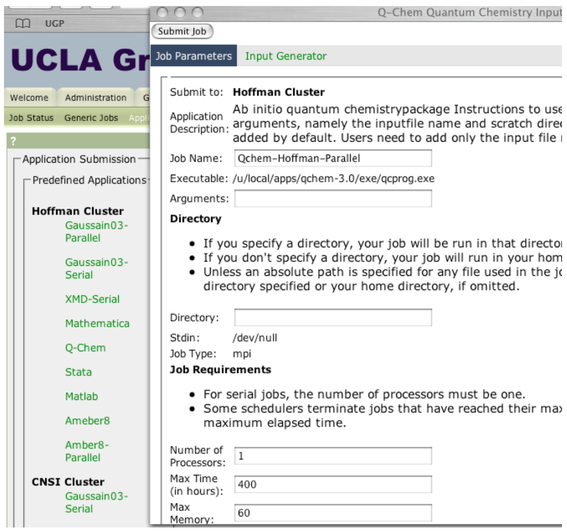
5.2 How appForm Works

appForm has the following parts: 1) a documented set of XML elements for creating widgets, 2) an XSLT translator for translating the XML to the corresponding HTML for the form and 3) all related stylesheets and JavaScript code, including YUI code, for the user's interaction with elements in the form; and Java servlets that provide server-side work such as saving files to the clusters and submitting jobs. The XSLT translator is generic for the widget set and is used for translating from XML to an HTML form for all applications.

A form for a new application can be added by: 1) describing the widgets to be placed in the form via XML, and 2) writing a small XSLT translator from: a simple XML file that gives the name and value of every data element the user has entered to: the input format required by the application. These translators are usually simple and can be based on existing translators for other applications.

When the user selects an application, appForm opens in a new window. As shown in the figures, each input form has two tabs:

Job Parameters (Figure 12) and Input Generator. The Job Parameters result from a separate set of XML that allow the Globus GRAM RSL job parameters to be customized for each application. The RSL and appForm XML files are each translated and combined into a single YUI tabbed view. The Input Generator has an additional tabbed view of its own (seen in Figures 13 and 14). What appears to be multiple pages is actually one very large web page most of which is hidden from view. The user can go back and forth between the tabs without loosing any data values that have either been selected in pull-down menus or input in text fields. Data would be lost between pages if individually linked HTML pages had been used.



Submit to: **Hoffman Cluster**
Application Description: Ab initio quantum chemistrypackage Instructions to use arguments, namely the inputfile name and scratch dire added by default. Users need to add only the input file
Job Name: Qchem-Hoffman-Parallel
Executable: /u/local/apps/qchem-3.0/exe/qcprog.exe
Arguments:
Directory

- If you specify a directory, your job will be run in that directo
- If you don't specify a directory, your job will run in your hom
- Unless an absolute path is specified for any file used in the jo directory specified or your home directory, if omitted.

Directory:
Stdin: /dev/null
Job Type: mpi
Job Requirements

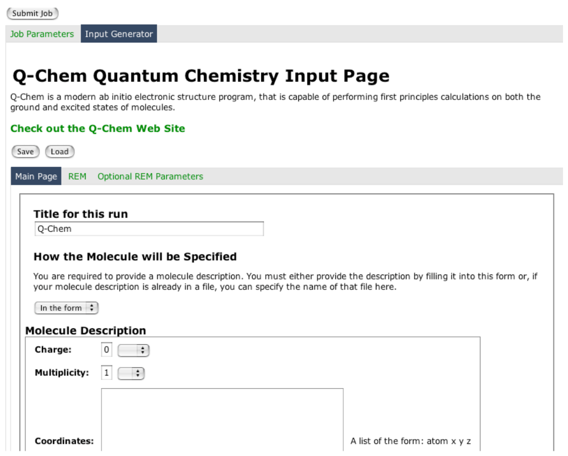
- For serial jobs, the number of processors must be one.
- Some schedulers terminate jobs that have reached their ma maximum elapsed time.

