The ASCI Computational Grid: Initial Deployment

Randal Rheinheimer
Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545
randal@lanl.gov

Judy I. Beiriger, Hugh P. Bivens, Steven L. Humphreys
Sandia National Laboratories[1], P. O. Box 5800, Albuquerque, NM 87185-1137
{jibeiri, hpbiven, slhumph}@sandia.gov

1.   Overview

The Accelerated Strategic Computing Initiative (ASCI) Grid Services project began in 1999 with a mandate to provide a common interface to the disparate resources of the ASCI tri-lab (Los Alamos National Laboratory, Sandia National Laboratories, and Lawrence Livermore National Laboratory) weapons complex. This computational grid is currently deployed in the secure network of the tri-lab complex. It is also deployed in the open protected networks of the laboratories, but current firewall policies necessitate that the "grid" in the open protected network be a set of localized site grids; there is no current grid connectivity between sites in that network. In the secure network, the grid consists of a handful of very large SMPs, some of which have dedicated visualization and IO nodes contained within them. There are also standalone visualization clusters of various architectures and three High Performance Storage Systems (HPSS[1]) within the computing network. The systems are geographically widely distributed, but are connected by four stripes of OC-12 bandwidth. In total, the grid comprises about 21 Tflops of processing and 3 Petabytes of storage. The user community, however, is small by grid standards, with only a few analysts accounting for a large percentage of computing cycles and storage bandwidth. This grid will soon be expanded to include the DOE Kansas City Plant and Y12 in Oakridge, Tennessee. Current security policy dictates that kerberos or DCE be the authentication mechanism, and this further limits the expansion of this particular ASCI grid deployment, as most grid deployments use a public key infrastructure.

The goal of the Grid Services project in this context is to simplify access to the diverse computing, storage, network, and visualization resources and to provide superior monitoring and job control mechanisms. To this point, our efforts have focused on implementing the grid infrastructure necessary to allow a user to submit, monitor, and control jobs in a secure manner. The final link in the initial deployment is the user interface itself. Introducing this system to users more accustomed to individualized scripts than to unified grids is a challenge for this particular grid computing environment.

2.   Architecture description

The ASCI Grid Services architecture can be broken down into three layers. Tools developed by the Grid Services team form the layer that a user sees and that used for high-level job control. A middleware layer is the interface between the Grid Services and the lowest layer, the local batch scheduling systems on the various compute resources. One key requirement for the ASCI grid architecture is that local control by the disparate batch scheduling systems (LSF, DPCS, PBS, and NQS) be preserved. The Globus MetaComputing Toolkit[2] was selected as the middleware layer to provide the core interface to those batch scheduling systems, for job control and for job monitoring. Specifically, we currently deploy GTK1.1.3; the changes we make to the Globus software preclude us from rushing to embrace new releases. The Grid Services team works closely with the Globus developers, even becoming Globus developers ourselves to adapt the software to our particular needs. The most major adaptation enables the entire Toolkit to function in a kerberos security infrastructure[3]. These kerberos adaptations have been rolled back into the Globus software distribution so that they are available to other users of the Globus Toolkit. Work in that area is ongoing.

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract: DE-AC04-94AL85000.

Additionally, numerous extensions to the Globus GRAM scripts were made to mesh better with local implementations and to account for the Globus services installation on front end machines rather than on the compute resources themselves. As an example, at LANL, a facility which uses LSF as the batch scheduler, the Globus LSF submit script has been significantly expanded, the queue script restructured and rewritten in perl, an additional information-gathering script written, and the Globus build-info-grid-site script expanded, in the interests of gathering far more job, system, and hardware information and better compartmentalization of the gathering of static versus dynamic information. The use of the data gathered will be detailed in a later section.

A final adaptation of the Globus Toolkit was undertaken to address a severe limitation to the deployment of a production grid, specifically in the Globus Resource Specification Language (RSL)[4]. As it exists, RSL does not allow much of the full range of capability of the local batch schedulers necessary to win the support of users accustomed to that range of capability. RSL was designed to implement a simple set of parameters understandable to any batch scheduler, while the Grid Services approach is to implement the widest range of functionality, even if that functionality is not uniform across batch schedulers. Through some simple changes to the Globus jobmanager and GRAM scheduler, and through more extensive changes to the GRAM submit scripts[5], implementation of some RSL variables was altered and the set of RSL parameters extended. Appendix A lists the Grid Services extensions to RSL and provides a brief description of each. One extension, in particular, is worth noting. The "pass" parameter takes as an argument a string, which is passed unchanged to the submission command of the local batch scheduler. This allows a grid user familiar with a particular batch scheduler to use any arbitrary parameter understood by that batch scheduler without exception. This is important for encouraging the adoption of Grid Services.

The top layer of services developed by ASCI Grid Services and designed to take advantage of the Globus middleware consist of the Production Wizard (a GUI), Workflow Manager, Resource Broker, Information Service, and Monitoring Services. These will be discussed in more detail in the next section; the languages and communication protocols are Java, C++, and Corba[6]. These choices were based partly on current standards at the time of initiation of software development (SOAP[7], for example, was not a viable choice at the time), and partly on envisioned interoperation with other end user tools already using Corba as the communication infrastructure.

