

A Comprehensive Framework for Scientific Applications Execution on Distributed Computing Infrastructures

Javier Rojas Balderrama, Tram Truong Huu, Johan Montagnat

▶ **To cite this version:**

Javier Rojas Balderrama, Tram Truong Huu, Johan Montagnat. A Comprehensive Framework for Scientific Applications Execution on Distributed Computing Infrastructures. Rencontres Scientifiques France Grilles 2011, Sep 2011, Lyon, France. hal-00653021

HAL Id: hal-00653021
https://hal.science/hal-00653021

Submitted on 16 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Comprehensive Framework for Scientific Applications Execution on Distributed Computing Infrastructures

Javier Rojas Balderrama Tram Truong Huu Johan Montagnat

University of Nice–Sophia Antipolis/CNRS I3S Laboratory
{javier,tram,johan}@i3s.unice.fr

Abstract

This paper reports on the implementation of a comprehensive framework for design and execution of scientific applications on multiple distributed computing infrastructures. The architecture leverages available tools of the scientific community underlining the effort of the integration process. The framework brings together heterogeneous technologies in order to provide access to end–users transparently without the need to manage technical aspects. Such a working environment enables a highly configurable, services–oriented, and standards–based alternative for reproducible and scalable experimentation.

1 Introduction

Scientific applications are increasingly demanding in term of IT resources, simulation complexity, and scale of the consortia involved in their lifecycle. Distributed Computing Infrastructures (DCIs) have been developed worldwide to tackle these new challenges, promoting resources sharing, reuse, and computing capacity. Nowadays, several middlewares and tools for using those facilities are available, however the stack of multiple technologies and their different sources often add extra complexity to scientific applications development, although each one aims at facilitating it individually.

From a user's point of view an end–to–end framework enabling the exploitation of DCIs should integrate (1) high expressiveness to describe applications execution and their results, (2) design and enactment of applications composition making use of consistent interfaces, (3) transparent access and use of DCIs without entering directly in technical aspects like authentication, delegation, invocation, etc., and (4) applications sharing and reuse.

The goal of this work is the design and development of a comprehensive framework addressing the problem of developing scientific applications on DCIs, shielding users from the complexity of the underlying infrastructures, tools and technologies. We show the integration of several initiatives within an extensible environment implemented along the service–oriented principles. Section 2 describes the context of the work. Then section 3 presents concisely the architecture of the framework. Finally, section 4 closes with the results and discusses perspectives for the future work.

2 Enactment and execution of scientific applications

Porting large–scale applications to DCIs is not a straightforward task due to the inherent complexity of the distributed approach at the design phase, and the heterogeneity of the underlying infrastructures and their technologies in the implementation phase. Designing a scientific application enactment involves choosing the appropriate strategy to enable the efficient execution on large data sets. Within grid communities, scientific workflows were adopted to a large extent to formally represent applications parallelism. The workflow language allows users to describe the application composition logic at abstract level. The workflow engine interprets that workflow description to enact it. The configuration of an application, for execution on a DCI, often requires

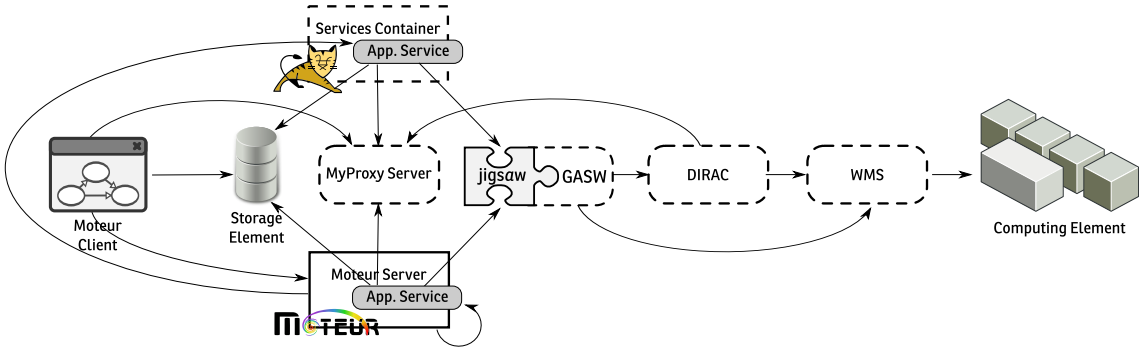


Figure 1: Overall framework architecture

extensive knowledge of the infrastructure. It is therefore a difficult task for non–expert end–users because it involves deploying applications and their dependencies on distributed resources; ensuring fault–tolerance in a highly heterogeneous environment; and addressing scalability issues.