Number of Processors: 1
Max Time (in hours): 400
Max Memory: 60

Figure 12. The Job Parameters "page"

The Submit Job button is on top of every tabbed view. The Save and Load buttons appear near the top of every Input Generator tabbed view. When the user presses the Save button, he/she is presented with a Save dialog to enter a file name. This results in 1) an XML version of the form data being saved in a file on the cluster, 2) invocation of the XSLT translator for the application being used to save the input file for the application and 3) the input file name being filled in automatically on the Job Parameters tabbed view so the job can be submitted. The Load button can be pressed if the user wants to populate the form from a previously saved set of data values.

Figures 13 and 14 show the same part of a form that prepares input for the chemistry program Q-Chem™. Here the user can choose to supply the molecule description in the form or to read it from a file and the form provides for both options. The user selects "In the form" or "From a File" from the pull-down menu under the heading "How the Molecule will be Specified" and because Dynamic HTML (DHTML) is used, the change to the form happens as soon as the use makes the selection.



Q-Chem Quantum Chemistry Input Page

Q-Chem is a modern ab initio electronic structure program, that is capable of performing first principles calculations on both the ground and excited states of molecules.

Title for this run
Q-Chem

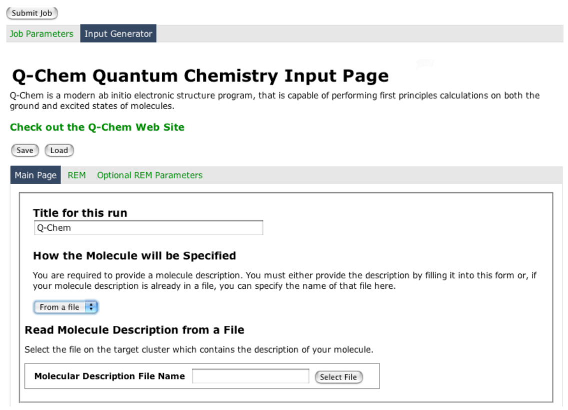
How the Molecule will be Specified

You are required to provide a molecule description. You must either provide the description by filling it into this form or, if your molecule description is already in a file, you can specify the name of that file here.

In the form

Molecule Description
Charge: 0
Multiplicity: 1
Coordinates: A list of the form: atom x y z

Figure 13. Q-Chem Input Generator Main Page --With Molecule Specified in the Form



Q-Chem Quantum Chemistry Input Page

Q-Chem is a modern ab initio electronic structure program, that is capable of performing first principles calculations on both the ground and excited states of molecules.

Title for this run
Q-Chem

How the Molecule will be Specified

You are required to provide a molecule description. You must either provide the description by filling it into this form or, if your molecule description is already in a file, you can specify the name of that file here.

From a file

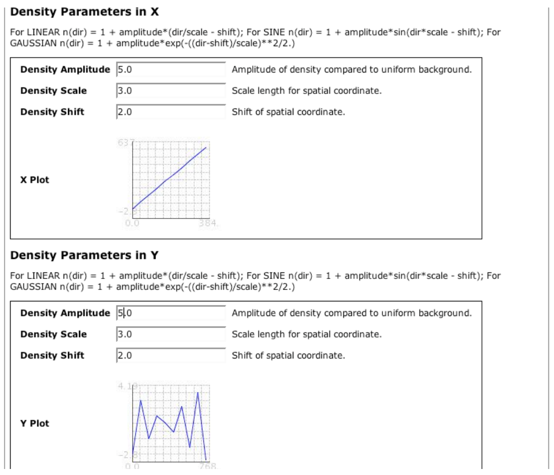
Read Molecule Description from a File

Select the file on the target cluster which contains the description of your molecule.

