Invocation of Grid Operations in the ViroLab Virtual Laboratory

Tomasz Bartyński[2], Maciej Malawski[1], Marian Bubak[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Kraków, Poland
[2] Academic Computer Center CYFRONET, ul. Nawojki 11, 30-950 Kraków, Poland

Abstract

This paper presents invocation of grid operations within the ViroLab Virtual Laboratory. Virtual laboratory enables users to develop and execute experiments that access computational resources on the Grid exposed via various middleware technologies. An abstraction over the Grid environment is introduced which is based on the concept of grid objects accessible from the experiment script based on Ruby. We describe the Grid Operation Invoker library which is the core of virtual laboratory engine and provides access to heterogeneous computational resources in a uniform manner using pluggable adapters. Sample applications include a script, which implements a data mining scenario using the Weka library and combines Web services with MOCCA technologies.

Keywords: virtual laboratory, ViroLab, grid middleware, grid resource virtualization

1 Introduction

The diversity of middleware technologies which allow access to computational resources in Web and Grid infrastructures [1] results in the fact that utilizing them in a coordinated way becomes a challenge. This problem is also faced by the virtual laboratory [4] developed for the scientists involved in HIV drug resistance research in the scope of ViroLab [2] project. Experiment scenarios are written in a scripting language (Ruby) on a relatively high level of abstraction. Therefore there is a need to provide means to uniformly interface existing middleware technologies with a high-level object-oriented API.

There are four major types of technologies which can be used as the means to provide access to computation. The first one is the Web Service technology based on SOAP and WSDL which provides stateless remote invocation and messaging semantics. Then there is WSRF, which is based on extensions of Web Service providing standardized access to stateful entities (resources). The third one is the component technology implemented in such technologies as MOCCA [5] and ProActive [11]. The last type is a batch processing model that allows submitting jobs on Grid infrastructures like EGEE and DEISA.

2 Related work

GEODISE [8] introduces the concept of accessing the Grid computational resources within a Jython or Matlab script. There are toolboxes for these languages that enable to interface Condor, Globus as well as to manage certificates and submit jobs. This solution can not be reused because it relies on commercial software, for instance Matlab, Microsoft .Net or IBM WebSphere.

The Grid Application Toolkit [10] provides a simple, language neutral and invariant API for basic Grid use cases. It allows file manipulations, monitoring events and job management. The support for multiple middleware technologies is based on a concept of providing a dedicated adapter for every middleware technology. This project has recently evolved into Simple API for Grid Applications (SAGA). The approach presented by the GAT system is valuable, albeit it does not provide the high-level programming language required in the ViroLab virtual laboratory.

Knowledge based Workflow System for Grid Applications (K-Wf Grid) [9] facilitates construction of workflows with ontology-based semantic reasoning and executing them in the Grid environment. User presents his request in a formal way. Next, an abstract workflow is constructed. Optimal are selected and resources allocated from a wide range of candidates. Workflow composition as a way of developing applications is not suitable for developing experiments due to limitations in expressing experiment logic, although the concept of multiple levels of description of the application is very valuable.

3 Abstraction over the Grid environment

As a solution to the problem of interfacing diverse technologies, we introduce grid object abstraction level hierarchy (Fig. 1). Each grid object class is an abstract entity which defines a set of grid operations. These operations are invoked from the script, while the actual computation is performed on a remote machine. Each grid object class may have multiple implementations with different middleware technologies representing the same functionality. Each of the implementations may have multiple instances, possibly running on different resources, thus with different levels of performance. Grid object instances of a specific class may use a variety of middleware suites and therefore they must be interfaced in their specific protocols. Moreover, grid objects may have different properties, such as stateless or stateful interaction mode, synchronous or asynchronous operation invocation or being private or shared between experiments runs and users. Developers are not concerned about finding the optimal instance and interfacing it, however must be aware of grid object's properties. For instance, they must know whether a grid object they are using preserve state between invocations of operations.

