

Workflow integration in VL-e medical

Tristan Glatard, Kamel Boulebiar, Silvia D. Olabariaga

University of Amsterdam, Institute of Informatics and Academic Medical Center
{glatard,boulebia,silvia}@science.uva.nl

Abstract—This paper presents the integration of a workflow management system into the VL-e Medical software architecture. Workflows are designed with the Taverna workbench, and then executed with the MOTEUR engine on the EGEE grid through the VBrowser, which is the basic front-end for grid-enabled applications in the VL-e Medical project. Data management is handled by the Virtual File System of the VL-e Toolkit. The resulting system provides a high-level interface to execute grid applications.

I. INTRODUCTION

Medical image analysis applications have been shown to benefit from the use of grid technologies for data/algorithm sharing and computing power. Yet, their adoption by non computer scientists is still low. One of the major reasons explaining this gap is the lack of high-level software interfaces, which refrains domain experts from adopting the grid in their daily routine. Most of the grid systems only offer low-level command-line interfaces, and a significant amount of manual operations are need to conduct an experiment to its end.

The goal of the VL-e Medical¹ subprogram of the Dutch Virtual Laboratory for e-Science (VL-e²) project is to tackle these problems in the field of medical image analysis applications. In the past two years, libraries and user interfaces for distributed data management have been developed and adopted by clinical researchers, thus bringing the grid closer to the end-users [1]. Experiments conducted on the VL-e computing infrastructure proved that the grid is powerful enough to tackle the considered compute-intensive problems, such as parameter sweeps, in a reasonable amount of time [2]. Yet, setting up these experiments is still in the hands of grid experts, to whom the data is given for computation. Results are inspected by the end-users in a second round. In addition to the delaying of the experiment cycle, this approach moves the clinical researchers away from the computation, reducing their confidence in the final results.

The use of a workflow system is expected to yield benefits in terms of higher-level interfaces to applications, thus bringing the application closer to the end-users. First of all, workflows constitute a suitable format to exchange applications among developers, end-users and grid experts. They would not only allow a transparent parallelization of the code (thanks to the intrinsically parallel nature of workflows) but also provide a friendly representation of the logic of the application to the end-user. Second, the use of workflows is expected to

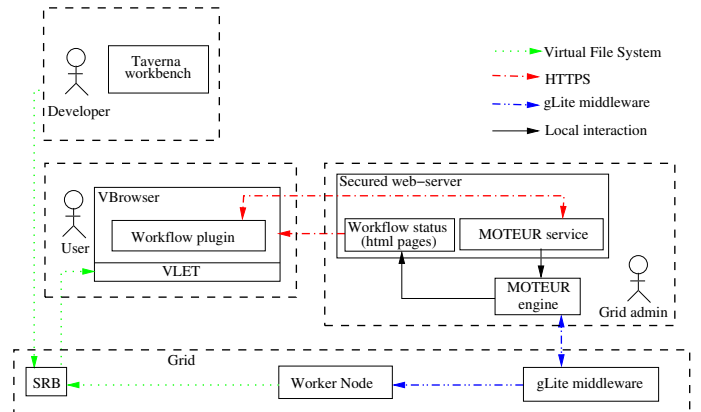


Fig. 1. Overall architecture of the VL-e Medical workflow system.

improve the management of large experiments, by providing a framework for application execution. It could allow intermediate results check-pointing, facilitating the detection of errors, which is hard to do with black-box executables. Last but not least, describing applications as workflows is promising in terms of performance: an increased level of parallelism can be achieved by splitting an existing monolithic code as a graph of functions. Concerning parameter sweep experiments, it could save a significant number of redundant computations. In the following we describe the integration of a workflow management system into the VL-e medical software architecture.

