

A portal interface to *my*Grid workflow technology

Stefan Rennick Egglestone ^a, M.Nedim Alpdemir ^b, Chris Greenhalgh ^a, Arijit Mukherjee ^c and Ian Roberts ^d

^a. School of Computer Science and IT, University of Nottingham

^b. School of Computer Science, University of Manchester

^c. School of Computing Science, University of Newcastle upon Tyne

^d. Department of Computer Science, University of Sheffield

Abstract

Workflow technology previously developed by the *my*Grid project has been used to automate a number of complex bioinformatics analyses, allowing much larger volumes of data to be produced than would be possible if these analyses were to be performed manually. *my*Grid workflows can be constructed using the Taverna workbench, and the analysis specified by such a workflow can be performed through the use of a workflow enactment engine.

This paper describes the design and implementation of a simple user-interface which aims to support the use of workflow technology by providing web-based access to workflow and results management facilities. This interface has been constructed using the Gridsphere portal framework, and makes use of storage facilities provided by the *my*Grid Information Repository.

1 Introduction

A rich selection of computational resources are available to scientists working with biological data. It is common for these scientists to wish to perform composite analyses involving multiple resources, and the *my*Grid project has developed workflow-based middleware which allows the performance of such analyses to be automated. Development of this middleware has focussed on providing support for professional bioinformaticians, who are likely to be expert computer users, and who may wish to automate complex analyses involving the use of large numbers of resources. However, workflow techniques may also be useful for experimental biologists, who are often less expert in computing, and who may wish to automate simpler analyses.

Members of this second group of users are unlikely to wish to construct their own workflows, but may wish to use workflows which have been constructed by other, more expert users. To support this scenario, the *my*Grid project have developed the *my*Grid Portal Interface (MPI), which is a simplified, web-based interface to *my*Grid workflow enactment and storage technology. The MPI provides functionality which allows groups of users to share and enact a number of work-

flows, and to browse data which has been produced by previous workflow enactments.

This paper describes the design and implementation of the MPI, and is structured as follows: Section 2 describes the use of workflow technology to automate biological analyses, and motivates the development of the MPI. Section 3 describes general details of the design and implementation of the MPI, and section 4 describes a number of interesting features of this design in more detail. Section 5 then discusses improvements that could be made in the design of the MPI, and in particular those that might be made to support the requirements of large groups of users who wish to publish many workflows.

2 Motivation for the development of the MPI

2.1 A categorization of users of bioinformatics resources

A wide variety of resources are available which support the storage, retrieval and processing of biological data. These resources are often publicly- and freely-available, and have become essential tools for scientists who work with this type of data.

Previous work [19] has identified a group of users of these resources who can be given the label of professional bioinformatician. Members of this group of users rarely gather new data experimentally. Instead, they mine storage resources for data which has been submitted by other biologists, from which they generate new knowledge by the application of complex combinations of data processing operations. The *my*Grid project has focussed on providing support for members of this group of users, by providing workflow-based middleware that allows these types of complex analyses to be automated.

However, dialog with the user community has revealed another group of users of distributed resources. Members of this group are primarily biologists who gather new data by the application of experimental techniques. They are likely to submit this experimental data for archiving in a storage resource, and are also likely to perform simple analyses of this data using a small number of distributed processing facilities. Members of this group of users are in general not expert in the use of computational techniques, and are unlikely to construct workflows to automate their analyses. They may, however, be willing to use workflows constructed by other, more expert users. The rest of this section introduces existing *my*Grid workflow technology, explains why it is less likely to be used by members of this second group, and motivates the provision of a simplified interface to this technology which may be useful to members of this group.

2.2 *my*Grid workflow technology

Existing *my*Grid technology allows a complex analyses involving multiple distributed resources to be expressed as a workflow. A workflow which has been constructed to represent an analysis consists of a description of the inputs required by the analysis, the resources used by the analysis, and the results that performing the analysis would generate. Such an analysis can be performed by the use of a workflow enactment engine. To support the use of workflow technology, the *my*Grid project have developed the Taverna workflow workbench [18, 9], which is a graphical user interface that provides functionality to construct and enact workflows, and to store both workflows and data produced by enactments to the local file system. *my*Grid workflow technology, including Taverna, has been successfully used in a number of research projects, including investigations into Williams-Beuren Syndrome [17] and Grave's Disease [15].