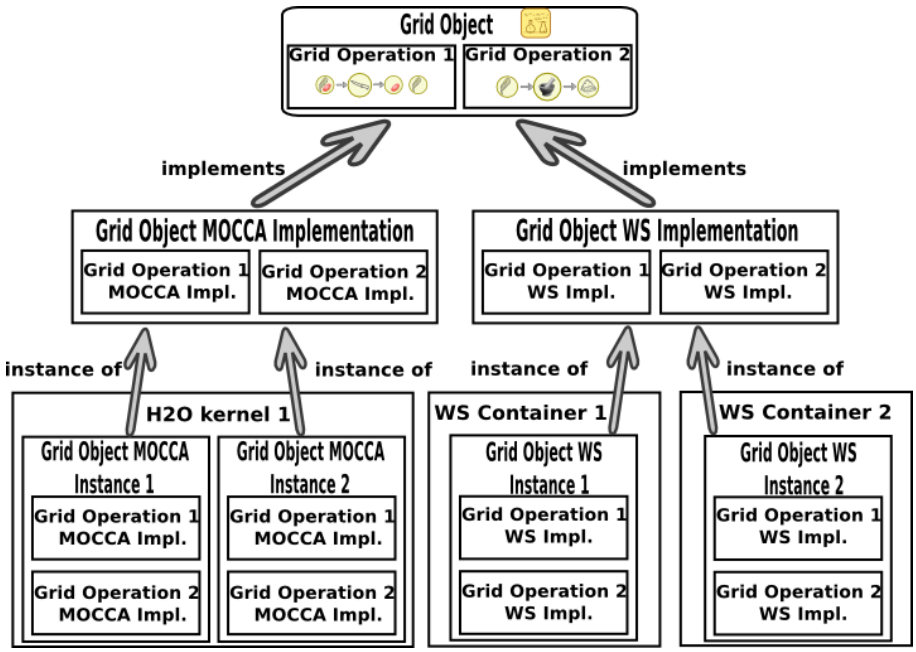

Fig. 1: Three levels of the abstraction over the Grid environment.

4 Grid Operation Invoker

In order to enable developers to use the abstraction described in Section 3, the virtual laboratory engine [12] includes as its core a Grid Operation Invoker (GOI) [3] module which is a lightweight client side library. Its goal is to instantiate grid object representatives (proxies) and to handle remote operation invocations using appropriate technologies. GOI handles different technologies by pluggable adapters implemented in JRuby, facilitating integration with Java client-side libraries. For job-based middleware, it is possible to create grid objects which expose the functionality through application-specific method invocation in object-oriented style. Every grid object instance is described with technology information that enables to use a dedicated adapter and interface it correctly during script execution (see Fig. 2). The Grid Operation Invoker cooperate with the following components of the virtual laboratory: Grid Application Optimizer and Grid Resource Registry. The former is used to find the best grid object instance for a given grid object class, while the latter provides technology information about instances. It is possible to configure the Grid Operation Invoker to use other optimizers and registries.

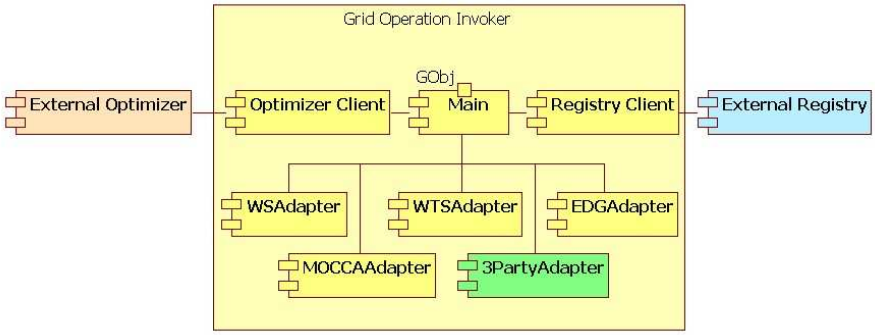

Fig. 2: Architecture of the Grid Operation Invoker.

5 Uniform interface to computational resources

Experiment developers use a simple uniform API for creating grid object representatives that hides the complexity of underlying middleware technologies, but provides full control if needed. They can use representatives within a script like any ordinary Ruby object. Representatives are created in three ways by:

- providing a name of a grid object class,
- providing unique grid object instance identifier,
- using low-level API.