For data movement, DRM uses a custom file transfer client and daemon that run in batch mode. The daemon, developed by Sandia National Laboratories, New Mexico, is very similar to the HPSS file movement protocol with the mechanisms for communicating with HPSS movers removed, so that a standard parallel file transfer protocol (PFTP) client can communicate both with a storage system and with any machine which has this daemon installed. Recently, a collaboration has begun which aims to bring the Globus GridFTP project and the HPSS data movement protocol together, and it is anticipated that the future data transfer mechanism will be GridFTP, both for storage and machine-to-machine.

Another element of a robust, secure, grid-computing environment is being addressed by the ASCI PSE program in the form of a distributed Credential Agent for a Kerberos/DCE environment. In an environment in which jobs may run for days or weeks, and workflows for months, it is necessary for the grid infrastructure to provide a mechanism for users to create and designate credentials for use by various processes without circumventing established security regulations and practices. In the short term, the Grid Services Workflow Manager will handle the task of refreshing and forwarding user credentials on a limited basis until Grid Services can become a client of a more flexible Credential Agent.

3. Implementation

The Grid Services Production Wizard (PW), Workflow Manager, Information Service, and Resource Broker intercommunications are all CORBA-based. The Workflow Manager and PW are Java tools; their means of communication is an XML[8] workflow definition vocabulary, the Grid Access Language for HPC Environments (GALE)[9]. A workflow specification, expressed in GALE, is the output of the Production Wizard and the input required by the Workflow Manager. See Appendix B for an example of a PW XML script and its GALE output.

Phase One of the Production Wizard allows an end user to generate a GALE workflow specification by choosing icons representing functional computational steps such as job submission, file transfer, or resource queries in graphical mode. These are dragged and connected in a serial representation of a workflow (see Figure 2). Each icon can be expanded into a user input screen; filling in pre-defined data fields defines a computational submission (Figure 3) or a data transfer step. Existing GALE specifications may be imported, altered, and saved in graphical format After a workflow is defined and has been submitted, standard error and output may be viewed, if desired. Color coding of the icons in the PW workflow representation show the state of each step—started, completed, or failed. In the event of failure, an event dictionary has been defined that translates error codes or error modes from any level of the system. For instance, a user trying to use Grid Services with an improperly configured kerberos credential would be provided with a probable cause (no valid credential) and possible solution (obtain a new kerberos ticket) to the error. Actual causes and solutions are not defined so tersely in the error dictionary. In another example, the ubiquitous Globus error code 12 usually represents a misconfigured resource and the user receives an indication that an administrator must be contacted.

Given a GALE workflow specification, the Workflow Manager sets up serial dependencies, executes resource queries to the Resource Broker, and has as its output a globusrun command with a well-formed resource contact string and Globus RSL argument, or a file transfer command. Current job dependencies are strictly serial; future extensions to the Workflow Manager and to GALE will allow better control for automatic computation restart and parallel dependencies. The Workflow Manager also manages and forwards appropriate user credentials and handles things like reconnecting a new instance of a PW to an existing workflow. Data transfer is handled within the Workflow Manager, as well, with both parallel and serial data transfer capabilities automatically chosen based on data size and source and sink capabilities.

The Resource Broker and Information Service provide information about the "best" resources to use to the Workflow Manager, given a minimal set of user requirements. The information resides in a Lightweight Directory Access Protocol (LDAP)-based Grid Information Server (GIS) which is populated primarily by Globus' original push model, in which job data is culled from local batch schedulers and published to the GIS every thirty seconds, and system configuration data is obtained and published every four hours. In addition to the information published in the Globus implementation, we publish queue data like run and processor limits, job data like service ratios for batch systems implementing fairshare algorithms, submission time, predicted (by the underlying batch scheduler) job start time, and current memory usage. Total wall time, completion time and date, and exit status are collected for completed jobs. Additional resource (not front end) data collected include utilization statistics, paging rates, and memory and scratch space. The purpose of these enhancements is primarily to enable the Monitoring Services, discussed below, to provide a common user view across our platforms. Some of this data might eventually be used by the Broker in the service of sophisticated brokering algorithms, particularly one based on historical queue wait time for similar jobs, with similarity defined by fairshare service ratio, CPUs and runtime requested, etc. The current brokering algorithms are fairly simple. To provide the best resource for a job submission, for instance, the Broker might use the Information Server to discover where the required software (user defined) is installed. It then returns the computational resource with the least jobs currently queued in the default queue of that resource. Because the demand for brokering services from the ASCI user community is low, however, little attention is currently being given to their enhancement, and development resources have been directed elsewhere.

