

Building a unified grid, Part 2: Portlet interfaces in the grid user environment

Level: Intermediate

Abel W. Lin (awlin@ncmir.ucsd.edu), Software Architect, National Center for Microscopy and Imaging Research

29 Nov 2005

"[Building a unified grid, Part 1](#)" discussed what it means to "grid-enable" an end-to-end process and described the grid-based system architecture developed in the Telescience Project at the National Center for Microscopy and Imaging Research. In this article, we tackle the grid interface. Find out why the grid interface is critical to the end-to-end process and how the use of portlets has led to a richer grid user environment for NCMIR's grid project.

A hallmark of our current advances in IT infrastructure is the ability to capture and process large amounts of information and data. The ability to process and manage -- and make decisions from -- vast amounts of information was once reserved for those privileged few that access to large "big-iron" supercomputers. Now, grid computing has flattened the playing field, allowing everyone to have access to (or to create) large pools of computing power. However, the development and deployment of the underlying grid infrastructure is but one step toward extending access to the end user. To maximize use of the grid by any end-to-end process, the grid must be transparently embedded into the natural working process of the particular user.

Why is the grid interface critical to the end-to-end process?

The classic rationale for grid portals is that grid technologies are often difficult to learn and use, so a simple, intuitive Web interface will give users a way to launch different applications. In a unified grid, like the one described in "[Building a unified grid, Part 1](#)," a simple, intuitive Web interface is indeed important, but a unified grid needs more than just a simplified grid interface. It needs a transparent way to integrate grid functionality into existing application interface environments. This transparent integration of the grid is the hallmark to an interface for a unified grid.

A grid interface needs more than just radio buttons and checkboxes

Use-case models for first-generation grids (and their supercomputing-center predecessors) were akin to the hub-and-spoke model used by the airline industry. User data environments were treated as the "hub," and at every step, the users (and their data) are routed toward and required to log in to one of the few "spokes" (or virtual organizations) across the country to use the computing resources. Initial implementations were not ideal for end users, as they often required the ability to optimally function using a command-line interface. Soon afterward, the login mechanism was integrated into a grid portal. Then, instead of logging in via a command prompt, users were directed to a Web page. These first-generation grid portals, however, simply replaced complex command-line arguments and syntax with radio buttons and checkboxes. While this was an effective method to simplify the syntactical interface, it did little to integrate the grid into the users natural workflow process.

Figure 1. First-generation grid portal interfaces

A. Command Line

```
globusrun -b -r tg-login1.sdsc.teragrid.org/jobmanager-pbs
"&(directory= /gpfs/awlin/test4) (executable= /users/awlin/bin/
ptbbq) (arguments= -i1 hpf3.homographies.txt -i2 hpf3.preali.rot.SL -
i3 20) (jobType=mpi) (count=16) (maxWallTime=720) (stdout=txbr.stdout)
(stderr=txbr.stderr) "
```

B. Example of 1st Generation Portal

Job Submission Form:

Enter Job Name : <input type="text" value="/bin/uname"/>	Environment: <input type="text" value="(count=16)(maxWallTime=720)"/>
Select machine: <input type="text" value="tg-login1.sdsc.teragrid.org"/> Type: <input type="text" value="mpi"/>	Directory: <input type="text" value="/gpfs/awlin/test4"/>
Enter the full path to the executable: <input type="text" value="/users/awlin/bin/ptbbq"/>	Stdout: <input type="text" value="txbr.stdout"/>
Arguments: <input type="text" value="-i1 hpf3.homographies.txt -i2 hpf3.preali.rot.SL -i3 20"/>	Stderr: <input type="text" value="txbr.stderr"/>

In "[Building a unified grid, Part 1](#)," I discussed a unified grid that allowed us not only the ability to compute more data faster but to change the way the entire process is perceived. The challenges of building a unified grid touched upon all aspects of the architecture. As part of the requirements of a unified grid, the next generation of portal interfaces must more accurately reflect the natural workflow process of the end user's application environment. This environment more closely mirrors a *point-to-point* flow, where data and information flow freely between external applications, rather than returning to a hub after every step. Quite simply, there should no longer be grid portals, but only application portals.

