Development and Execution of Collaborative Application on the ViroLab Virtual Laboratory

Marek Kasztelnik[3], Tomasz Gubała[2,3], Maciej Malawski[1], and Marian Bubak[1,3]

[1] Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[3] ACC CYFRONET AGH, Kraków, ul. Nawojki 11, 30-950 Kraków, Poland
email: m.kasztelnik@cyfronet.pl

Abstract

Creating applications that use distributed Grid resources is a complex and time-consuming process. To help developers and end users to create, test and execute this kind of applications, the integrated environment is needed. This paper shows how to develop and execute collaborative applications on the ViroLab virtual laboratory. Furthermore, the collaboration tools which allow to communicate between end users (scientists) and developers are presented.

Keywords: virtual laboratory, collaborative applications, collaborative environment, ViroLab, e-Science experiments, grid

1 Introduction

Modern practices of science in such area as investigation of HIV virus drug resistance require collaborative sharing, processing and analysing of virological, immunological, clinical and experimental data, as well as advanced tools for (bio) statistical analysis, visualization, modelling and simulation [1]. A process integrating these computational tools and data, which leads to obtaining results relevant to the application domain, is called *in-silico* experiment. An experiment in virtual laboratory combines data and activities which are available on the distributed Web- and Grid-based infrastructure and it needs to orchestrate them in possibly complex scenarios.

A common approach to experiment orchestration is to use one of many scientific workflow systems available for the Grid, such as Pegasus [2], Triana [3] and K-WfGrid [4] systems. They are intended to assist non-programmer users in developing applications, however, in the case of workflows with many components and complex interactions, they can become difficult to understand and use.

To overcome the limitations of workflow systems, we decided to define an experiment plan notation based on a high-level scripting language, namely Ruby [5]. An experiment plan is a Ruby script which features a concise and clear syntax combined with a full set of control structures, allowing expressing experiments of arbitrary complexity level.

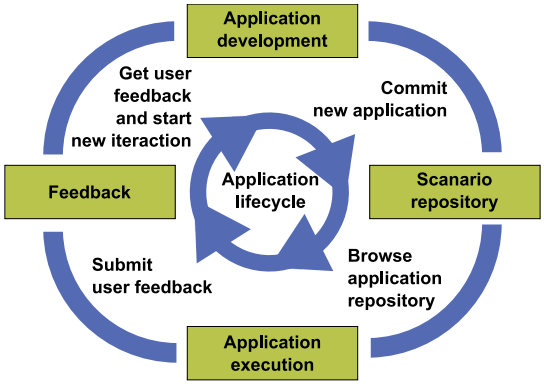

Fig. 1: Application lifecycle

The process of experiment planning and execution in the ViroLab virtual laboratory is collaborative in the sense that the virtual laboratory supports cooperation of multiple experiment developers and users (see Fig. 1).

2 Experiment planning

When a developer prepares an experiment script, it can be published in the experiment repository and thus become available to others. Then, a scientist, who does not intend to get into the details of scripting, can access the virtual laboratory through a portal, and execute the published experiments using Web browser, providing only input data when necessary. The main idea is that the experiment in the repository can be shared and reused what is a very efficient way of promoting collaboration between scientists. Provenance data related to the experiment is also recorded and available for queries thus making the results more reliable, reproducible and scientifically relevant.

The first stage of experiment lifecycle is performed by the experiment developer, whose task (with the assistance of the domain researcher) is to develop an experiment plan using a scripting notation. To hide sophisticated details of the underlying grid infrastructure, a high level object-oriented API has been introduced. It allows to define "which" computational functionality is required, without a need to specify "how" to access it with available middleware. Uniform access to computational resources in a Grid environment is possible due to three levels of abstraction that describe resources, namely Grid Object Class, Grid Object implementation and Grid Object Instance. While creating experiment, only the highest level of resources description may be used – on the other hand, however, if the developer needs to retain a full control over the experiment plan, it is possible to specify all the technical details on one of the lower levels of abstraction. When a resource is registered in the virtual laboratory, it is available for the whole community and other developers can reuse it in new applications.

Another feature, provided by the ViroLab virtual laboratory, which is useful

during development, is the high-level API that allows to connect and query various databases. Thanks to OGSA-DAI [9] system, that is accessible through this API, applications are able to query for the data located in distributed data sources. It is a very important tool that allows users to share the experiment results with the whole community.

