GridSpace Engine of the ViroLab Virtual Laboratory

Ciepiela, E.; Kocot, J.; Gubala, T.; Malawski, M.; Kasztelnik, M.; Bubak, M.

Publication date
2008
Document Version
Final published version
Published in
Cracow'07 Grid Workshop: October 15-17, 2007 Cracow, Poland: proceedings

GridSpace Engine of the ViroLab Virtual Laboratory

Eryk Ciepiela[1], Joanna Kocot[1], Tomasz Gubala[1,3] Maciej Malawski[1], Marek Kasztelnik[1], Marian Bubak[1,2]

[1] Academic Computer Center CYFRONET, ul. Nawojki 11, 30-950 Kraków, Poland
[2] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Kraków, Poland
[3] Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

GridSpace Engine is the central operational unit of the ViroLab Virtual Laboratory. This specific runtime environment enables access to computational and data resources by coordinating execution of experiments written in the Ruby programming language extended with virtual laboratory capabilities. Experiments harness published and semantically described services which constitute a *GridSpace*. The GridSpace Engine is a reliable service acting as an entry point to the Virtual Laboratory, its execution capabilities and a facade for specialized services such as Data Access Service. Moreover, owing to the provided dedicated libraries, the GridSpace Engine supports interactive execution and runtime monitoring of experiments. Furthermore, the GridSpace Engine is capable of retrieving experiment source not only from file systems but also from multiple *Application Repositories* accessed by dedicated adapters. Currently, our repository is based on the Subversion source code management and version control system. The GridSpace Engine is also responsible for storing obtained experimental results in the Laboratory Data Base.

Keywords: e-Science, virtual laboratory, grid resources coordination, experiment repository.

1 Introduction

Nowadays, virtual laboratories attract attention of scientists needing computational power and specialized software in order to process and analyze existing huge data sets, particularly in the field of bioinformatics. Moreover, such laboratories may successfully support specialists from a vast range of domains in making decision [1]. The aim of ViroLab [6] Virtual Laboratory [4, 5] is to provide high-quality environment dedicated to create, develop, manage, and run in-silico experiments in the domain of virology.

The role of GridSpace Engine within Virtual Laboratory is to provide experiment execution capabilities, so it can be considered an experiment execution engine or experiment enactment engine. The idea behind the GridSpace engine is to separate the wide range of client tools assisting experiment planning [8] from the engine that actually enacts the experiments. Even more significant

is that this distinction enables the GridSpace Engine to be a shared, reliable and efficient service independent of end-user machines, and as such dedicated to perform time-consuming experiments making use of grid resources. Such an approach delivers the experiment execution engine as a facility for dispersed groups of (possibly mobile) users who employ computationally intensive experiments in their research. For them, the GridSpace Engine constitutes a remote and stable entry point to the ViroLab Virtual Laboratory.

2 State of the art

Existing virtual laboratory solutions differ in the means of expressing experiments which in turn determines the way the experiment is enacted.

Projects like Kepler [10], myGrid Taverna [11] or Triana [12] use workflows to model experiments e.g. through Modeling Markup Language (MoML) used in Kepler. Therefore, their enactment engines are provided with declarative description which has to be done without explicitly specifying how to do this, and such a description is often assembled using graphical tools. An alternative approach, also undertaken in the ViroLab Virtual Laboratory, is to use scripts in order to express experiments in a more imperative manner. In this approach evaluation of experiments is performed by script interpreters such as Matlab in the case of Geodise [13] project.

The other major issue is how to run such an execution engine on grid resources. Since GridSpace Engine is required to support interactive experiments, data streaming and monitoring of running experiments a suitable middleware technology has to be employed. Globus Toolkit [14] enables stateful WSRF services, as well as job management services, but the drawbacks are the large overhead of the whole toolkit and limited support for interactivity. Component frameworks such as MOCCA [15] can also be used for dynamic deployment of experiment on Grid resources.

3 Structure and functionality of GridSpace Engine

The goal of the GridSpace Engine is to combine capabilities of accessing computational and data resources, of interacting with experiment executor, and of evaluation of a "glue code" of experiment script. The experiment code, hereinafter called *experiment plan*, is entirely written in Ruby [16], and the capabilities offered by Virtual Laboratory are provided through specialized libraries written in this language. In this approach, the whole functionality and utilities of Virtual Laboratory are exposed to experiment developer and used from the level of experiment plan.