Molecular Description File Name

Figure 14. Q-Chem Input Generator Main Page --With Molecule Input from a File

If desired for visual feedback, graphs that correspond to the input values selected by the user can be included in the form. An example of this is shown in Figure 15, in which input is being prepared for a physics code. In that example, when any of the values for Density Amplitude, Density Scale or Density Shift are changed, the accompanying graph is changed correspondingly. The graphs are generated by a servlet that was written using the open source jmathplot [14] API and updated dynamically.



Density Parameters in X

For LINEAR n(dir) = 1 + amplitude*(dir/scale - shift); For SINE n(dir) = 1 + amplitude*sin(dir*scale - shift); For GAUSSIAN n(dir) = 1 + amplitude*exp(-((dir-shift)/scale)**2/2.)

Density Amplitude	5.0	Amplitude of density compared to uniform background.
Density Scale	3.0	Scale length for spatial coordinate.
Density Shift	2.0	Shift of spatial coordinate.
X Plot		

Density Parameters in Y

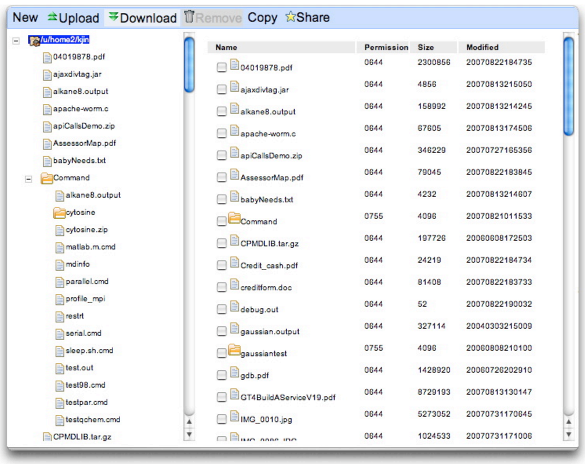
For LINEAR n(dir) = 1 + amplitude*(dir/scale - shift); For SINE n(dir) = 1 + amplitude*sin(dir*scale - shift); For GAUSSIAN n(dir) = 1 + amplitude*exp(-((dir-shift)/scale)**2/2.)

Density Amplitude	5.0	Amplitude of density compared to uniform background.
Density Scale	3.0	Scale length for spatial coordinate.
Density Shift	2.0	Shift of spatial coordinate.
Y Plot		

Figure 15. Graph can be Generated from User Input

6. Future Work

We are currently in the process of updating UGP from portlets for GridSphere 2 to portlets for the GridSphere 3 framework. As part of this work we are redoing parts of UGP. Because of its excellent performance, we have recently begun rewriting the AJAX Data Manager using Google Web Toolkit (GWT) [15]. At the heart of GWT there is a Java to JavaScript compiler that allows all code to be written and debugged in Java. This approach improves performance by minimizing the amount of data that needs to be transmitted to the browser. Figure 16 shows our new GWT Data Manager.



```
/u/home2/kjin
    04019878.pdf
    ajaxdivtag.jar
    alkane8.output
    apache-worm.c
    apiCallsDemo.zip
    AssessorMap.pdf
    babyNeeds.txt
    Command
        alkane8.output
        cytosine
        cytosine.zip
        matlab.m.cmd
        mdinfo
        parallel.cmd
        profile_mpi
        restrt
        serial.cmd
        sleep.sh.cmd
        test.out
        test98.cmd
        testpar.cmd
        testqchem.cmd
    CPMDLIB.tar.gz
```

Name	Permission	Size	Modified
04019878.pdf	0644	2300856	20070822184735
ajaxdivtag.jar	0644	4856	20070813215050
alkane8.output	0644	158992	20070813214245
apache-worm.c	0644	67605	20070813174506
apiCallsDemo.zip	0644	346229	20070727165356
AssessorMap.pdf	0644	79045	20070822183845
babyNeeds.txt	0644	4232	20070813214607
Command	0755	4096	20070821011533
CPMDLIB.tar.gz	0644	197726	20060608172503
Credit_cash.pdf	0644	24219	20070822184734
creditform.doc	0644	81408	20070822183733
debug.out	0644	52	20070822190032
gaussian.output	0644	327114	20040303215009
gaussiantest	0755	4096	20060808210100
gdb.pdf	0644	1428920	20060726202910
GT4BuildAServiceV19.pdf	0644	8729193	20070813130147
IMG_0010.jpg	0644	5273052	20070731170645
[illegible]	0644	1024533	20070731171006

