OpenCF-R: R IN THE CLOUD[*]

J. C. Castillo, F. Almeida, V. Blanco and A. Santos
Dpto. Estadistica, I. O. y Computacion, University of La Laguna, Tenerife, Spain

Keywords: Cloud Application Portability, Parallel Computing, Web Services, Dynamic Service Generation, R.

Abstract: One of the main goals of cloud computing-oriented environments is to offer access to distributed resources through interfaces and technologies based on web services. OpenCF is a portable framework that shares these goals and can be used as a development platform that offers hardware and software as a service. We include in this paper how to adapt an OpenCF portal to add and execute statistical package R routines. Thus, we contemplate the option that user can launch or incorporate R scripts as a new service dynamically to the portal, making remote computing easier for inexpert users. Automatic management and dynamic aggregation of R services are showed as new concepts in this context.

1 INTRODUCTION

Nowadays, cloud computing environments present the following features (Mark Baker and Laforenza, 2002): multiple administration domains, heterogeneity, scalability and dynamicity or adaptability. Large-scale systems add the difficulty of manage large amounts of resources. Web services standards provide an increased level of usability, extensibility and interoperability between pairs of services. The adoption of these technologies in the context of Grid and Cloud Computing has improved the efficient use of computing resources. Projects such as Globus (Foster, 2006), OpenCF (Santos et al., 2007), OpenNebula, or Nimbus (Sempolinski and Thain, 2010) have been generated based on web services technologies to manage computing resources, monitoring or management systems and scheduling, among others.

Open Computational Framework (OpenCF) provides access to resources in high performance computing for inexperienced users. The main goal is to reduce technology and knowledge walls to that users face when trying to access High Performance Computing Systems (HPCS). Web services technologies has been widely adopted in the OpenCF implementation. Performance monitoring systems or description of computational resources are offered to users as web services. Proposes to decouple the different services developed through a task scheduler, since these services can be used by third-party client applications. Its composition leads to a distributed meta-scheduler based on a web services platform that provides a wide range of applications.

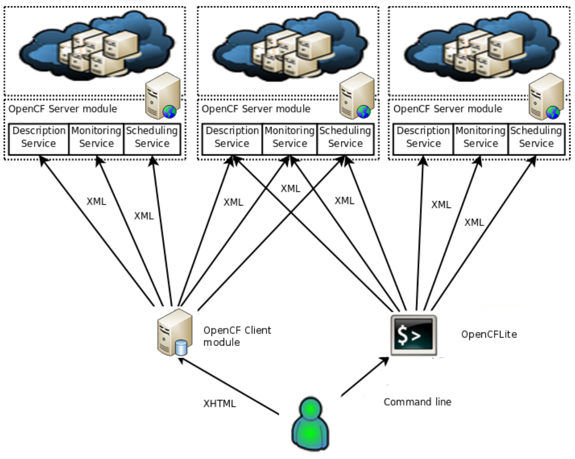

Figure 1: OpenCF-R providing services for HPC system description.

One of the main problems facing users of HPC is the portability of applications or scripts. The compilation of source code or execution of binaries is strongly linked to the platform where user wants to operate, limiting the ability of a plurality of existing infrastructure. Interpreted languages (like Perl, Matlab, IDL, Mathematica or Python) are a fairly robust alternative to circumvent this problem, because performance in

[*]This work has been partially supported by the EC F(EDER) and the Spanish MICINN (Spanish MEC through I+D+I contract, TIN2008-06570-C04-03, and TIN2011-24598).

C. Castillo J., Almeida F., Blanco V. and Santos A..
OpenCF-R: R IN THE CLOUD.
DOI: 10.5220/0003901504680471
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 468-471
ISBN: 978-989-8565-05-1

these languages has been gradually approaching compiled languages (Newhall and Miller, 1998).

In this paper we contribute with new facilities to the OpenCF architecture to support task performance of interpreted languages. The proposed solution, OpenCF-R, enables automatic generation of services as well as incorporation of new services dynamically. The user will be able to incorporate new computing services that would be available to other researchers. We have used the R language (Eddelbuettel, 2011) as a proof of concept: all main routines in R has been exported as new platform of services, in OpenCF-R the user is able to launch R scripts as a service to be executed in the HPC platform and, the end user is enabled also to upload R scripts to be added as a new web service in OpenCF-R dynamically. The project Ricardo (Das et al., 2010) follows a similar objective, however, OpenCF-R is a general purpose tool, allowing the use of scripts for users avoiding the use of low level jaql sentences to invoke R. Although Ricardo allows to manage large scale data base the know-how required for the user is higher.