II. SYSTEM DESIGN

Motivated by the requirements of VL-e Medical use cases, the designed system (depicted on figure 1) integrates the MOTEUR workflow engine [3] with the Virtual Resource Browser (VBrowser) high-level data management and front-end [1]. In this system, the workflow and data management components are decoupled to achieve higher extensibility. Data management is handled by a generic interface adopted by all system components, including workflow management; therefore, the impact of introducing new data storage resources is minimized. Conversely, the workflow engine could be switched without any impact on the other components of the system. Below we motivate, present and discuss our design and implementation choices concerning the workflow language and engine, and data management.

¹<http://www.science.uva.nl/silvia/vlemed>

²<http://www.vl-e.nl>

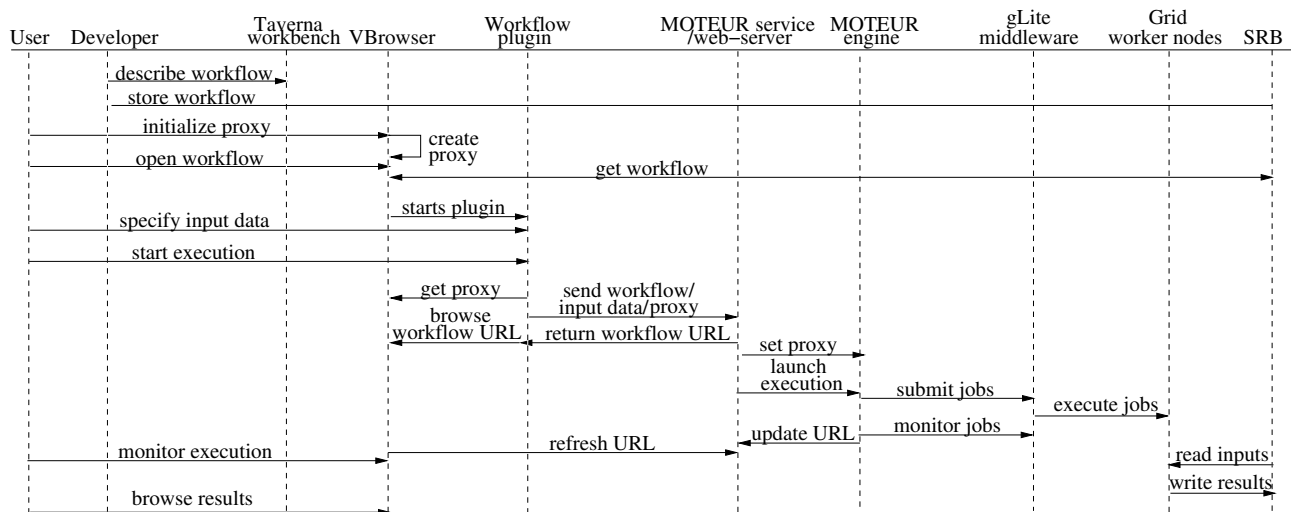


Fig. 2. Sequence diagram of a workflow execution.

A. Workflow language and engine

In VL-e medical, workflows are not only used to facilitate application parallelization, but also to facilitate application sharing among developers and users. An important requirement of a workflow language is therefore to *separate the functional description* of the workflow, built by the developer, *from the data* on which it runs, which is instantiated by the user. Additionally, the language should adopt a *graphical representation* of the workflow so that users have an insight about the functioning of the application they use. *Iterations* of functions over a large number of data items or parameter sets should also be easily described in the language. The workflow engine needs to run on the *production infrastructure* available for the vlemed Virtual Organisation (VO), which is part of the EGEE grid. This infrastructure is deployed over a WAN, with constraints such as port filtering between sites.

For the reasons presented above, we chose to adopt the Scuff language of the Taverna workbench [4], which provides functional description of workflows, as well as graphical representation and iteration strategies. Additionally, the MOTEUR engine [3] is adopted, since it enables running Scuff workflows on EGEE, with built-in fault-tolerance mechanisms such as timeout for job execution, automatic resubmissions and job grouping handling.

