Dressing Apache Airavata Services With Automatically User-Generated Interfaces

Daniele D'Agostino, Luca Roverelli, Gabriele Zereik,
Emanuele Danovaro, Andrea Clematis, Antonella Galizia
Institute for Applied Mathematics and Information Technologies "E. Magenes",
National Research Council of Italy, Genoa, 16149, Italy,
{dagostino, roverelli, zereik, danovaro, clematis, galizia}@ge.imati.cnr.it

Abstract: *The development of community-specific user interfaces of a science gateway can be a challenging task for non-IT experts. This contribution proposes an original, easy-to-use solution to tackle this issue, based on the PortalTS Web portal. In particular, we present how an extended instance of PortalTS, named EasyGateway, can "dress" Apache Airavata to manage job submissions.*

1. Introduction

Most of the toolkits and software frameworks used for the creation of science gateways provide an API-based interface that requires to the Gateway communities "only'' the effort of the design of front-end solutions [1]. However, this task may be challenging for non-IT communities, and a wrong selection of the front-end technology, combined with frequent developers turnover, can represent a major issue for the gateway sustainability [2]. In the following we present how this problem can be effectively tackled by using an extended instance of PortalTS, named *EasyGateway,* to "dress" Apache Airavata [3], one of the most used framework for gateways implementation in the US.

Main related works are [4-7]. A complete analysis will be provided in the full paper.

2. EasyGateway

PortalTS[1] is a Web portal developed in Typescript using the NodeJS and Express frameworks. It is composed by reusable modules, and implements a set of core modules for the standard functionalities available for a Web site (e.g. user and content management systems). PortalTS enables a fast development of custom modules through different features, such as a simple APIs for data persistence and users management.

Although PortalTS is a very young project (started in January 2016), it has been already used as the base for the development of the Science Gateway of the EU-funded FP7 EXTraS (Exploring the X-ray Transient and variable Sky[2]) project. This portal enables users to perform image analysis operations on the database collected by the European Photon Imaging Camera (EPIC) onboard the ESA's X-ray space observatory XMM-Newton. EXTraS Science Gateway currently supports the periodicity and the transient analyses [8]. A dedicated *micro-service* corresponds to each analysis and enables the execution of the specific code on target computing resources such as High Performance Computing facilities and the EGI Federated Cloud. The set of available analysis can be easily extended by adding adequate micro-services.

On the basis of this experience we decided to optimize and engineer the EXTraS portal in a new general-purpose Science Gateway toolkit, enchaining PortalTS (and using it as a base). We developed new modules which provide basic science gateways functionalities, like workflow configuration and submission, and the automatic generation of user friendly configuration interfaces for scientific models. The result of this process is called EasyGateway. In details, the new modules are the *Model Configuration module*, the *Workflow Configuration module* and the *Submission Handler module*, shown in Fig. 1.

The *Model Configuration module* is an innovative element, conceived for the automatic design and support of suitable interfaces to collect

[1] https://portalts.it

[2] http://www.extras-fp7.eu

Presented at Gateways 2016, San Diego Supercomputer Center, La Jolla, CA, November 2-3, 2016.
https://gateways2016.figshare.com/

the configuration parameters of any software tool provided within the gateway. In particular, this component allows to automatically generate a form based model configuration interface through a user friendly graphical interface, the Json-GUI builder[3]. Moreover, it is possible to easily define and manage the model metadata, like names or types, as well as the configuration parameters with associated metadata, like the field type (select, float, integer, date/time, etc.), the parameter default value, and other specific properties (e.g. the possible options for a select type). These features have been implemented using Json-GUI[4], a front-end library developed as a set of reusable AngularJS directives, that allows the dynamic generation of full-featured form-based web interfaces including validation and constraints. Once the model interface has been defined, the model is automatically added to the set of available models in the Workflow configuration module, and it is ready to be instantiated and configured in a workflow.

Json-GUI is especially effective when developers have to deploy software modules subject to frequent updates. Indeed, each user interface is dynamically built starting from the JSON file created by the Json-GUI builder, without the need to write any line of code. It is clear that the ability to modify the interface by simply changing a configuration object allows a faster development cycle for new gateway releases.

The *Workflow configuration module* provides users with the possibility to design, configure, submit and share experiments in terms of workflows, composed of software tools provided within the science gateway. Furthermore, it allows a user to retrieve a previously saved or shared workflow, and to edit it, before the execution.

The *Submission Handler module* manages submission of workflow created by the user on a specific set of resources and provides a full view of the status of all components of the submitted workflow, results and logs. In order to improve reusability and maintainability the Submission Handler has been conceived as an independent component, to be customized for each middleware layer or job submission system that the gateway exploits for the workflow execution.