Although workflow technology has been proven to be useful to these projects, it has been found that the construction and maintenance of work-

flows is a difficult process, which requires extensive expertise in computing and bioinformatics. As such, members of the second group of users introduced in subsection 2.1 above, who are not expert users of computer technology, are unlikely to wish to construct workflows to automate their analyses. However, since many of the analyses that these users perform are simple and standardized, it may be that the enactment of a standard set of workflows which have been constructed by other, more expert users may provide a useful automation mechanism for them.

Since members of this group of users are less expert in computing, it may be unreasonable to expect them to download, install and learn the interface provided by the Taverna workflow workbench. Although this interface is well-designed, it does aim to support both workflow construction and enactment, and is hence more complex than an interface which solely provided workflow enactment facilities could be. As an alternative to Taverna, the *my*Grid project have developed a simplified, portal-based interface to *my*Grid workflow enactment technology called the *my*Grid Portal Interface (MPI), the design of which has been targeted at this group of users. This interface is used through a standard web-browser, access to which is ubiquitous amongst the user community, and provides facilities for the storage and retrieval of both workflows and enactment data. The rest of this paper describes and evaluates the implementation of the MPI.

3 Basic design details

3.1 Access control and data sharing

Discussion with potential users of the MPI have revealed that many work in small groups, with a relatively fixed membership and which are localized to one organization. These groups often consist of a number of scientists with similar research interests. It is common for users in these groups to perform similar analyses on similar sets of input data, and to wish to share any results which are produced by the performance of these analyses. They may, however wish to restrict access to this data to members of the group. Currently, they normally use local or networked file systems for data storage, and share data by the use of shared network areas or email.

The design of the initial prototype of the MPI has focussed on support for small, static groups such as these, rather than for larger, cross-organizational and more dynamic groupings. This has had the advantage of simplifying security and storage requirements, meaning that rapid development of the initial prototype has been possible. It has

been assumed that one MPI installation will be made per group of users, and that all members of this group will be allocated an account in this installation, which will be protected by a username and password. The design of the MPI supports a course level of sharing of data, in which all members of a group share access to all workflows and enactment data which have been published in a particular installation.

3.2 Selected details of the interface design

The MPI provides user account administration, workflow management, workflow enactment and results management functionality. This paper, due to considerations of space, does not attempt to describe all of these in detail, but instead introduces a small number of features of the MPI user interface design.

3.2.1 Workflow management functionality

Users can upload workflows into an MPI installation by providing details of a file on their local file system into which the workflow has been saved. Once a workflow has been uploaded, it can be grouped with other workflows into a collection. Figure 1 below shows how the list of available collections is presented to an MPI user.

Workflow collection name	Links
collection1	view delete
collection2	view delete

load new reload all add new

Figure 1: A list of workflow collections

Hyperlinks labelled *view* can be used to view a list of the workflows in a given collection. Workflows can be added to this list, deleted from this list, and the details of workflows in the list can be viewed.

Hyperlinks labelled *delete* can be used to delete a given collection, thereby deleting all workflows which have been added to that collection. The button labelled *add new* can be used to add a new, named collection, and the buttons labelled *load new* and *reload all* are used to synchronize this page if other MPI users have modified the list of collections.

3.2.2 Workflow enactment functionality

Users can choose to enact a workflow in a collection. A user who chooses to enact a workflow is presented with an form which gathers input parameters necessary for this enactment. An example

of such an form is shown in figure 2 below, which has been generated for a workflow which has two inputs labelled *in0* and *in1*.

Figure 2: An input form for a workflow with two textual inputs

Once a user has started an enactment, they can monitor its progress. After the enactment has completed, an enactment summary page is generated. An example of such a summary page is shown in figure 3 below.

Input parameters provided by user
in1
in0
Output parameters
output1

Figure 3: A enactment summary page for a workflow enactment

This summary page contains hyperlinks which allow a user to browse both the input parameters which were provided to start an enactment, and the output parameters which were produced by this enactment. These input parameters can consist of items of text, HTML or images. Figure 4 below shows an example of an image which has been produced by an enactment, and which is being rendered by the MPI.

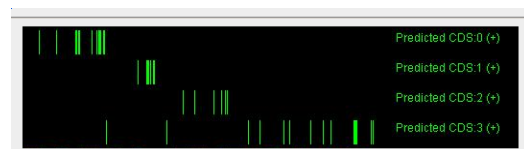


Figure 4: An image produced by enacting a workflow

4 Selected implementation issues

4.1 Technology choices for implementation

Necessary features of portal interfaces, such as login systems, are difficult to implement securely and reliably. However, a number of portal interfaces exist which provide such generic features.