In the first case, the process of selecting the computational resource to be used is hidden from the user. The GOI system queries the optimizer for an optimal instance and an identifier of an instance is returned. Next, the registry is queried for technology information for the selected instance. Subsequently, a class of an adapter for an instance is determined and loaded. Finally, a grid object representative is created and can be used within the script. In the second scenario developer chooses the instance and provides its identifier, therefore the optimizer is omitted. The rest of the process in identical to the former case. If developers decide to use a low-level API, they need to choose an appropriate adapter and provide technical information about the instance directly in the script.

6 Implementation status

The Grid Operation Invoker has been implemented and integrated with the virtual laboratory engine [4]. It uses a remote registry providing technology descriptions and an optimizer which assists in choosing the best instances of grid objects. Currently available adapters are for Web Service, MOCCA [5] components and LCG jobs. These technologies can be used within the script in a uniform manner.

```
1 require 'cyfronet/gridspace/goi/core/g_obj'
2 retriever =  GObj.create('virolab.weka.WekaGem')
3 classifier =  GObj.create(virolab.weka.OneRule')
4 data = retriever.loadDataFromDB(DB, QUERY, USER, PASSWORD)
5 classification = classifier.train(data)
```

Fig. 3: Using GOI API to interface Grid computational resources.

The GOI system enables to concisely develop applications accessing multiple middleware technologies. Fig. 3 presents a snippet of an experiment that uses two grid objects: a data retriever and a remote Weka [7] classifier employing One Rule algorithm. The retriever is used to load data from remote database in a specific format. Then the classifier is invoked for the data obtained. This application consists only of five lines of source code

1. `GObj` factory is required in the script,
2. `GObj` is used to produce a retriever (representative for a grid object of the `virolab.weka.WekaGem` grid object class),
3. `GObj` is used to produce a classifier (representative for a grid object of the `virolab.weka.OneRule` grid object class),
4. Grid object representative for the retriever (`retriever` variable) is used load data from the data base,
5. Grid object representatives for the classifier (`classifier` variable) is trained with obtained data.

Both, the `retriever` and the `classifier` are used like ordinary local Ruby objects.

The Grid Operation Invoker was applied to virology problems. ViroLab experiments, such as analysis of HIV genomic structure and prediction of resistance of virus to various types of drugs [6], were implemented using GOI API and can be executed in the virtual laboratory.

7 Summary and future work

In this paper we show how the Ruby scripting language combined with the GOI library are suitable for creating high-level experiments. This easy-to-use but powerful approach allows rapid development of collaborative grid applications using multiple middleware technologies. Initial experiments in the ViroLab virtual laboratory have successfully demonstrated the applicability of the proposed approach for virological applications.

Current work includes the development of adapters for WSRF, AHE and Unicore middleware suites. Future work will focus on support for asynchronous and parallel computations, as well as on integration with security, provenance [13] and monitoring subsystems of virtual laboratory. There are also prospects for interesting research in the area of supporting more application composition models, including direct connections between components, notifications ad streaming.

Acknowledgements. Acknowledgments: This work was supported by EU project ViroLab IST-027446 with the related Polish grant SPUB-M and by the Foundation for Polish Science.

References

1. Wolfgang Gentzsch: Major grid projects around the world, *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International Conference p. 1, 25-29 April 2006*
2. ViroLab – EU IST STREP Project 027446; www.virolab.org
3. Tomasz Bartyński, Marian Bubak, Tomasz Gubała, 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)*
4. ViroLab Virtual Laboratory, *http://virolab.cyfronet.pl*
5. 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.
6. 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
7. Ian H. Witten, Eibe Frank: Data Mining: Practical machine learning tools and techniques, *Morgan Kaufmann, San Francisco, 2005*
8. GEODISE Grid Enabled Optimisation and Design Search for Engeneering *http://www.geodise.org/*
9. Tomasz Gubała, Daniel Harężlak, Marian Bubak, Maciej Malawski: Semantic Composition of Scientific Workflows Based on the Petri Nets Formalism, E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, IEEE Computer Society, 2006
10. Grid Application Toolkit, *http://www.gridlab.org/WorkPackages/wp-1/*
11. ProActive - Parallel, Distributed, Multi-threaded solutions, *http://proactive.inria.fr/*
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. Bartosz Baliś, Marian Bubak, Michał Pelczar, Jakub Wach: Provenance Tracking and Querying in ViroLab. In *Proceedings of Cracow Grid Workshop 2007*, This volume.