When preparing the experiment plan, the experiment developer uses the Experiment Planning Environment (EPE) [10] based on the Eclipse [11] platform which offers user-friendly, integrated with a set of tools, editor for writing scripts. The developer can use the semantic-web based Domain Ontology Store graphical browser to discover available data and computational services, coupled with Grid Resources Registry which provides the available operations that can be invoked directly from a script. To facilitate collaboration, EPE is integrated with the Experiment Repository based on the Subversion (SVN) version control system – as a result many developers can work on single experiment or share experiment codes. When the experiment is ready (it fulfills all requirements of an end user) the developer uses dedicated EPE wizard to release it. When it is released, it instantly becomes available through the web interface for the scientific communitie. Moreover, clear releasing and versioning policy is very important for provenance data that has to be connected with specific version of the experiment.

3 Experiment execution

A script can be executed during development phase from the EPE which is integrated with the GridSpace Engine [12] (GSEngine, Fig. 2) which acts as a core of the runtime system. GSEngine includes the Grid Operation Invoker [13, 14] which translates high-level operations specified in the script into concrete invocations on computational resources using appropriate technologies.

After an experiment plan is developed, tested and released, a scientist can use a dedicated web interface (Experiment Management Interface - EMI [10]) for executing the experiment. The main advantage of the web based interface is that the scientist does not have to have any additional software installed (only a web browser with Java Script support is required). This tool hides the whole complexity of technology used underneath. Scientists may browse released applications, can see their documentation and execute them. EMI is also connected with the Gridspace Engine that is able to connect to SVN repository, download the experiment and run it. While application executes, data for the PROToS provenance system [15, 16] is recorded and stored into a dedicated provenance storage. This information can be searched by scientists through QUery TRanslation tOols (QUaTRO) [16, 17] - web interface dedicated for searching provenance data.

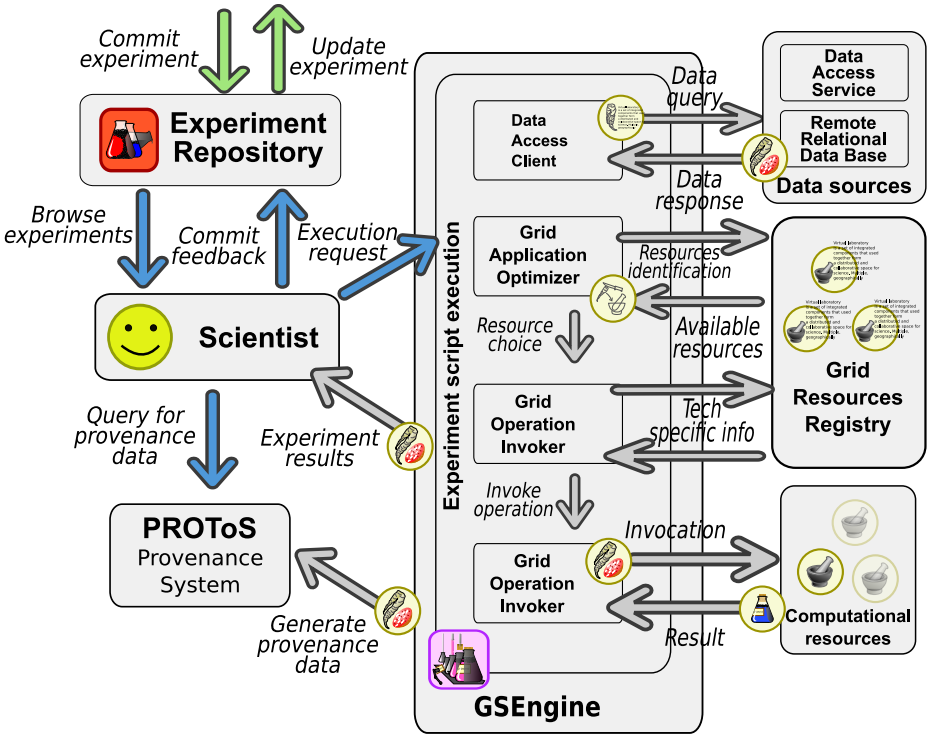

Fig. 2: Execution of a collaborative application on ViroLab virtual laboratory

4 Feedback from the end user to the experiment developer

Experiment development is a long-lasting process: bugs may occur in the code, not every user's requirement works correctly, additional enhancements should be implemented, etc. These problems, in most cases, are discovered by the end user of the application during its execution. The ViroLab virtual laboratory supports solving this kind of problems by simplifying process of communication between the experiment user and its developer. EMI allows the end user to submit *feedback* that can be useful for the developer who is able to browse it using EPE, respond to it and take it into account when creating a new version of the experiment. Scientists can track progress of the new version of the experiment development using simple SVN client (as a standalone or web based application) and help developer during this phase (with e.g. further comments).