Portlets and JSR168

It's no surprise that portals have emerged as a dominant source of application and information delivery. A leading analyst has championed the portal as a mechanism that provides access to and interaction with relevant information, applications, business processes, and human resources by select, targeted audiences in a highly personalized manner. According to the analyst, the enterprise portal has become the most desired user interface in Global 2000 enterprises and is ranked as one of the top 10 areas of technology focus by CIOs.

Until recently, however, portal tools have limited the creativity and dynamism of application portals. These tools were often developed around custom-developed application components, often based on Sun Microsystems' Java™ 2 Platform, Enterprise Edition (J2EE) Web application model. In the early days of portal development, little emphasis was placed on component interoperability or reuse because portal APIs for development or customization was often limited to only a single portal framework. Moreover, because applications portals were a nascent concept, little emphasis was placed on managing a consistent presentation layer or on the management of the larger processes that often encapsulate a particular application.

Recently, two important standards have emerged to address the portable development of portals: the Web Services for Remote Portlets Specification (WSRP) and Java Specification Request 168 Portlet Specification (JSR168). Independent of programming languages and platforms, WSRP defines Web Services Description Language (WSDL) interfaces and semantics for presentation-oriented Web services. JSR168 meanwhile, defines a standard Java portlet API, a portlet container, and the contract between the API and the container. These two emerging standards, combined

with new robust portal framework projects, such as those from the GridSphere Project, have enabled the development of a unified grid interface. Armed with these standards, portlets have become one of the most exciting areas for presenting portal environment. The number of vendors (and open source projects) that support portlets serve as evidence. These include IBM WebSphere®, Sun ONE Portal Server, Oracle 9iAS, Jetspeed, and the GridSphere project.

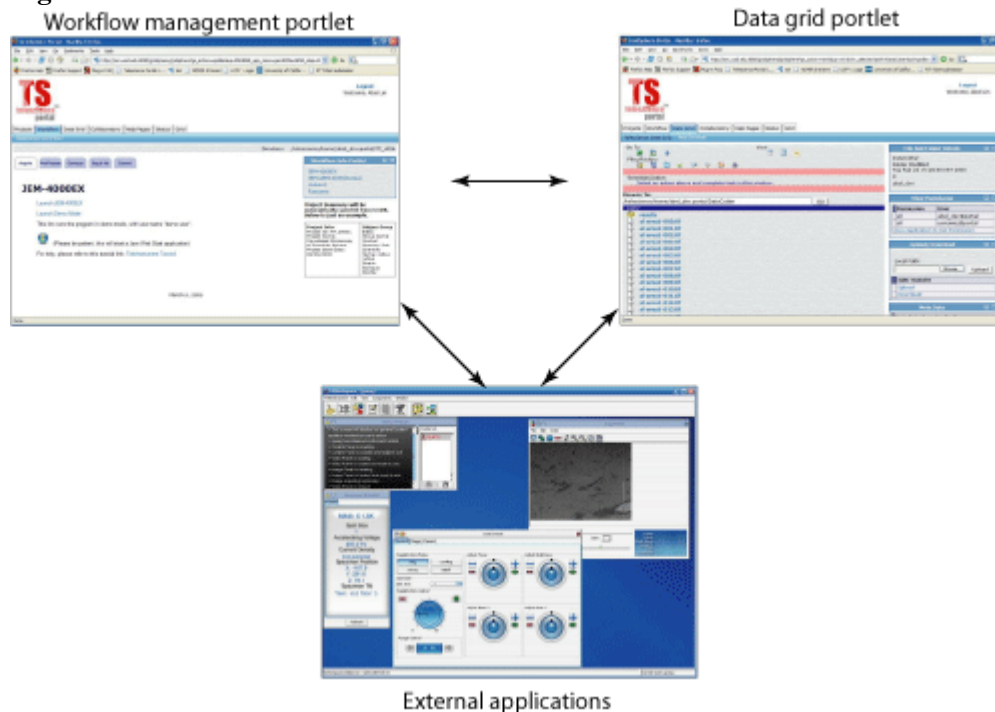
Application-centric portals, such as Telescience, take advantage of the portable presentation layers of portlets, and of persistence logic and state information between portals (and further passed on to external applications). It is this vital information that makes a unified point-to-point grid interface possible.