Monitoring Services, as noted above, are dependent on the GIS; these services consist of a number of HTML and CGI scripts which query the GIS whenever a monitoring page is accessed. The main computational resource information page, which requires proper authentication for access, displays all users' running and pending jobs in graphical format, including the number of processors used. Also displayed are the total number of processors in use for each compute resource, and a graphical node-by-node representation of the usage of Sandia National Laboratories', New Mexico, Cplant resource. Monitoring services have been identified as a useful bridge between current user practices and fully enabled grid computing. Work is ongoing to upgrade these services in a manner that will help to integrate the grid view into the user perspective. An example of a User Job Status Page is shown in Figure 4.

4. Grid Security

Security is one of the first considerations in the deployment of any ASCI software. Kerberos is the primary authentication mechanism, implemented via the Generalized Security Framework (GSF)[10], a security framework developed at Sandia National Laboratories. This mechanism enables secure user-to-user and process-to-process communication throughout the ASCI grid, except for access to data in the GIS. That security is implemented via a Netscape Directory Service (NDS) plug-in that provides a Simple Authentication and Security Layer (SASL) mechanism for authentication using GSSAPI over Kerberos. Access to data is authorized using standard LDAP Access Control Instructions (ACIs).

Within this secure grid infrastructure, ASCI grid services provide job and system status monitoring, job submission and control capabilities with elementary computational resource brokering, and automated data transfer capabilities.

5. Project Status and Future Plans

The ASCI grid infrastructure has been deployed at three national laboratories, with the Grid Services layer due to be deployed at two DOE plants in the near future. Job Monitoring is in production use, and a six-month pilot program has been initiated to introduce ASCI compute services customers to Grid Services. Near-future security issues include better credential management with more precise delegation and coordination of compute system and Globus authorization mechanisms. Data transfer mechanisms will no doubt improve with the recent establishment of collaborations between HPSS, ASCI data management personnel, and Globus GridFTP. Parallel job control, automated restart mechanisms based on signal reading from job output files, and better error control and reporting are current development tasks within the Workflow Manager. An important step in advancing user acceptance of grid concepts will be enhanced monitoring of local and remote job and systems status; an important key to that is continued enhancement of the Globus GRAM interfaces to publish relevant information to the GIS. An important development project is the return of more runtime information to the user's desktop via the Production Wizard. Currently job submission and job monitoring are separate functions; their integration depends on the development of a browser-based Production Wizard. More sophisticated brokering systems are available for development, should demand increase. Research and development efforts into co-scheduling are underway, and some early research into network quality of service could be revived, should the need arise. Development of the Production Wizard will necessarily continue both in reaction to user demand and in anticipation of it, and collaborations with other tool developers to grid-enable their utilities will be an important part of future work.

6. References

[1] http://www4.clearlake.ibm.com/hpss/index.jsp

[2] Foster, I., and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, 1997

[3] Moore, P.C., Johnson, W.R., and Detry, R.J.,Adapting Globus and Kerberos for a Secure ASCI Grid. In *Proceedings of SC2001*, Denver, Colorado, November 2001, http://www.sc2001.org/techpaper.shtml.

[4] http://www.globus.org/gram/rsl_spec1.html

[5] Beiriger, J., Johnson, W., Rheinheimer, R., Jette, M.,, *The ASCI Extensions for Globus GRAM Scheduler Scripts*, Software Requirements Specification, Sandia National Laboratories, 2000

[6] CORBA/IIOP Specification: http://www.omg.org/technology/documents/formal/corba_iiop.htm

[7] SOAP Version 1.2: http://www.w3c.org/TR/soap12

[8] W3C XML Specification, http:/www.w3.org/TR/REC-xml

[9] http://vir.sandia.gov/~hpbiven

[10] Detry, R., Kleban, S., Moore, P., and Berg R., The Generalized Security Framework, Presented at CSCoRE 2000, http://www.ccs.bnl.gov, Brookhaven National Laboratory, NY.

Appendix A: Grid Services RSL Extensions

Parameter	Description
Begin_time	Time before which a job may not start
Core_limit	Core file size limit
Debug	Debug level for submit script implementation
End_time	Time before which a job must complete execution
Local_dir	Working directory for a job
Mail_user	Mailing address for job status notifications
Name	Name assigned to job
Pass	String to be passed unchanged to local batch scheduler
Rel_priority	Relative priority for job
Restartable	If "false" scheduler is to make job nonrestartable
Tail_err	File to tail the scheduler spool file for stderr if streaming stderr not supported. For PBS.
Tail_out	File to tail the scheduler spool file for stdout if streaming stdout not supported. For PBS.
Use_script	Path for batch script to be executed in lieu of creating a submission script
Wait_for_job	Job ID of job which must complete before this job starts

Appendix B: Sample Production Wizard XML Script with GALE Output

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkFlow id="Generic" instance="Generic-1" status="READY TO SUBMIT" ortholinks="1" grid="0">

    <WFLayout>

        <!-- movie generation -->
        <WFNode id="AN_start" wfaid="-1" type="Start" x="20" y="20" label="Start" />




        <WFNode id="AN_2" wfaid="2" type="Computation" x="140" y="140" label="Computation Step">