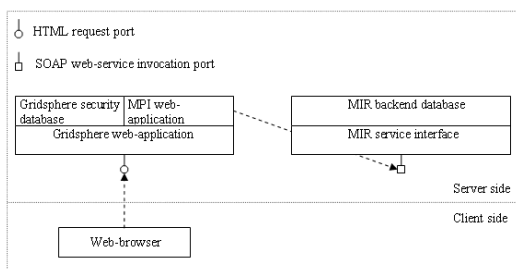


Figure 5: Basic architecture of the MPI

These portal interfaces are intended to be frameworks into which custom applications can be deployed.

Gridsphere [20, 2] is an example of an existing portal framework, which has been used during development of the MPI. Gridsphere has been used in a number of eScience projects, including GeneGrid [16] and HPC-Europa [3], and seems to be a lightweight and reliable choice. The MPI inherits its login system and administration tools from Gridsphere, and takes advantage of layout features that it provides. The MPI application itself has been developed to conform to the JSR-168 portlet standard [5] which attempts to define a standard mechanism for specifying portal-based web-applications. This standard is supported by a number of other portal frameworks, including uPortal [10] and Jetspeed-2 [4], meaning that it should be possible to deploy the MPI into these portal frameworks with minor modifications to a number of configuration files.

The MPI takes advantage of storage facilities provided by the myGrid Information Repository (MIR), which has previously been developed as part of the myGrid project. This is a web-service [12] which provides specific support for the storage and retrieval of workflows and enactment data, and which uses a relational database as a backend. The MIR web-service interface is defined using the Web Service Description Language (WSDL) [11] and is invoked using the Simple Object Access Protocol (SOAP) [7]. The MIR does not currently provide any form of role-based access control to data, but can be protected using HTTP basic authentication [6] which requires the provision of a valid username and password with every service invocation.

4.2 Basic architecture

Figure 5 above shows the basic architecture supporting the MPI.

A user accessing the MPI through their web-browser must first login through Gridsphere, providing a username and password. These login details are

authenticated against the Gridsphere security database. Once a user has logged in, any requests that they make for MPI web content are forwarded by Gridsphere to the MPI web-application, which generates relevant web-pages. This may require the retrieval of data from the MIR, in which case information retrieval facilities provided by the MIR web-service interface are used. Alternatively, generated web-pages may define a form which will be used to gather information to be added to the MIR.

If an MIR protected by basic HTTP authentication is being used, the MPI installation must be statically configured with a single username and password to be used during MIR access.

4.3 Caching of data from the MIR

Sets of results generated by enacting workflows can be large. For example, the enactment of one particular workflow used as a test case for the MPI regularly generates 40mb of data. If MPI and MIR installation are hosted on different machines, the cost of transferring this data on demand over a network can be excessive.

To attempt to reduce the demand on network resources, the MPI constructs a write-through cache per logged-in portal user. This cache is constructed when a user logs in to the portal, and is destroyed either when they log out or when their portal session times out. All communication with the MIR is made through this cache. An item of data is downloaded into the cache from the MIR when first requested, from where future requests for this entity may be served directly. Any entities added to the MIR by the MPI are added through the cache, and are then available without ever having to be downloaded.

It could be dangerous, however, to cache all items of data indefinitely. This might cause the memory requirements of the cache to expand excessively until a memory overrun was caused. Instead, a cache discard policy must be supplied to selectively drop entities from the cache to restrict its memory usage. Designing this discard policy to reflect common MIR data access patterns is important to maintain a useful level of cache performance.

Currently, a simple discard policy has been implemented. This is based on the observation that the only items of data which can consume a significant amount of memory are those which relate to the enactment of a workflow. The amount of memory consumed by other items is relatively small. As such, the discard policy specifies a maximum number of sets of enactment data that can be cached. If a request is made for a set of

enactment data which is not in the cache, and if the maximum number of sets of data are already cached, then the set of data which was least recently accessed is dropped from the cache to free up space, into which the newly requested set of data can be downloaded.

This discard policy is simple, and may not be good enough in some situations. However, it has so far proven to be effective in a typical server environment. The simple cache policy is aided by the use of a per-user cache that is destroyed when a user logs out, thereby freeing up space in memory.