Since the GridSpace Engine uses the JRuby [17] implementation, pure Ruby code can access the Java programming language classes and vice versa. Therefore, some Virtual Laboratory-specific libraries are developed in Java, and can be seamlessly called form within the Ruby wrapping library. This brings into action

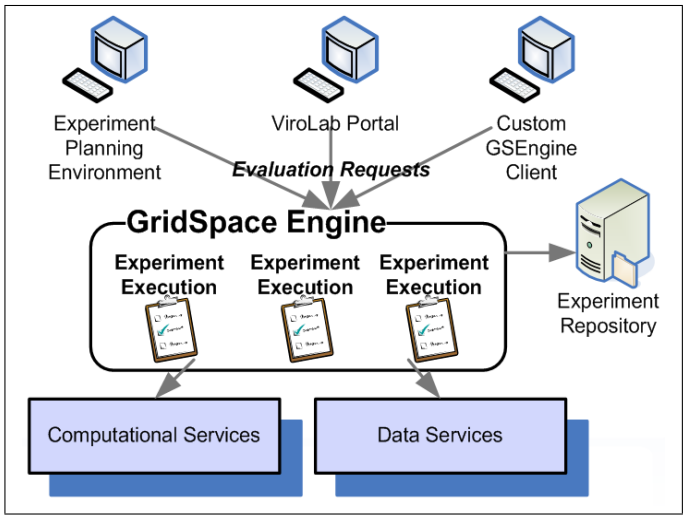

Fig. 1: GridSpace Engine placed in an environment of ViroLab Virtual Laboratory.

a vast range of Java-based libraries and utilities which can be easily incorporated in the execution engine.

The most essential libraries incorporated in the GridSpace Engine are responsible for access to grid resources. The Grid Operation Invoker (GOI) [2, 3] provides a library to invoke *grid operations* from Ruby code. Data Access Client (DAC), in turn, is a Ruby client for accessing Data Access Service [7] that integrates all data sources available in the Virtual Laboratory.

Besides these two, a user data input library is provided to enable experiments to interactively request the input from the user, which is especially desired in the case of decision support experiments. Furthermore, dealing with the outcome of experiments demands some means of streaming output of experiments and a library for sending results back to the user, with rendering modules on the client side.

The GridSpace Engine, including the JRuby interpreter, is accessed via API and can be called from any Java application. In particular, considering the Virtual Laboratory, it is used by specialized tools such as the Experiment Planning Environment [8] dedicated for experiments developers, and the Experiment Management Interface [8], which allowing experiment launching via the ViroLab Portal. Aside of these robust and complex tools, the GridSpace Engine comes along with a simple client in a the form of command line tool.

To keep the GridSpace Engine operable in the environment of Virtual Laboratory, and also to keep it generic there are a number of ways of providing the experiment plan to the engine. First of all, there is a way of programmatically passing the experiment plan via an API. However, to facilitate retrieving experiment plans from Experiment Repository a dedicated module called *GridSpace*

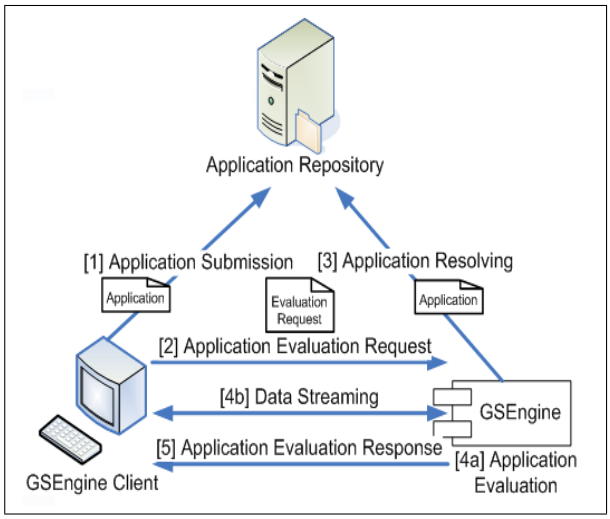

Fig. 2: A common scenario of using GridSpace Engine to execute experiement plans stored in the Experiment Repository.

Engine Application Repository Client is introduced. It enables plugging of clients for different implementations of Experiment Repositories.

GridSpace Engine placement within the Virtual Laboratory including specialized tools, laboratory resources and Experiment Repository is shown in Fig. 1.

A significant part of the functionality of the GridSpace Engine is to manage the session of the experiment execution. In this aspect, the engine preserves the scope of the user context (including security credentials) enabling Single Sign On (SSO) access to *grid objects*, as well as experiment execution context indispensable e.g. for the monitoring and provenance events correlation. The interactions between building blocks of the Virtual Laboratory which are involved in a common usage scenario of experiment execution are depicted in Fig. 2.