                <DynamicVariable id="DV_1" datatype="string">
                        <GuiWidget id="GW_3_1" type="radiobuttons" count="5" prompt="Resource">
                        <ButtonWidget id="AN_3_1_1_1" prompt="White" />
                                <ButtonWidget id="AN_3_1_1_2" prompt="Red" />
                                <ButtonWidget id="AN_3_1_1_3" prompt="QSC" />
                                <ButtonWidget id="AN_3_1_1_4" prompt="Blue Mtn" />
                                <ButtonWidget id="AN_3_1_1_5" prompt="CPlant" />
                        </GuiWidget>
                </DynamicVariable>

                <DynamicVariable id="DV_2" datatype="string">
                        <GuiWidget id="GW_3_1" type="radiobuttons" count="2" prompt="Resource">
                        <ButtonWidget id="AN_3_1_1_1" prompt="Theta Cluster" />
                                <ButtonWidget id="AN_3_1_1_2" prompt="Lambda Cluster" />
                        </GuiWidget>
                </DynamicVariable>

                <DynamicVariable id="DV_10" datatype="string">/bin/ls
                         <GuiWidget id="GW_2_1" type="textfield" dataLength="30" prompt="Executable File Path  " />
                </DynamicVariable>

                <DynamicVariable id="DV_13" datatype="string">-lt
                         <GuiWidget id="GW_2_1" type="textfield" dataLength="30" prompt="Application Arguments" />
                </DynamicVariable>

                <DynamicVariable id="DV_15" datatype="string">/scratch/wbarber
                         <GuiWidget id="GW_2_1" type="textfield" dataLength="30" prompt="Directory            " />
                </DynamicVariable>

                <DynamicVariable id="DV_16" datatype="string">
                        <GuiWidget id="GW_2_1" type="textfield" dataLength="20" prompt="Job Name                 " />
        </DynamicVariable>

                <DynamicVariable id="DV_17" datatype="string">
                        <GuiWidget id="GW_2_1" type="textfield" dataLength="30" prompt="Input File Path(s)       " />
        </DynamicVariable>

                <DynamicVariable id="DV_18" datatype="string">
                        <GuiWidget id="GW_2_1" type="textfield" dataLength="30" prompt="Log File Path            " />
        </DynamicVariable>

                <DynamicVariable id="DV_19" datatype="int">
                        <GuiWidget id="GW_2_1" type="textfield"  dataLength="4" prompt="Number Of Processors" />
        </DynamicVariable>
```

```
			<DynamicVariable id="DV_20" datatype="int">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="4" prompt="Number Of Hosts     " />
		</DynamicVariable>

			<DynamicVariable id="DV_21" datatype="int">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="8" prompt="Wall Clock Time (Min)"/>
		</DynamicVariable>

			<DynamicVariable id="DV_22" datatype="int">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="8" prompt="CPU Time (Min)"/>
		</DynamicVariable>

			<DynamicVariable id="DV_23" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="15" prompt="Run After (MM:DD:hh:mm)"/>
		</DynamicVariable>

			<DynamicVariable id="DV_24" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="15" prompt="Stop Before
(MM:DD:hh:mm)"/>
		</DynamicVariable>

			<DynamicVariable id="DV_25" datatype="string">drmqueue
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Queue Name        "/>
		</DynamicVariable>

			<DynamicVariable id="DV_26" datatype="string">single
				<GuiWidget id="GW_2_1" type="radiobuttons" count="3" prompt="Job Type">
				<ButtonWidget id="AN_3_1_2_1" prompt="MPI" />
				<ButtonWidget id="AN_3_1_2_2" prompt="Single" />
				<ButtonWidget id="AN_3_1_2_3" prompt="Multiple" />

				</GuiWidget>
			</DynamicVariable>

			<DynamicVariable id="DV_27" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Local Scheduler Flags"/>
		</DynamicVariable>

			<DynamicVariable id="DV_28" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Extensions          "/>
		</DynamicVariable>

			<DynamicVariable id="DV_29" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
		</DynamicVariable>
			<DynamicVariable id="DV_30" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
		</DynamicVariable>

			<DynamicVariable id="DV_31" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
		</DynamicVariable>
			<DynamicVariable id="DV_32" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
		</DynamicVariable>

			<DynamicVariable id="DV_33" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
		</DynamicVariable>
			<DynamicVariable id="DV_34" datatype="string">
				<GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_35" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_36" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_37" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_38" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_39" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_40" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_41" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_42" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_43" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_44" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_45" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_46" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

            <DynamicVariable id="DV_47" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Variable Name"/>
    </DynamicVariable>
            <DynamicVariable id="DV_48" datatype="string">
                    <GuiWidget id="GW_2_1" type="textfield"  dataLength="20" prompt="Environment Value"/>
    </DynamicVariable>

     </WFNode>




  <WFNode id="AN_finish" wfaid="-1" type="Finish" x="380" y="380" label="Finish" />

  <WFLink id="WFL_0" from="AN_start" to="AN_2" label="" />
  <WFLink id="WFL_2" from="AN_2" to="AN_finish" label="" />
```