4.4 Rendering of enactment data

Workflows that can be enacted in the MPI are limited to those that accept plain text values as inputs, and which produce plain text, images or HTML as results. Users can view inputs and results of a particular enactment by using the hyperlinks on a summary page for an enactment, as shown in figure 3 above, to navigate to a page capable of rendering the chosen item of data.

Different types of data are rendered using different HTML elements. Plain text is rendered inside an HTML

<TEXTAREA> element. Images are rendered using an HTML element, whose *src* attribute points to a servlet capable of rendering the image.

HTML produced by workflows can be slightly more difficult to render. One feature of the current myGrid system is that the enactment of a workflow can generate HTML containing hyperlinks to other items of data produced during the same workflow enactment. Figure 6 below shows an example of such a hyperlink.

```
<A HREF="urn:lsid:lsdocument:
A3434355">link text</A>
```

Figure 6: Example of a hyperlink to an item of enactment data

The string `urn:lsid:lsdocument:A3434355` is an internal identifier which has been assigned to an item of data produced during the same workflow enactment. The MPI renders such HTML by replacing this identifier with the URL of an MPI web-page capable of rendering this item of data. This mechanism is useful, because it means that workflows can be modified to produce not just items of data but also HTML visualizations which refer to these items of data. As an example, the image shown in figure 5 above is involved in an HTML visualization. As part of the same enactment, HTML was produced con-

taining a <MAP> element which defines which regions of this image should be clickable. In this case, the clickable regions are the small vertical bars which represent interesting features of a DNA sequence. Users can click on one of these vertical bars to reveal further textual information about this feature.

5 Discussion

The MPI which has been described in this paper is the first working prototype of a portal which allows the publishing and enactment of pre-written workflows. Implementation has been simplified by focusing the design on support for users who work in small groups, and who wish to publish only small numbers of workflows. It is hoped that even with these constraints on design, the MPI will still be useful to much of the potential user community. However, it may be the case that some users do work in larger, more dynamic groupings, and this section attempts to describe how this design might be improved to support these types of user.

5.1 Improving mechanism for locating workflows

As the number of users of the MPI increases, it is likely that the number of workflows that have been uploaded into it will also increase. Of course, even a small group of users could upload a substantial number of workflows. Presenting available workflows in one large list can make it difficult for a user to locate a particular workflow that they are looking for. The MPI attempts to provide a limited hierarchical structure to make the management of workflows easier by allowing named workflows to be grouped into named collections. However, for larger numbers of workflows this one-level hierarchy may be too rigid.

One simple solution to this problem would be to add extra levels of nesting to the hierarchy, or to allow an arbitrary number of hierarchical levels, much like directories in a file system. This approach is already supported by the MIR, so would just involve changes to the interface design of the MPI. However, making these changes would necessarily increase the complexity of the interface.

An alternative approach would be to allow the visibility of workflows to be limited so that users only see workflows that they are interested in. This might be achieved by simply marking a workflow as only being visible to the user that uploaded it. Sharing of workflows could still be supported by allowing the status of a workflow to be changed so that it becomes globally visible. A

more advanced design might provide finer-grained role-based access control (RBAC), which might allow users to limit the visibility of workflows to members of dynamic subsets of users of a particular MPI installation.

Implementing these features in an interface would require a backend storage device that supports RBAC. The current MIR implementation does not include any such support, but future projects intend to add this. An alternative might be to use the Storage Resource Broker (SRB) [14, 8], a remote file store which provides the organization of users into dynamic groupings, and which has provides a number of flexible access control methods.

The provision of some form of RBAC would also be dependent upon the existence of a security architecture that supports the authentication of users. Ideally, this security architecture would allow a user to authenticate themselves through the Gridsphere login system. They might then be granted a temporary token which would allow them to authenticate using a security mechanism provided by the chosen storage solution. A candidate technology for this security architecture is the Grid Security Infrastructure (GSI) [13]. Gridsphere provides support for GSI through the GridPortlets application [1] and SRB allows users to authenticate themselves via GSI. Exactly how to use these facilities to construct a security architecture is an issue for further research.

An alternative mechanism which might allow workflows to be located more easily might be the provision of features which allowed users to construct simple queries for workflows. Users might be allowed to search for workflows, for example, which they had uploaded in the last week, which inputs or outputs with a given name, or which used a particular service. Again, the provision of a querying mechanism would be dependent upon support for querying in the backend data store. The current MIR uses a database for storage, and queries over this database could be constructed in SQL. If SRB were to be used as a replacement, then this also provides facilities for the construction of queries over metadata attached to individual files and directories.