Other task-graph workflow approaches typically used in grid computing (e.g., DAGMan³ or Pegasus [5]), are more suitable for performance (in particular for scheduling), but they do not enable separation from functional description and data. Other visual workflow languages such as MoML (used by Kepler [6]) might constitute interesting alternatives to the Scuff language. Yet, iteration strategies offered by the Scuff language provide a very expressive way to handle medical data in the workflow.

B. Distributed data management

Our requirements concerning data management are (1) storage federation supporting an uniform view of distributed storage resources and (2) security, enabling file access control at user level to guarantee privacy of medical data.

To fulfil the first requirement, the system was built using the Virtual File System (VFS), which is part of the VL-e Toolkit (VLET⁴). The VFS provides a Java API to transparently access files on several storage systems. A single method is used to copy files from/to any location, independently from the data transfer protocol involved. The current VFS drivers support access to the local host, sftp, GridFTP, Globus Reliable File Transfer (RFT) and the San Diego Storage Resource Broker (SRB)⁵. The VFS manages grid certificates and sftp passwords so that file operations are performed on behalf of the user, ensuring a proper authentication. Recent work such as JavaGAT [7] constitutes an interesting alternative to the VFS.

A suitable security-level is obtained by using the SRB. Beyond GSI authentication, this system also supports user-level access control lists (ACL). Such a fine grained access control is mandatory for the medical use-cases, due to strict regulations of privacy in the manipulation of medical images. Moreover, it prevents users from ruining someone else's data by mistake, which might happen if only VO-level access control is present.

EGEE Storage Elements (SE) could enhance storage reliability in this architecture. In particular, fault-tolerance would be greatly improved by the replica management facilities offered by the gLite middleware. Their integration using the Medical Data Manager [8] to cope with security requirements is currently being considered.

³<http://www.cs.wisc.edu/condor/dagman/>

⁴<https://gforge.vl-e.nl/projects/vlet/>

⁵<http://www.sdsc.edu/srb>

C. System Integration

The overall architecture of the proposed software system is depicted on figure 1. The components are the *SRB*, used to store all data, workflows and monitoring information; the Taverna workbench, used for workflow development as a stand-alone application; the VBrower; the MOTEUR workflow service and engine; and the EGEE computing resource broker.

The VBrower provides a user-friendly front-end with a GUI that allow users to browse local and remote resources. It is built based on the VLET, which offers an abstract layer to a large variety of middleware, including for data resources. The VBrower is extensible and as such provides a convenient platform to develop application front-ends that are “easy” to use by the VL-e Medical target users. Plug-ins can be associated to specific file types to invoke external services. For implementing the system described here, a new VBrower plug-in was developed to enable the users to specify input data to an existing Scufi workflow document. This plug-in offers basic parameter sweep facilities such as parameter ranges or lists. It is interfaced with the MOTEUR service over an HTTPs connection used to send the Scufi workflow, the input data set to be processed (URIs in case of files), and the user proxy.

The MOTEUR service instantiates an engine that can submit and monitor EGEE jobs using the user credentials. The I/O operations are implemented with the VFS, therefore all the files involved in a workflow execution (workflow description inputs and results) are directly available for inspection in the VBrower. Legacy applications corresponding to workflow components are transparently encapsulated into EGEE jobs by using the Generic Application Service Wrapper of the MOTEUR engine. The workflow status is maintained through a URL, thus enabling the user to monitor the workflow execution with the VBrower. The sequence diagram of a workflow development and execution cycle using this system is depicted on figure 2.

A screenshot of the VBrower running a Scufi workflow on EGEE using MOTEUR is shown in figure 3. In this architecture, the end-user interacts at a high-level, with the VBrower only. Thanks to the workflow approach, new grid applications are made available by the developers as simple documents opened with the same interface (the plug-in). Apart from the proxy generation that has to be done by the user through the VBrower, the whole grid configuration is delegated behind the MOTEUR service. The grid administrator can then configure the workflow execution parameters (such as job retry counts or Resource Broker selection) without any burden for the user.