```
        </WFLayout>

        <ResourceQuery id="WFR_1" output="WFR_1" >
                <SoftwareRequest>sw=globus</SoftwareRequest>
                <ResourceRequest>hn=malaga.lanl.gov</ResourceRequest>
                <ResourceManagerRequest>schedulertype=lsf</ResourceManagerRequest>
        </ResourceQuery>

        <Computation id="1" input="WFR_1">

        <Description>Generic Computation Node</Description>

                <ComputationAttribute name="executable">@DV_10@</ComputationAttribute>
                <Argument name="cmd args">@DV_13@</Argument>
                <ComputationAttribute name="directory">@DV_15@</ComputationAttribute>
                <ComputationAttribute name="extend">(args -lt)(name wbarber_ls_job)(stdin /dev/null)(logfile /dev/null)(begin_time 12:15
pm)(end_time 23:00)</ComputationAttribute>
                <ComputationAttribute name="count">@DV_19@</ComputationAttribute>
                <ComputationAttribute name="hostCount">@DV_20@</ComputationAttribute>
                <ComputationAttribute name="maxWallTime">@DV_21@</ComputationAttribute>
                <ComputationAttribute name="maxCpuTime">@DV_22@</ComputationAttribute>
                <ComputationAttribute name="queue">@DV_25@</ComputationAttribute>
                <ComputationAttribute name="jobType">@DV_26@</ComputationAttribute>
                <ComputationAttribute name="pass">@DV_27@</ComputationAttribute>

                <Environment name="@DV_29@">@DV_30@</Environment>
                <Environment name="@DV_31@">@DV_32@</Environment>
                <Environment name="@DV_33@">@DV_34@</Environment>
                <Environment name="@DV_35@">@DV_36@</Environment>

                <ComputationAttribute name="directory">@WFR_1.scratchdir@/@grid.user@</ComputaionAttribute>