The Telescience Portal

The basic functionality of Telescience Portal is the user-managed 3-D microscopy workflow, where the sequence of steps required for acquiring, processing, visualizing, and extracting useful information from the 3-D microscopy data is presented to the user in an intuitive single sign-on Web environment. As with all first-generation portals, a major accomplishment for Telescience V1.0 was to create simple Web-accessible interfaces to a heterogeneous set of middleware via a single user name and password. Through Telescience V1.0, for example, users could browse the data grid via a custom interface or launch jobs via Web-wrapped middleware commands. These interfaces, however, were designed in autonomy for singular interactions whose capabilities were developed by mirroring a command-line interface to the particular middleware tools.

The Telescience V2.0 infrastructure (described in [Part 1](#)) allows us to move beyond interfaces with singular actions and integrate them into a richer user environment that's automated and dictated by the process and not by the grid middleware. Information displayed by the data grid portlet, for example, is fully controllable by actions within the main workflow portlet. As the user navigates to various applications within the workflow context, the data grid portlet and other portlets dynamically change to reflect the user actions. Actions, for example, can range from simple directory actions like creation and deletion to advanced file manipulation, such as on-the-fly format conversions.

Figure 2. The Telescience Portal



That capability is enabled by the ability to track stateful information unique to each user. Telescience V1.0 could rudimentarily track the user's progress through the overall experimental workflow. Telescience V2.0 logs and tracks all

user information, down to the individual parameters used to launch specific applications. This function is enabled by the ability to manage persistence logic and state information via the portlet frameworks. The availability of this fine-grain information makes Telescience portlets and Application to Middleware Interaction Component (ATOMIC) tools amenable to projects where that audit information is legally required, such as Health Insurance Portability and Accountability Act (HIPAA) requirements for the management of clinical data.

Moreover, [Part 1](#) described an on-demand, unified grid where data from the results of analysis can flow freely and interface with instruments to provide refinement of data collection parameters. Telescience begins to enable that process by merging real-time user actions and state information -- managed by the portal, with metadata from past experiments from a federated database -- to create feedback scenarios during the experimental process. This feedback is critical to the concept of the unified grid introduced in the previous article, and the vision of grid serving not only as a means to faster data but also as a means to better data.

A richer grid user environment

A unified grid enables the transition to an applications environment, where the actions of the underlying middleware are completely transparent to the end user. This transparency can now be reflected across the user interface components. Quite simply, we have enough interfaces to middleware tools that simply replace command-line arguments with easy-to-use checkboxes and radio buttons. A unified grid, coupled with portlet interfaces, enables and marks the transition to a richer user environment, where middleware functionality is natively encapsulated within the natural scientific workflow, and the need for explicit interfaces to specific middleware tools are no longer necessary.

Resources

Learn

- Learn more about the [Telescience Project](#).
- Learn more about research at the [National Center for Microscopy and Imaging Research \(NCMIR\)](#).
- Learn more about the [The GridSphere Framework](#).
- Learn more about the various grid software components at the [NSF Middleware Initiative](#).

Get products and technologies

- Find out about downloading Telescience open source software used for deploying grids at the [Telescience Project](#).
- Download a beta version of the [Telescience ATOMIC Toolkit](#).

Discuss

- Get involved in the grid portals community at [OpenGridPortals.org](#).
-

About the author

Abel W. Lin is the architect and technical lead for the Telescience Project at the National Center for Microscopy and Imaging Research. He has more than five years' experience applying grid and other computer science technologies to scientific processes. He designed and led the implementation of the first-generation proof-of-concept systems for the Telescience Portal and ATOMIC components of the Telescience Project, and is an interdisciplinary scientist with published research in biology and computer science. His interests include distributed systems architecture, software project management, and structural biology. Outside of the office, he is an avid reader, golfer, and bodysurfer.