5.2 Caching

The present portal prototype constructs a cache per user, which has a simple discard policy and which is destroyed when the user logs out.

As more users are added to the portal interface, this approach becomes less efficient, as it becomes more likely that multiple users will cache the same item of data. This means that with a large num-

ber of users, even a well-designed per-user discard policy might not be sufficient to avoid memory overruns. Destroying a cache that has been created for a user when this user logs out is also inefficient, as regularly accessed data will have to be quickly reloaded into a new cache when the user logs in again.

It seems that a better caching strategy would maintain one cache for all portal users, and for this cache to persist even when users logs out. Design of a discard policy for this cache might be difficult, and should aim to avoid penalizing individual users of the portal interface by dropping their data too regularly whilst ensuring that excessive memory usage is avoided.

6 Conclusion

This paper has described the design and implementation of an initial prototype of a simple, portal-based interface to *myGrid* workflow enactment and storage technology. This interface has been designed to allow small groups of users to share a set of workflows, which they can enact, producing data which is also shared. It is hoped that this interface will be a useful addition to the existing selection of *myGrid* software.

At the time of writing, this interface has not yet been trialled by a significant number of members of the user community. It is hoped that such trialling will take place, and that this will lead to improvements in the interface design.

7 Acknowledgements

The authors would like to acknowledge the assistance of the whole *myGrid* consortium. This work is supported by the UK e-Science programme EPSRC grant GR/R67743.

References

- [1] GridPortlets application home-page (Verified 28/06/2005).
<http://www.opengridportals.org/space/Toolkits/GridPortlets>.
- [2] Gridsphere portal home-page (Verified 28/06/2005).
<http://www.gridsphere.org/gridsphere/gridsphere>.
- [3] HPC-Europa home-page (Verified 28/06/2005).
<http://www.hpc-europa.org/>.
- [4] Jetspeed-2 home-page (Verified 28/06/2005).

- <http://portals.apache.org/jetspeed-2/>.
- [5] JSR-168 portlet interface specification (Verified 28/06/2005).
<http://www.jcp.org/en/jsr/detail?id=168>.
- [6] RFC 2617 - HTTP Authentication (Verified 28/06/2005).
<http://www.faqs.org/rfcs/rfc2617.html>.
- [7] Simple Object Access Protocol specification (Verified 28/06/2005).
<http://www.w3.org/TR/soap/>.
- [8] Storage Resource Broker home-page.
<http://www.sdsc.edu/srb/>.
- [9] Taverna workflow workbench home-page (Verified 28/06/2005).
<http://taverna.sourceforge.net/>.
- [10] uPortal home-page (Verified 28/06/2005).
<http://www.uportal.org/>.
- [11] Web Services Description Language specification (Verified 28/06/2005).
<http://www.w3.org/TR/wsdl>.
- [12] Web-Services specification (Verified 28/06/2005).
<http://www.w3.org/2002/ws/>.
- [13] W. Allcock, A. Chervenak, I. Foster, L. Pearlman, V. Welch, and M. Wilde. Globus toolkit support for distributed data-intensive science. *Computing in High Energy and Nuclear Physics*, 2001.
- [14] Arcot Rajasekar et al. Storage Resource Broker - managing distributed data in a grid. *Computer Society of India Journal, Special Issue on SAN*, 33(4):42–54, 2003.
- [15] Matthew Addis et al. Experiences with e-science workflow specification and enactment in bioinformatics. pages p.459–467. *UK e-Science All Hands*, 2003. ISBN - 1-904425-11-9.
- [16] P. Donachy et al. Genegrid: Grid based virtual bioinformatics laboratory. *UK e-Science All Hands*, 2003.
- [17] R. Stevens et al. Exploring Williams Beuren Syndrome using *mygrid*. pages i303–i310. *12th International Conference on Intelligent Systems in Molecular Biology*, 2004. published *Bioinformatics Vol. 20 Suppl. 1*.
- [18] Tom Oinn et al. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
- [19] Carole Goble, Stephen Pettifer, Robert Stevens, and Chris Greenhalgh. Knowledge integration: *In Silico* experiments in bioinformatics. *The Grid: Blueprint for a New Computing Infrastructure*, pages 121–134, 2003.
- [20] Jason Novotny, Michael Russell, and Oliver Wehrens. Gridsphere: An advanced portal framework. *EUROMICRO Special Session on Advances in Web Computing*, 2004.