        </Computation>

</WorkFlow>
```

Figures

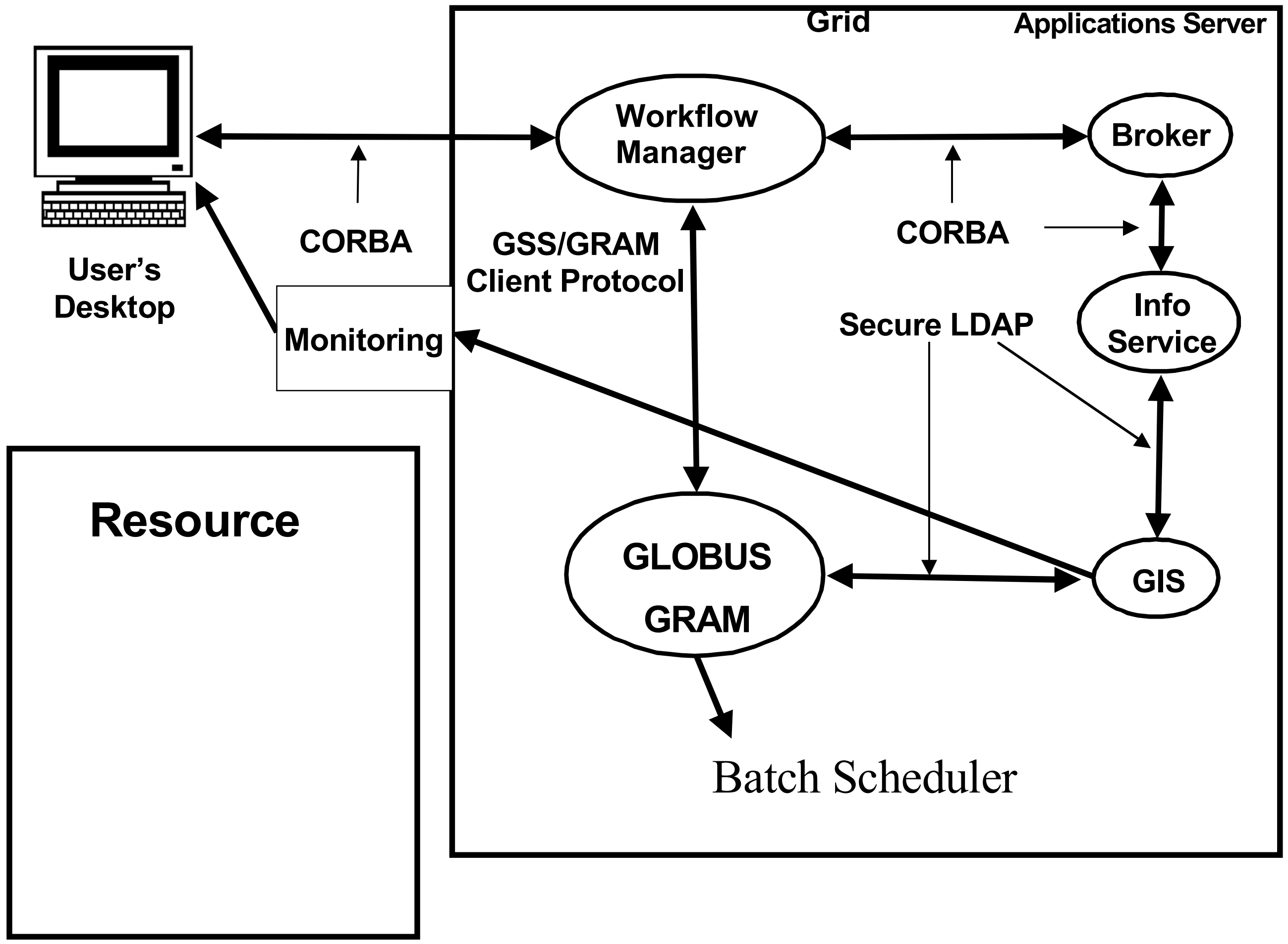


Figure 1: Architecture, including Grid Services, Globus, and local batch scheduler

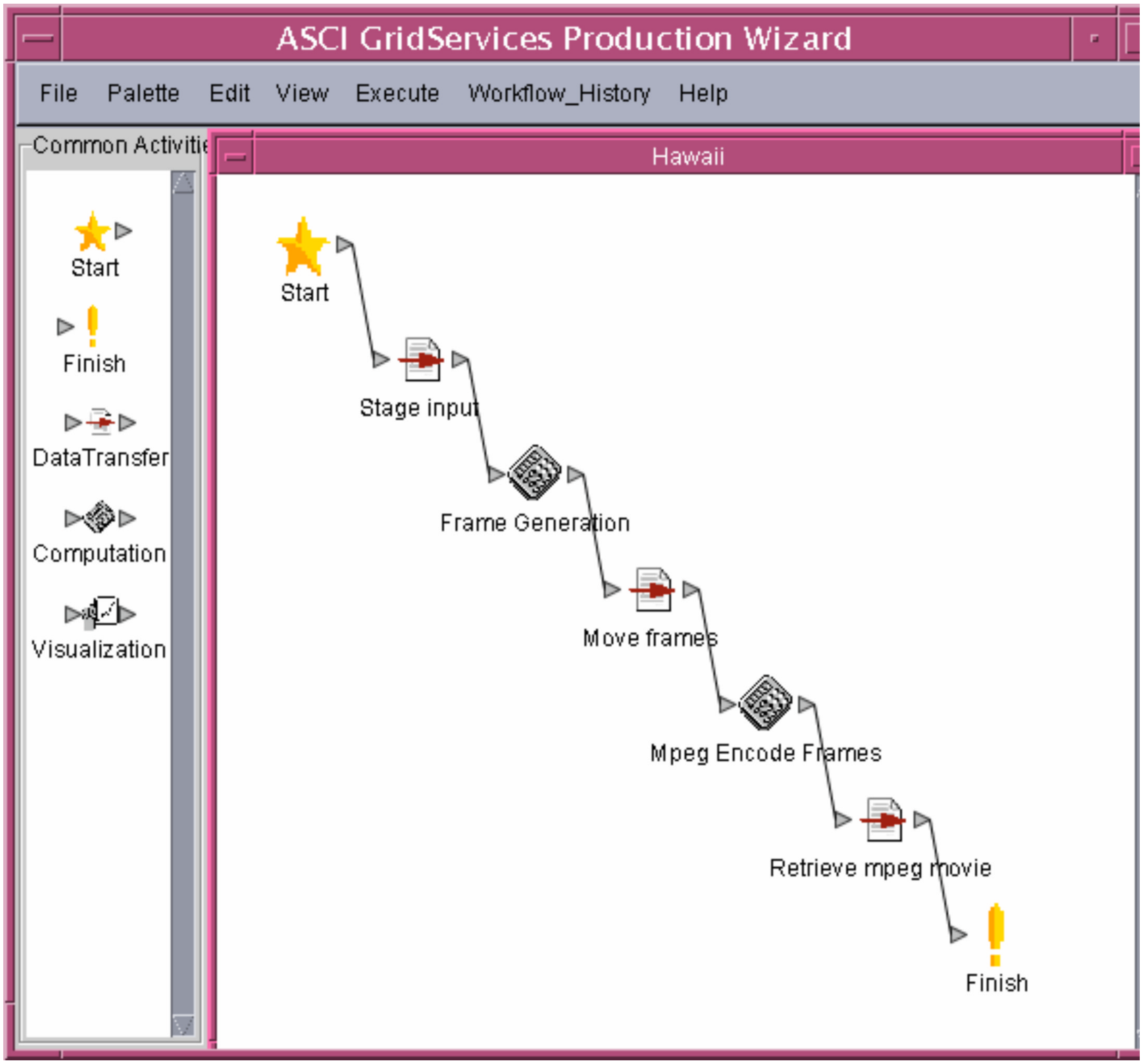
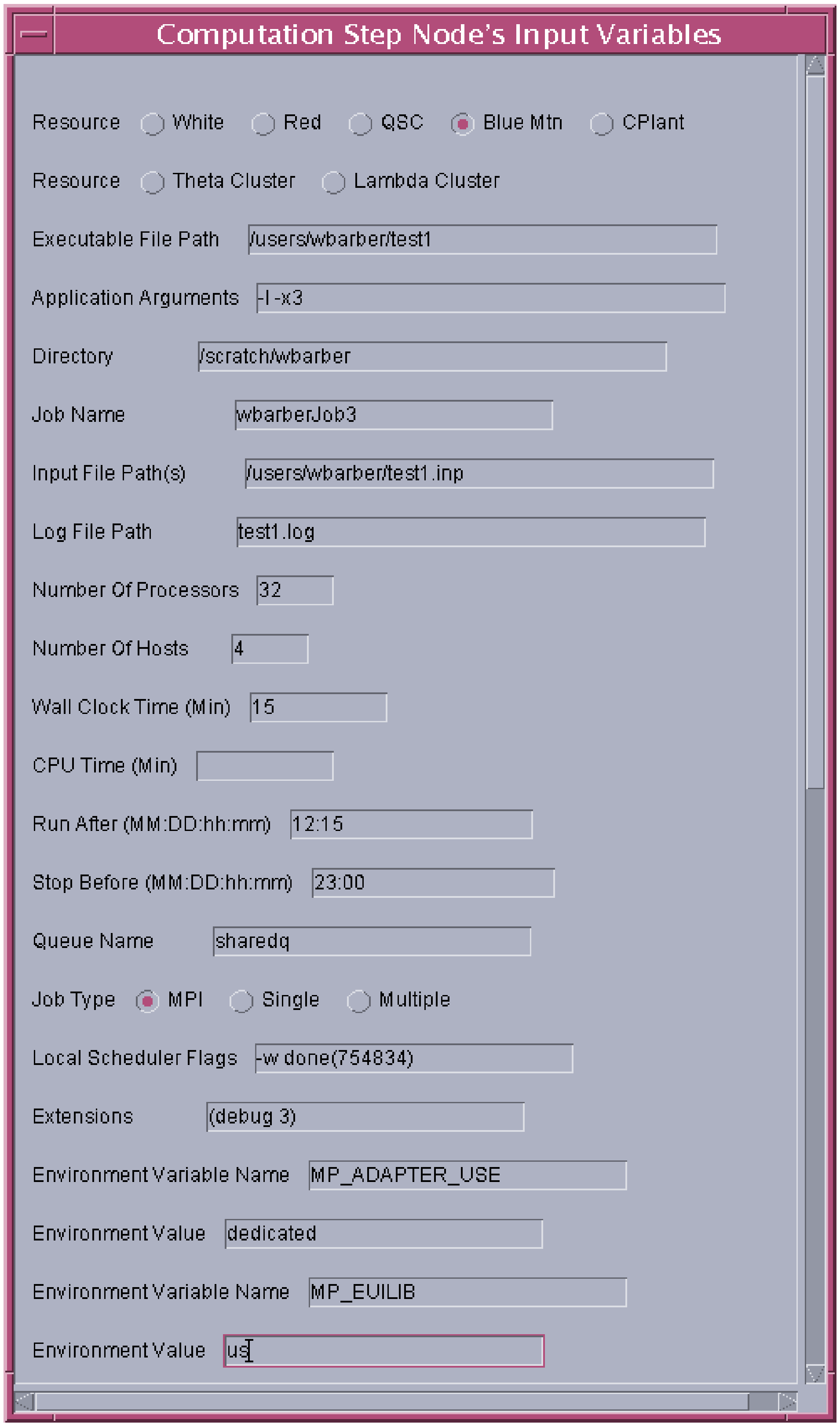


Figure 2: Production Wizard workflow creation GUI showing completed drag and drop workflow representation.



Resource White Red QSC [✓ Blue Mtn] CPlant

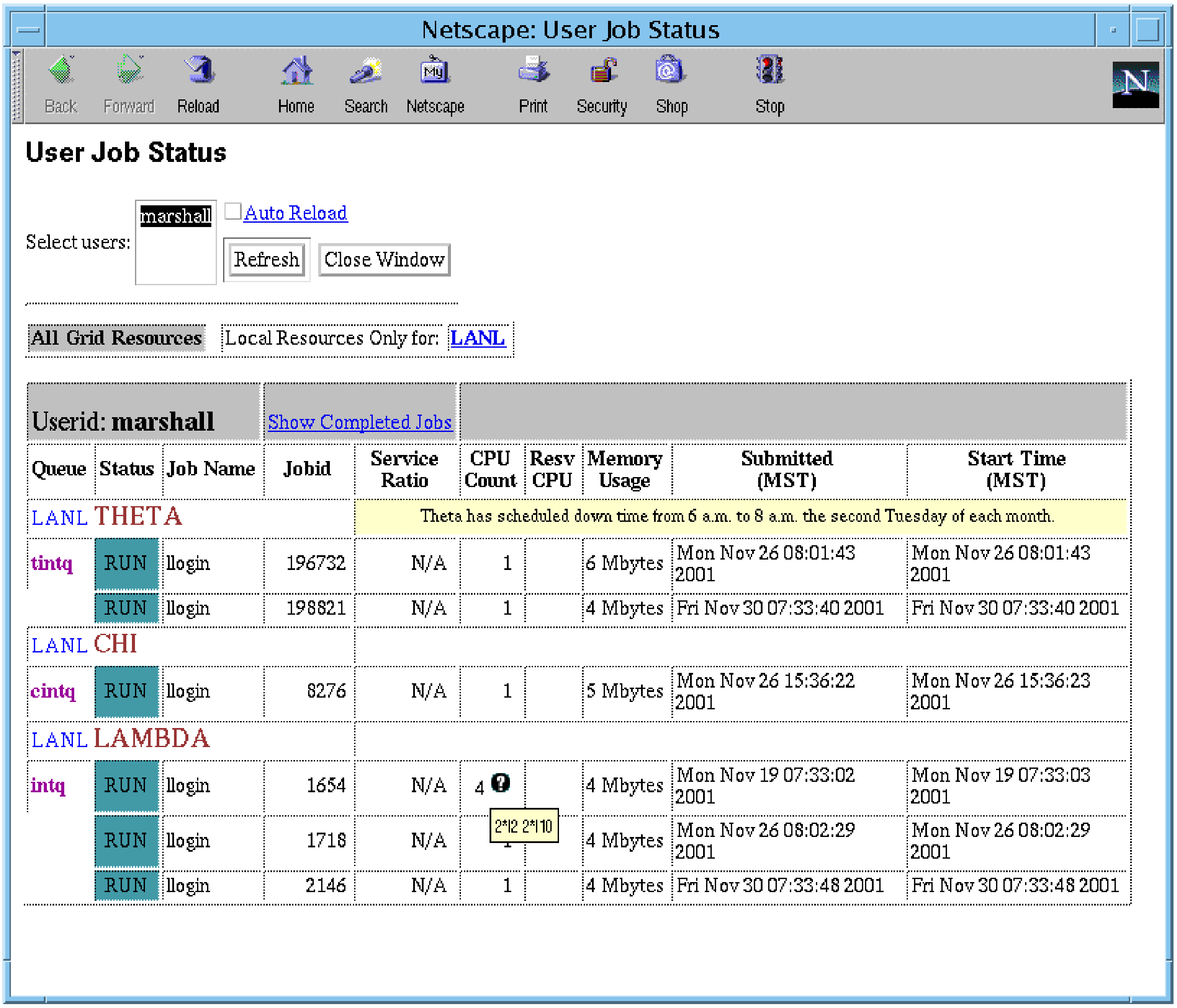
Resource Theta Cluster Lambda Cluster

Executable File Path	/users/wbarber/test1
Application Arguments	-I -x3
Directory	/scratch/wbarber
Job Name	wbarberJob3
Input File Path(s)	/users/wbarber/test1.inp
Log File Path	test1.log
Number Of Processors	32
Number Of Hosts	4
Wall Clock Time (Min)	15
CPU Time (Min)	
Run After (MM:DD:hh:mm)	12:15