Additionally, to migrate an application from one DCI to another requires an extra effort to ensure compatibility. Middleware incompatibilities prevent the straight exploitation of other DCIs than the one for which the application was originally designed. Even considering an application that has been deployed on multiple DCI, its execution usually cannot be dispatched on those infrastructures with the same mechanisms. Hence, the execution framework should support multiple invocation means that allow users to identify which implementation and DCI may be used in each case. We present in the next section an end–to–end framework that facilitates end–users the *gridification* of applications and their executions on different DCIs.

3 End-to-end framework

The overall framework architecture is depicted in figure 1. The MOTEUR client is the front–end component that connects the user to the rest of the framework. It is also the working environment where users manipulate their applications at the design time. A MOTEUR client interacts with the MOTEUR server at runtime to execute the applications with a specific data set. The MOTEUR server is responsible to invoke each application deployed locally or remotely through generic interfaces. The final command–line (CLI) application, encapsulated by *jigsaw*, is submitted to a DCI by means of intermediate middlewares like DIRAC. During the execution, a user's credentials may be needed for authentication with the infrastructure, thus all of framework components can connect to a MyProxy server to fetch a proxy certificate. Finally, data transfers between executions are enabled through the VL-e Toolkit. The detailed description of each component is given below.

Jigsaw (formerly known as jGASW), a wrapper of CLI applications as services [1], provides a complete mechanism to characterize in details the execution of those applications. For each one, *jigsaw* enables the definition of multiple execution profiles corresponding to different DCIs. This execution description is then packaged along with the binary files in portable artifacts which are deployed and published as services. *Jigsaw* also enables the results mapping of the CLI applications after executing them on a computing element of the infrastructure.

MOTEUR, a scientific workflow environment [2], targets a coherent integration of a data–driven approach to achieve transparent parallelism and manipulate complex data structures. The MOTEUR client provides to users a graphical interface to configure services and describe the semantics of data flows. The description is represented by the GWENDIA workflow language [3], that supports the required expressiveness to represent services composition. While MOTEUR client offers design tools to build workflows and configure an environment of execution, MOTEUR server provides asynchronous invocation and orchestration of services, and improves the execution of large–scale, data–intensive workflows. The integration of *jigsaw* with MOTEUR provides a full range of functionality, making them suitable for the purpose of applications reuse and large–scale experimentation.

Addressing the credentials management task, the framework supports two alternatives to create and renew proxy certificates. The first one is to create a proxy directly from the user's certificate file and private key granted by any Certification Authority (CA). The proxy may be used in the MOTEUR client where the user's credentials are available. Since at runtime services are invoked remotely from the MOTEUR server or a Web services container, a second alternative is used to download a proxy from any MyProxyserver [4]. The user is just required to provide the login and password of the credential stored on the MyProxy server. The validity of the proxy is checked each time a connection is performed. An expired proxy will be automatically renewed without interrupting the entirely execution of the application.

Thanks to the integration of GASW [5], the framework can dispatch executions on multiple DCIs like PBS clusters or the European Grid Infrastructure (EGI). For the later, applications may be submitted directly to the WMS or through DIRAC [6]. The DIRAC environment provides an efficient submission mechanism implementing a pilot system. Concerning data transfers, the VL-e Toolkit [7] is integrated in the framework because it provides a unified view of heterogeneous file systems. It supports several protocols such as grid FTP or HTTP, and file schemes like the grid LFN or local FILEs. The MOTEUR client uses the operations implemented by the VL-e Toolkit to download services descriptions and upload artifacts required for service execution. It is also used for file staging on the services container to provide the data inputs to the application.

In summary, the integration of each tool is effective at several levels. At the front–end level, this integration provides an interface to *gridify* scientific applications, and to invoke the deployed services by means of standard Web services mechanisms or convenient bindings accessing the distributed infrastructure directly. At the back–end level, the integration gives a transparent access to multiple DCIs. It brings to the user the ability to execute the applications on those DCIs with the same execution and enactment engine.