Though EasyGateway is currently under development, it has been adopted for the refactored version of the DRIHM (Distributed Research Infrastructure for Hydro-Meteorology[5]) portal [9]. In this case we have to support the design and the execution of chains of Meteorological and Hydrological models, possibly on heterogeneous computing resources. Therefore, the focus is on the user experience during the chain configuration and on the composability of models avoiding consistency errors.

3. The dress

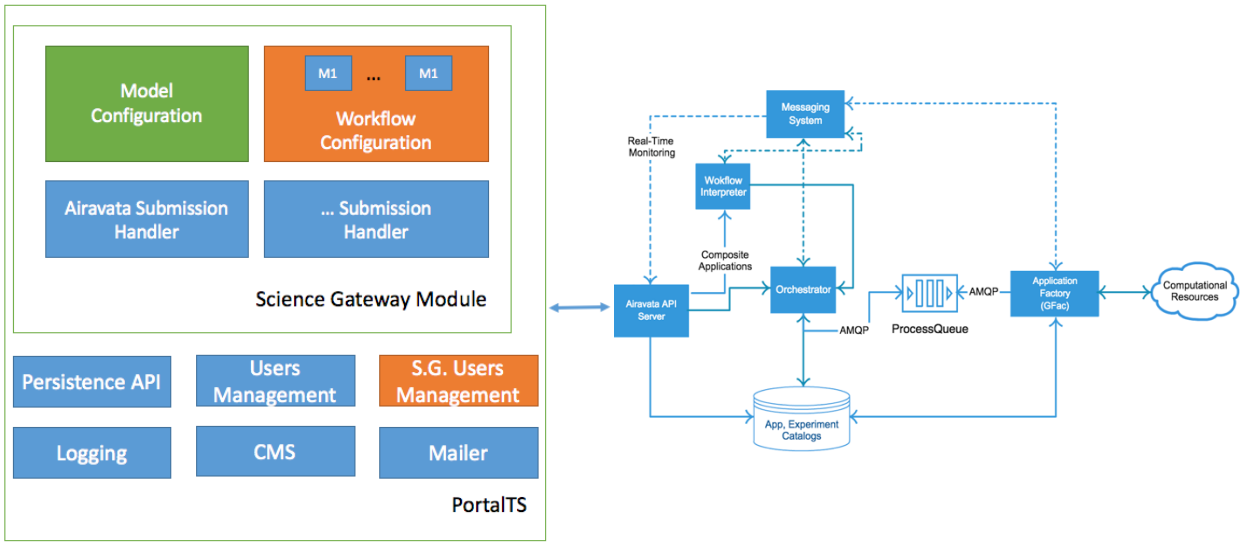
Apache Airavata is a powerful middleware, supporting the development of solid and feature-rich science gateways, thanks to its support to long running applications and workflows on distributed computational resources. The Airavata data model describes several entities: Project, User, Group, Gateway and Experiment. We mainly focused on Experiment data model that nicely matches EasyGateway workflow, jobs and job descriptions. The integration of EasyGateway with Apache Airavata leads to a rich user interface, with the support for the submission on a large set of middleware and queue managers.

We designed the customization of the Submission handler, named *Airavata Submission Handler*, able to exploit Apache Thrift-based API exposed by Apache Airavata API Server. We use the Java implementation of the API, wrapping the submission and monitoring of a job in a command-line Java application. The Java application allows us to test the Apache Airavata API independently from EasyGateway, and the application is easily integrated in the Submission Handler developed in Typescript. A more effective implementation could be the direct integration of the generated Typescript (or Javascript) API stubs. However, at the moment, the Apache Airavata repository contains scripts for the generation of the API only in Java, Phyton and CPP.

A complete description on the usage of EasyGateway with Apache Airavata will be provided in the full paper. Here we focus on how

[3] https://github.com/portalTS/Json-gui-builder
[4] https://github.com/portalTS/Json-gui
[5] http://www.drihm.eu

Fig. 1 The EasyGateway architecture and its interaction with Apache Airavata.



EasyGateway enriches the Apache Airavata framework by offering the user a graphical environment to enhance and simplify the configuration process and the input generation for job submissions.

At first, the EasyGateway basic users and roles management functionality, along with a rich administration interface, automatically provides the Science Gateway with authentication and, as a consequence, authorization handling. Thus, it is straightforward for the administrator to grant accessibility rights to single portal features, like permitting only to specialized users to define application interfaces. Moreover, when a specialized user defines an application interface, she/he has the possibility to limit its accessibility: in this way, the portal filters the users also at the single application granularity, deciding which groups (of users) will be allowed to execute each application.

Notably, the portal dresses Apache Airavata with the following valuable features. First, the possibility for users to easily share their experiment configurations and results with different groups or make them public. Subsequently, users can analyze the results of shared experiments, without the need to execute again an application with the same configuration or to use external tools; moreover, they can derive their own configuration, varying some parameter values.

