Databases and ontologies

Integrating ARC grid middleware with Taverna workflows
Hajo N. Krabbenhöft, Steffen Möller and Daniel Bayer*
Institute for Neuro- and Bioinformatics, University of Lübeck, 23538 Lübeck, Germany

Received on January 21, 2008; revised on February 19, 2008; accepted on March 6, 2008
Advance Access publication March 19, 2008
Associate Editor: Alfonso Valencia

ABSTRACT
Summary: This work presents two independent approaches for a seamless integration of computational grids with the bioinformatics workflow suite Taverna. These are supported by a unique relational database to link applications with grid resources and presents these as workflow elements. A web portal facilitates its collaborative maintenance. The first approach implements a gateway service to handle authentication certificates and all communication with the grid. It reads the database to spawn web services for workflow elements which are in turn used by Taverna. The second approach lets Taverna communicate with the grid on its own, by means of a newly developed plug-in. It reads the database and executes the needed tasks directly on the grid. While the gateway service is non-intrusive, the plug-in has technical advantages, e.g. by allowing data to remain on the grid while being passed between workflow elements.
Availability: http://grid.inb.uni-luebeck.de/
Contact: bayer@inb.uni-luebeck.de

1 INTRODUCTION
Biological processes are complex. Consequently, an enormous number of specialized applications and databases are in use today. There is a strong demand for software tools supporting information integration in such a way that a multitude of applications can be used in a single in silico experiment (Merelli et al., 2007). The workflow environment Taverna (Stevens et al., 2004) offers access to hundreds of today’s most prominent web services.

Today’s more challenging problems, e.g. in statistical genetics, are computationally very demanding or work with extremely large datasets. These cannot be addressed with public web services alone. Many groups in bioinformatics have a computational cluster on their own or share one, whereas most scientists in the field do not have direct access. Local or international grid computing initiatives allow communities to share resources and ease collaborations between biological and bioinformatical research.

This work presents a seamless integration of these resources in the workflow management software Taverna. It is exemplified on the ARC grid middleware (Ellert et al., 2007) which uses libraries of the Globus Toolkit (Foster, 2006). ARC is employed by multiple grids throughout Europe (Eerola et al., 2003; Podvinec et al., 2006) and beyond.

2 APPROACH
A distinguishing feature of ARC is the support of runtime environments (REs), that encapsulate software packages. The REs required by a job are specified in its xRLS description (Smirnova, 2007). Usually, a grid site’s administrator manually installs prepared packages to provide REs, but recent progress (Bayer et al., 2007) allows REs to be deployed automatically.

The link between REs, templates for grid jobs and workflow elements is established by the preparation of short XML files that outline use cases (Fig. 1). They name the input (line 4–6) and the output (line 7–9), specify the command line (line 3) and list the needed REs (line 10). Given this information, a grid job description can be deduced.

A web portal is provided to support the user community in exchanging use cases and allowing collaborative maintenance. This concept reflects that of myExperiment (de Roure et al., 2007) on the lower level of workflow elements rather than complete workflows. The use cases resemble the work of Kandaswamy and coworkers (Kandaswamy et al., 2006) in providing web interfaces without the need to perform any programming. This further leverages grid computing to application specialists who are not trained in web programming.

The remainder of this section describes two technical routes to achieve an integration of the use cases with Taverna (Fig. 2). While the first approach is non-intrusive and not bound to Taverna, the second approach is Taverna-specific, allowing a tight integration which has several advantages for the user.

2.1 Use cases as dynamically generated web services
Access to a computational grid can be organized via a web service (Foster, 2006). This principle was previously adopted for an integration of grid computing with workflows in SOAPlab (Senger et al., 2003). In analogy, the grid access from Taverna was implemented as a gateway with a web service interface. This presents the use cases to Taverna in a dynamically generated WSDL file. To execute jobs on the grid, the gateway needs the user’s grid proxy certificate. Taverna presents the operations in the WSDL file as ordinary workflow elements. Upon their invocation the gateway prepares and submits jobs to the grid. The gateway waits for the submitted job to finish and sends its result to the caller.

2.2 Use case scavenger and ARC grid processor
The second approach eliminates the need for a gateway service: a Taverna plug-in provides a processor to execute workflow elements on the grid and a scavenger to retrieve the use cases.

*To whom correspondence should be addressed.
Fig. 1. Example for a use case description for a sequence alignment with the program TCoffee (Notredame et al., 2000). It shows the command line to be executed, references a set of needed RE and names the input and output files.

```xml
<program name="t_coffee">
  <description>:align mult. FASTA seq:
  <command="t_coffee -infile in -outfile out">
  <input name="multiple_sequences_FASTA">
  <file path="in"/>
  </input>
  <input name="alignment">
  <fromfile path="out"/>
  </input>
  <output name="alignment">
  <tofile path="out"/>
  </output>
  <RE name="APPS/BIO/TCOFFEE-5.05"/>
</program>
```

Fig. 2. Schematic representation of the web service approach (left) and Taverna’s grid processor (right). The gateway retrieves use case descriptions and presents these in WSDL format to Taverna. The requested operations are transformed into grid jobs which are controlled by the gateway. The plugin to Taverna instead performs the job control itself.

The scavenger collects the use cases from the portal and presents them in Taverna’s services list. The Graphical User Interface (GUI) allows the user to select the proxy certificate and lists those grid sites and storage elements which are accredited by the given certificate.

The processor can access the grid directly. It supports the handling of grid certificates, job creation and the retrieval of status information. This saves the detour in using a web service. Results are returned as special reference objects which download the actual content on-demand. Whenever the user chains grid processors in Taverna, this allows to pass the output of the first job on the grid to the second job. So it is not necessary to download and re-upload the possibly large intermediate dataset. The data will be retrieved on-demand when a user decides to inspect the data in Taverna or it is needed as input for a non-grid processor.

3 METHODS

The work presented here built upon unmodified Taverna versions 1.6.2 and 1.7.0. The plug-in and the gateway service are implemented in Java on Debian Linux. The gateway web services were run on Tomcat 5.5. The use case portal was implemented using PHP and MySQL. The complete source code and a description of the technical details are available at above URL.

4 DISCUSSION

To offer a web service to the community may be too demanding for small research groups. The approach presented here separates the actual program from the resources needed. This allows every researcher to easily publish software as a use case without worrying about resources. These will be dynamically allocated based on the user’s grid certificate and grid organization memberships.

The Taverna plugin is superior to the gateway in two ways. Firstly, it allows to handle data more efficiently. Secondly, it does not require to transfer the grid proxy certificate to an additional remote instance. As these credentials allow to submit arbitrary jobs, the user has to trust the gateway to keep his certificate secure.

The presented approach is not specific to ARC. Most parts of the code are reusable for other Globus-based middlewares. The scavenger does not depend on grid technologies. If the concept of runtime environments is not available, it might be mimicked by a list of suitable sites.

5 CONCLUSIONS

This effort presents a seamless orchestration of web services and grid computing. It uniquely features the deep embedding of grid computing in Taverna and the remote handling of data between jobs. The implementation was provided for the ARC grid middleware but can be extended to support other Globus-based grid systems.

Computational biology spearheaded the public sharing of code and data. myExperiment brings in the sharing of conceptional knowledge and with the embracement of grid technologies this adds the sharing of resources.

ACKNOWLEDGEMENTS

This work was funded by EU FP6 IST project ‘KnowARC’. We thank Thomas Martinetz for his support and comments on the manuscript. The Taverna developers are thanked for their fine work and support.

Conflict of Interest: none declared.

REFERENCES