4 Results and future work

Based on projects covering different aspects of DCIs, the resulting framework (MOTEUR & *jigsaw* et al.) brings added value through an end–to–end integration of technologies involved in the complete application lifecycle management. Nevertheless, to produce a comprehensive tool including third–party components like this framework represents a real challenge with respect to the development. The multiple sources of projects involves to ensure compatibility and proper integration, dealing with configuration constrains and management of component versions, among others.

The framework enables performing experiments in different DCIs or even local servers. Services created and deployed on the infrastructure are published and invoked using open standards. The high configurable support allows users to grant access to infrastructures based on the X.509 protocol transparently. Thus, the service–oriented approach for service invocation and composition, embraces flexible executions combining changeable infrastructures and handling data from heterogeneous sources. The effortless deployment of servers, for replication of services, ensures the scalability of experiments beyond the load balancing strategies implemented internally by any infrastructure.

The resulting framework is aligned in the current trend of the scientific community projects like Taverna (`http://www.taverna.org.uk`) or LONI Pipeline (`http://pipeline.loni.ucla.edu`), with significant differences though. Compared to Taverna, both take into account the management of applications as services and enable the use of Web services invocations, however DCIs are not integrated out–of–the box in Taverna. On the other hand, the LONI Pipeline does not support the Web service invocations but the native integration to DCIs are taken in consideration as they are in the framework. In this way, the framework represents a good trade–off between open standards and DCIs integration.

The future work includes to add another layer to the framework in order to support workflow interoperability with other scientific workflows specifications. In parallel, the support to additional DCIs is also considered in the measure that such extension has a real impact in the current work of regular users of the framework.

Acknowledgment

This work is supported by the SHIWA project (`http://www.shiwa-workflow.eu`) under a grant from the European Commission's FP7 INFRASTRUCTURES-2010-2 call, agreement number 261585. The authors would like to thank the members of the CREATIS laboratory, the Lhcb Dirac, and the VL-e Toolkit teams for the technical assistance during the integration of their tools in the framework.

References

[1] Javier Rojas Balderrama, Johan Montagnat, and Diane Lingrand. jGASW: A Service-oriented Framework Supporting High Throughput Computing and Non-functional Concerns. In *The IEEE International Conference on Web Services*, ICWS 2010, Miami (FL), USA, July 2010.

[2] Tristan Glatard, Johan Montagnat, Diane Lingrand, and Xavier Pennec. Flexible and Efficient Workflow Deployment of Data–intensive Applications on Grids with MOTEUR. *International Journal of High Performance Computing Applications*, 22(3):347–360, 2008.

[3] Johan Montagnat, Benjamin Isnard, Tristan Glatard, Ketan Maheshwari, and Mireille Blay-Fornarino. A Data-driven Workflow Language for Grids Based on Array Programming Principles. In *Workshop on Workflows in Support of Large-Scale Science*, WORKS'09, Portland (OR), USA, November 2009.

[4] Daniel Kouril and Jim Basney. A Credential Renewal Service for Long-Running Jobs. In *The 6th IEEE/ACM International Workshop on Grid Computing*, Grid 2005, Seattle (WA), USA, November 2005.

[5] Rafael Ferreira da Silva, Sorina Camarasu-Pop, Baptiste Grenier, Vanessa Hamar, David Manset, Johan Montagnat, Jérôme Revillard, Javier Rojas Balderrama, Andrei Tsaregorodtsev, and Tristan Glatard. Multi-infrastructure Workflow Execution for Medical Simulation in the Virtual Imaging Platform. In *The Ninth Healthgrid Conference*, HealthGrid 2011, Bristol, UK, June 2011.

[6] Adrian Casajus, Ricardo Graciani, Stuart Paterson, Andrei Tsaregorodtsev, and the Lhcb Dirac Team. DIRAC Pilot Framework and The DIRAC Workload Management System. *Journal of Physics: Conference Series*, 219(1–6), 2010.

[7] Silvia D. Olabarriaga, Tristan Glatard, and Piter T. de Boer. A Virtual Laboratory for Medical Image Analysis. *IEEE Transactions on Information Technology in Biomedicine*, 14(4):979–985, 2010.