The GridSpace Engine is also intended to carry out on-line monitoring of experiment course including invocation of grid operations, access to data, current status of experiment plan execution, logging messages etc. These data are to be provided both to client tools requesting and tracing execution and to the monitoring infrastructure. Thereafter, historical information can be used to further optimize execution through choosing the most efficient grid objects by the GridSpace Application Optimizer (GrAppO) [9].

4 Implementation status

The current status of work (as of the end of September 2007) covers the first stable version of the GridSpace Engine implementation embeddable in a Java Virtual Machine of a client tool. Moreover, a command line tool is provided allowing users to evaluate experiments locally on their own machines. This

version of the engine was successfully integrated with Experiment Planning Environment and with a prototype of ViroLab Portal. Furthermore, GridSpace Engine already cooperates with SVN-based [18] implementation of Experiment Repository owing to a dedicated Application Repository Client adapter.

The GOI and DAC libraries are developed externally and independently, and have been also integrated with the engine. The libraries for handling user data input and result management are on the very early stages of their development roadmaps with only simple prototypes existing for feasibility studies. Monitoring features are not supported yet.

5 Summary and future work

GridSpace Engine considered as an experiment enactment service combines capabilities of Virtual Laboratory building blocks such as Grid Operation Invoker, Data Access Client, Experiment Repository and a number of dedicated libraries. The concept, design and prototype were verified by several fully functional applications that make use of computational resources, data resources and user data input library. The experiments include a *Genotype to drug ranking* application, supporting decision making by medical doctors who choose the most suitable drug to apply in HIV treatment.

The next step in the scope of this work will be to develop GridSpace Engine server remotely accessible through a interoperable protocol developed for clients written in diverse technologies. Future plans also include enrichment of GridSpace Engine with robust libraries for user data input and experiment result management. It is planned as well to enable online monitoring of running experiments along with integration with monitoring infrastructure of the Virtual Laboratory.

Acknowledgements. This work was supported by the EU Virolab project IST-027446 with 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. Tomasz Bartynski, Marian Bubak, Tomasz Gubala, Maciej Malawski: Universal Grid Client: Grid Operation Invoker *Proceedings of International Conference of Parallel Processing and Applied Mathematics (PPAM'07), Gdansk, September 2007, LNCS (to appear)*
3. Tomasz Bartynski, Maciej Malawski, Marian Bubak: Invocation of Grid Operations in the ViroLab Virtual Laboratory *In Proceedings of Cracow Grid Workshop 2007, this volume.*
4. ViroLab Virtual Laboratory, http://virolab.cyfronet.pl

5. Tomasz Gubala, Bartosz Balis, Maciej Malawski, Marek Kasztelnik, Piotr Nowakowski, Matthias Assel, Daniel Harezlak, Tomasz Bartynski, Joanna Kocot, Eryk Ciepiela, Dariusz Krol, Jakub Wach, Michal Pelczar, Wlodzimierz Funika, Marian Bubak: ViroLab Virtual Laboratory *In Proceedings of Cracow Grid Workshop 2007, this volume.*
6. ViroLab - EU IST STREP Project 027446, http://www.virolab.org
7. Matthias Assel, Bettina Krammer, and Aenne Loehden: Data Access and Virtualization within ViroLab *In Proceedings of Cracow Grid Workshop 2007, this volume.*
8. Wlodzimierz Funika, Daniel Harezlak, Dariusz Krol, Piotr Pegiel and Marian Bubak: User Interfaces of the Virolab Virtual Laboratory *In Proceedings of Cracow Grid Workshop 2007, this volume.*
9. Maciej Malawski, Joanna Kocot, Eryk Ciepiela, Marian Bubak: Optimization of Application Execution in the ViroLab Virtual Laboratory *In Proceedings of Cracow Grid Workshop 2007, this volume.*
10. Ilkay Altintas, Efrat Jaeger, Kai Lin, Bertram Ludaescher, and Ashraf Memon. A web service composition and deployment framework for scientific workflows. *ICWS*, 0:814, 2004.
11. R. Stevens et.al. Exploring williams-beuren syndrome using mygrid. *Bioinformatics*, 1(20):303–310, 2004.
12. 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.
13. Geodise project homepage http://www.geodise.org
14. The Globus Toolkit homepage, http://www.globus.org/toolkit/
15. M. Malawski, D. Kurzyniec, and V. Sunderam: MOCCA - towards a distributed CCA framework for metacomputing *In Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models - HIPS-HPGC, April 4-8, 2005, Denver, Colorado, USA, page 174a. IEEE Computer Society Press, 2005.*
16. Ruby programming language home page http://www.ruby-lang.org/
17. JRuby - Java powered Ruby implementation home page http://jruby.codehaus.org/
18. Subversion version control system home page http://subversion.tigris.org/