The second added value is represented by the automatic definition of enhanced application interfaces. While Apache Airavata has basic features for providing application inputs, exploiting EasyGateway, a specialized user can design an interface where the input parameters: a) can be validated before the experiment submission; b) can be processed, to produce complex configuration files at runtime. Beside the basic parameter types, there is the possibility to define some particular types: *domains*, *datetime* and *fileupload*. The first allows to define geo-referenced points and rectangular regions over a geographical map; the datetime permits the definition of date and time parameters, while the fileupload gives the user the possibility to upload one or more files. For each parameter, the module allows to define different constraints with different error messages, where each constraint can be constituted by one or more conditions. Moreover, within each condition, the user can compare the value of the parameter with the value of another one, and also with a static value. It is also possible to define a help text, in order to show to the user a hint for the input completion.

To provide a clear picture about parameter specification and validation, let us consider an example: Isabella, a scientist, defines an application interface with start time, end time and two rectangular regions. Firstly, for each parameter she can define the general behavior: a) whether the parameter is mandatory or not; b) the possible default value; c) whether the parameter value can be edited or not. In addition, for the datetime parameter type, she can also specify whether the parameter is composed by the date, the time or both. Then, she is

able to define a constraint between the two datetime parameters: the start time parameter should be always earlier than the end time parameter. More interesting is the possibility to check the relation among geographical domains (i.e. nested, intersecting or disconnected). During the workflow configuration, if a user tries to set a configuration that violates the constraint, then the application interface will raise the previously specified error.

Finally, it is worthwhile to mention how the input is generated for the Job Submission. The Submission Handler Module retrieves the generated configuration files needed for the execution of the experiment from the Workflow Configuration Module. Depending on a user choice, the configuration files can have two different formats: they can be a standard key-value couples, or each value can be elaborated to accomplish a different representation. For example, the value of a datetime parameter must be formatted in a specific standard. Also the geographical domain may require a projection into a different coordinate system (e.g. from the WGS-84 Mercator projection used by Google Maps into Rotated Lat-Long projection). Currently, the translation into a specific standard is only partially supported, but it is allowed to implement custom routine that performs the translation in the specific standard.

Once the experiment is launched, the Submission Handler Module communicates the configuration files to Apache Airavata, that will pass it as input for the experiment.

4. Conclusion

In this contribution we presented the main features of EasyGateway, a front-end toolkit for Science Gateways development.

Future contribution will be dedicated to provide a full support for the Apache Airavata workflows and an enhanced accounting and resource management.

5. References

[1] M. Pierce, M. Suresh; M.A. Miller, A. Majumdar, B. Demeler, Science Gateway Operational Sustainability: Adopting a Platform-as-a Service Approach http://dx.doi.org/10.6084/m9.figshare.790760

[2] Lawrence, K. A., Zentner, M., Wilkins-Diehr, N., Wernert, J. A., Pierce, M., Marru, S., and Michael, S. (2015) Science gateways today and tomorrow: positive perspectives of nearly 5000 members of the research community. Concurrency Computat.: Pract. Exper., 27: 4252–4268. doi: 10.1002/cpe.3526

[3] Pierce, M. E., Marru, S., Gunathilake, L., Wijeratne, D. K., Singh, R., Wimalasena, C., Ratnayaka, S., and Pamidighantam, S. (2015) Apache Airavata: design and directions of a science gateway framework. Concurrency Computat.: Pract. Exper., 27: 4282–4291. doi: 10.1002/cpe.3534

[4] V. Balasubramanee, C. Wimalasena, R. Singh and M. Pierce, "Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development," 2013 IEEE International Conference on Cluster Computing (CLUSTER), 2013, pp. 1-1.

[5] van Hemert, J., Koetsier, J., Torterolo, L., Porro, I., Melato, M. and Barbera, R. (2011), Generating web-based user interfaces for computational science. Concurrency Computat.: Pract. Exper., 23: 256–268. doi:10.1002/cpe.1664

[6] M. McLennan and R. Kennell. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering". Computing in Science & Engineering, 12(2):48-52, 2010

[7] Brookes, E. H., Anjum, N., Curtis, J. E., Marru, S., Singh, R., and Pierce, M. (2015) The GenApp framework integrated with Airavata for managed compute resource submissions. Concurrency Computat.: Pract. Exper., 27: 4292–4303. Doi: 10.1002/cpe.3519.

[8] D'Agostino D, Roverelli L, Zereik G, De Luca A, Salvaterra R, Belfiore A, Lisini G, Novara G, Tiengo A. (2016) A microservice-based portal for X-ray transient and variable sources. PeerJ Preprints 4:e2519v1. Doi: 10.7287/peerj.preprints.2519v1

[9] D. D'Agostino, E. Danovaro, A. Clematis, L. Roverelli, G. Zereik, and A. Galizia, From Lesson Learned to the Refactoring of the DRIHM Science Gateway for Hydro-meteorological Research. J Grid Computing (2016). Doi:10.1007/s10723-016-9377-8