Figure 16. Data Manager using Google Web Toolkit

The grid portal of the future will be the interface of choice for High Performance Computing. One challenge is to attract expert users, who are used to command line interfaces, to the portal. To do so, portal interfaces have to be as rich, flexible and easy to use as Unix commands are today. Portals also have to provide new services that are currently either unavailable or difficult for researchers to access outside of portal environments. To that end, we are continually improving UGP. We are planning to add services in the near future that will enable collaboration among researchers across multiple organizations, and provide the capability of sharing data between the users of the UC grid portals, in a secure fashion. AJAX and related Web 2.0 technologies definitely have a place in our future work as they can be used to create intuitive user interfaces on the web.

7. ACKNOWLEDGMENTS

We are indebted to the In-VIGO developers Renato Figueiredo, Mauricio Tsugawa and Jose Fortes and to James Kohl of Oak Ridge National Laboratory whom we consulted as part of the integration of interactive applications with UGP.

8. REFERENCES

[1] High-performance computing – Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/High-performance_computing [28 August 2007].

[2] UC Grid Portal. http://portal.ucgrid.org [28 August 2007].

[3] Novotny J, Russell M, Wehrens O. 2004. GridSphere: a portal framework for building collaborations. Concurrency and Computation: Practice and Experience 16, 5 (Mar. 2004), 503-513. DOI = 10.1002/cpe.829.

[4] XSL Transformations (XSLT) Version 1.0. W3C Recommendation. 1999.

[5] Adams C, Farrell S, Kause T, Mononen T. Internet X.509 Public Key Infrastructure: Certificate Management Protocol (CMP). The Internet Society. 2005.

[6] Ajax toolkit download for Zimbra open source collaboration software. http://www.zimbra.com/community/kabuki_ajax_toolkit_download.html [28 August 2007].

[7] The Yahoo! User Interface Library (YUI). http://developer.yahoo.com/yui [28 August 2007].

[8] McLennan M, Kennell R, Ebert D, Klimeck G, Qiao W. 2006. Hub-based Simulation and Graphics Hardware Accelerated Visualization for Nanotechnology Applications. IEEE Transactions on Visualization and Computer Graphics 2006; 12,5 (Sep. 2006): 1061-1068. DOI = 10.1109/TVCG.2006.150.

[9] Adabala S, Matsunaga A, Tsugawa M, Figueiredo R, Fortes AB. Single Sign-On in In-VIGO: Role-based Access via Delegation Mechanisms Using Short-lived User Identities. Parallel and Distributed Processing Symposium, Proceedings. April 2004; 22-30.

[10] TACC > Texas Advanced Computing Center. http://www.tacc.utexas.edu [28 August 2007].

[11] How to Setup Interactive Applications – UGP-Wiki. http://www.ucgrid.org/wiki/index.php/How_to_setup_Interactive_Applications [28 August 2007].

[12] Rappture – Trac. https://developer.nanohub.org/projects/rappture [29 August 2007].

[13] Mozdev.org – gemstone: index. http://gemstone.mozdev.org [29 August 2007].

[14] SourceForge.net: jmathplot. http://sourceforge.net/projects/jmathplot [29 August 2007].

[15] Google Web Toolkit – Build AJAX apps in the Java language. http://code.google.com/webtoolkit [28 August 2007].

[16] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. Acm Press Frontier Series. ACM Press, New York, NY, 19-33. DOI= http://doi.acm.org/10.1145/90417.90738