5 Conclusions and future work

The unique feature of the virtual laboratory developed for ViroLab is that by providing a set of user friendly tools, both advanced experiment developers and domain scientists can productively collaborate and conduct their research

in modern highly distributed environment. Thanks to scripting language it is possible to define even complex experiments easily, still remaining on a high-level of abstraction and concealing the details of underlying grid middleware.

Currently the first prototype of the integrated virtual laboratory is released, installed and accessible by the experiment developers and the scientists (see [18] for software download and access to the Experiment Management Environment) who start to set up communities that share the data, resources, created experiments scripts and the knowledge.

Future work will concentrate on providing additional methodologies and tools that allow to create applications based on the data, resources and experience of the community. It is worth mentioning that those applications are created by many developers and are available for many scientists. Currently the work on management of results produced by experiments is in progress. Afterwards, data will be connected with ontological description of the environment. Consequently, finding interesting information will be easy and connecting it with provenance data will allow to track back the origin of this data. This functionality is very important for the virologists who are the main end users of the ViroLab virtual laboratory.

Acknowledgements This work was partially funded by the European Commission, Project ViroLab IST-027446, the related Polish grant SPUB-M and the Foundation for Polish Science.

References

1. Peter M.A. Sloot, Ilkay Altintas, Marian Bubak, Charles A. Boucher: From Molecule to Man: Decision Support in Individualized E-Health; *IEEE Computer Society*, vol 39, no.11, pp. 40-46, Nov., 2006
2. Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In Grid Computing: *Second European AcrossGrids Conference, AxGrids*, volume 3165 of Lecture Notes in Computer Science, pages 11-20. Springer, 2004.
3. Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison: Visual grid workflow in Triana. *Journal of Grid Computing*, 3(3-4):153-169, September 2005.
4. Tomasz Gubała and Andreas Hoheisel: Highly dynamic workflow orchestration for scientific applications. In *CoreGRID Intergation Workshop* 2006 (CIW06), pages 309-320. ACC CYFRONET AGH, 2006.
5. Ruby programming language `http://www.ruby-lang.org`
6. Web Service `http://www.w3.org/2002/ws`
7. Web Service Resource Framework `http://www.oasis-open.org/committees/wsrf`
8. Maciej Malawski, Marian Bubak, Michał Placek, Dawid Kurzyniec, and Vaidy Sunderam: Experiments with distributed component computing across grid boundaries. In *Proceedings of the HPC-GECO/CompFrame workshop in conjunction with HPDC 2006*, Paris, France, 2006.
9. OGSA-DAI homepage `http://www.ogsadai.org.uk`

10. Włodzimierz Funika, Daniel Harężlak, Dariusz Król,Piotr Pęgiel, Marian Bubak Developer and User Interfaces to the Virolab Virtual Laboratory. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
11. Eclipse - an open development platform `www.eclipse.org`
12. Eryk Ciepiela, Joanna Kocot, Tomasz Gubała, Maciej Malawski, Marek Kasztelnik, Marian Bubak: Virtual Laboratory Engine - GridSpace Engine. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
13. Tomasz Bartyński, Marian Bubak, Tomasz Gubała, Maciej Malawski: Universal Grid Client: Grid Operation Invoker In *Proc. PPAM 2007, Seventh International Conference on Parallel Processing and Applied Mathematics*, LNCS, Gdansk, Poland, Sept. 2007. Springer. In print.
14. Maciej Malawski, Tomasz Bartyński, Marian Bubak: Invocation of Grid Operations in the ViroLab Virtual Laboratory. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
15. Bartosz Balis, Marian Bubak, and Jakub Wach: Provenance Tracking in the ViroLab Virtual Laboratory. In *Proc. PPAM 2007, Seventh International Conference on Parallel Processing and Applied Mathematics*, LNCS, Gdansk, Poland, Sept. 2007. Springer. In print.
16. Bartosz Bali, Marian Bubak, Michał Pelczar, Jakub Wach: Provenance Tracking and Querying in ViroLab. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
17. Bartosz Balis, Marian Bubak, Jakub Wach: User-Oriented Querying over Repositories of Data and Provenance. In *Proc. 3rd IEEE International Conference on e-Science and Grid Computing, e-Science 2007*, IEEE Computer Society, Bangalore, India, Dec. 2007.
18. The ViroLab Project Consortium. The ViroLab Virtual Laboratory Website, 2007. `http://virolab.cyfronet.pl`