III. CONCLUSION

We presented the integration of a workflow management system (MOTEUR) into the VL-e medical software architecture (VBrower). The workflow description relies on the Scufi language used in Taverna. Users can launch and monitor workflow executions from the high-level interface of the VBrower. Behind that, a workflow service enacts and monitors grid jobs on behalf of the user through the MOTEUR engine. The VFS has greatly facilitated the interface between the MOTEUR

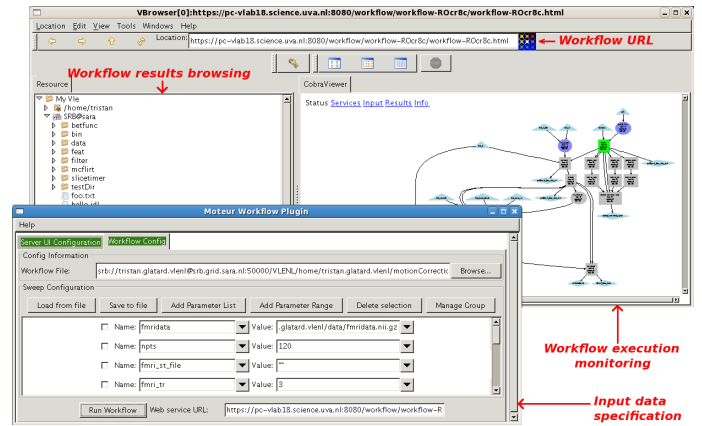


Fig. 3. A Scufi workflow designed with Taverna and running in the VBrower on the EGEE grid using the MOTEUR engine

engine and the VBrower. It allows workflow inputs/outputs to be browsed by the users in their usual environment. Besides, the plug-able design of the VBrower allows us to easily provide new features to the end-users. The resulting system offers a high-level interface to execute grid applications.

IV. ACKNOWLEDGEMENT

This work is funded by the Virtual Lab for e-Science (VL-e). We thank P.T. de Boer for support on the usage and extension of the VBrower. We are also grateful to the NIKHEF’s and SARA’s grid support teams for providing user and development assistance.

REFERENCES

- [1] S. Olabarriaga, P. T. de Boer, K. Maheshwari, A. Belloum, J. Snel, A. Nederveen, and M. Bouwhuis, “Virtual Lab for fMRI: Bridging the Usability Gap,” in *e-Science’06*, Amsterdam, Dec. 2006.
- [2] S. Olabarriaga, A. Nederveen, and B. O’Nuallain, “Parameter Sweeps for Functional MRI Research in the Virtual Laboratory for e-Science Project,” in *Biogrid’07*, Rio de Janeiro, May 2007, pp. 685–690.
- [3] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec, “Flexible and efficient workflow deployment of data-intensive applications on grids with MOTEUR,” *IJHPCA*, 2008.
- [4] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, “Taverna: A tool for the composition and enactment of bioinformatics workflows,” *Bioinformatics journal*, vol. 17, no. 20, pp. 3045–3054, 2004.
- [5] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbre, R. Cavanaugh, and S. Koranda, “Mapping Abstract Complex Workflows onto Grid Environments,” *JGC*, vol. 1, no. 1, pp. 9–23, 2003.
- [6] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, “Scientific Workflow Management and the Kepler System,” *Concurrency and Computation: Practice & Experience*, vol. 18, no. 10, pp. 1039 – 1065, Aug. 2006.
- [7] R. van Nieuwpoort, T. Kielmann, and H. Bal, “User-friendly and reliable grid computing based on imperfect middleware,” in *SC’07*, Nov. 2007.
- [8] J. Montagnat, Á. Frohner, D. Jouvenot, C. Pera, P. Kunszt, B. Koblitz, N. Santos, C. Loomis, R. Texier, D. Lingrand, P. Guio, R. Brito Da Rocha, A. Sobreira de Almeida, and Z. Farkas, “A Secure Grid Medical Data Manager Interfaced to the gLite Middleware,” *JGC*, 2007.