Stop Before (MM:DD:hh:mm)	23:00
Queue Name	sharedq

Job Type [✓ MPI] Single Multiple

Local Scheduler Flags	-w done(754834)
Extensions	(debug 3)
Environment Variable Name	MP_ADAPTER_USE
Environment Value	dedicated
Environment Variable Name	MP_EUILIB
Environment Value	us

Figure 3: Computational step specification (see Appendix B for XML input and GALE output).



User Job Status

Select users: marshall

Auto Reload

Refresh Close Window

All Grid Resources | Local Resources Only for: LANL

Userid: marshall | Show Completed Jobs

Queue	Status	Job Name	Jobid	Service Ratio	CPU Count	Resv CPU	Memory Usage	Submitted (MST)	Start Time (MST)
LANL THETA				Theta has scheduled down time from 6 a.m. to 8 a.m. the second Tuesday of each month.					
tintq	RUN	llogin	196732	N/A	1		6 Mbytes	Mon Nov 26 08:01:43 2001	Mon Nov 26 08:01:43 2001
	RUN	llogin	198821	N/A	1		4 Mbytes	Fri Nov 30 07:33:40 2001	Fri Nov 30 07:33:40 2001
LANL CHI									
cintq	RUN	llogin	8276	N/A	1		5 Mbytes	Mon Nov 26 15:36:22 2001	Mon Nov 26 15:36:23 2001
LANL LAMBDA									
intq	RUN	llogin	1654	N/A	4		4 Mbytes	Mon Nov 19 07:33:02 2001	Mon Nov 19 07:33:03 2001
	RUN	llogin	1718	N/A	1		4 Mbytes	Mon Nov 26 08:02:29 2001	Mon Nov 26 08:02:29 2001
	RUN	llogin	2146	N/A	1		4 Mbytes	Fri Nov 30 07:33:48 2001	Fri Nov 30 07:33:48 2001

2*12 2*110

Figure 4: Current jobs across resources for a given user. Pending jobs will also display projected start times when such information is available from the local batch scheduler. A rollover of the cpu count shows processor layout on an SMP.