The paper is structured as follows: we will talk briefly about interpreted languages in 2; section 3 summarizes OpenCF infrastructure; and support for these languages in OpenCF-R will be covered in section 4. Finally we conclude the paper with some considerations and comments on the project in 5.

2 INTERPRETED LANGUAGES

The increase in interpreted languages such as Python, Visual Basic, MATLAB, IDL, Maple and Mathematica, for algorithm development, prototyping, data analysis and graphical user interfaces (GUI) is an important trend in software engineering. However, using interpreted languages in HPCs is currently a challenge in academic and scientific enviroments.

From the standpoint of scientific, most of the solutions provided to problems dealt with interpreted languages are partial solutions, such as using an interpreted language to establish a calculation and then interact with a calculation core written in a compiled language (eg: C, C++, Fortran) (Kepner et al., 2000). However, this trend has been changing in recent years and we can find solutions to problems with high degree of computing such as SOLVCON (Chen, 2011), Python software environment for solving partial hyperbolic differential equations.

3 THE OpenCF ARCHITECTURE

OpenCF is a open source computational framework based on web services technologies. Facilitating access to distributed resources through web interfaces while simultaneously ensuring security is one of its main goals, as well as portability, generality, modularity and compatibility with a wide range of HPCs. In order to keep a modular design, two modules (see (Santos et al., 2007)) are given in the OpenCF package, namely, the server (side) module and the client (side) module. Modules can be independently extended or even replaced to provide new functions without altering the other system components.

The *client* is the interface between the end user and the system. We can find three submodules:

- The client `DataBase`, it has been implemented as a relational MySQL database and it is accessed through PHP scripts.
- The client `Query Processor` consists on a web interface through which users can interact with server's functionalities, such as listing available services, launching tasks or checking task's state.
- The client `Collector` manages the job's server-generated output.

The *server* manages all job-related issues, making them available at the service and controlling their state and execution. When Apache catches a new query from the client, it allocates a new independent execution thread to create a new instance of the server module. Server submodules:

- The `Query Processor` is a set of PHP scripts that distributing the job among the different components. Queries addressed for the computational system are dispatched to the Queue Manager Interface, and the rest of the queries are served by the `Query Processor`. The web service is also generated and served by this module.
- The `Queue Manager Interface` handles the interaction with the HPCs Queue System. The server needs to know how a job will be executed and how to query the status of a job being executed on the server supporting it.
- The `Scripts Generator` produces the scripts needed for the job execution under the various queue systems. Research is ongoing into making this module compatible with the Job Submission Description Language (JSDL) GridForum proposal as an alternative to the XML document currently used to describe a job.
- The `Launcher` is the interface between OpenCF and the operating system; it forks the process to

be executed, returns its identification and unlocks the thread handling the client query. The implementation is Perl-based to be independent from the architecture. In future versions of the OpenCF accounting system, this module will be responsible for collecting and reporting on individual and group use of system resources.

- The `Collector` is the client interface which delivers the output data produced by an executed job. Once a job has finished, the `Collector` automatically sends an e-mail to the user.

4 OpenCF-R: R IN THE CLOUD

Most development environments based on Web services offer facilities to include services that are accessible through a web interface. However, it is true that these environments do not provide general automatic mechanisms for adding new web services from user codes. Typically, export a service involves rewriting or adapt server and describing the service by a XML interface, meaning in practice to develop a laborious task for the developer. When the number of services is limited is a manageable task, however, when the number of services to manage are more than a certain size (hundreds or thousands of services), it must negotiate with automated tools to manage them.

One of the problems found during the design of OpenCF-R is the hard work needed to add new services to it. If the number of routines that we offer is limited for addition work is acceptable, only has to define the job description and verify that the code match the requirements of OpenCF-R. However, if you want to work with a greater number of jobs we have to create an automated mechanism.

The solution developed in this work is based on source code analysis to export as work. Generally, libraries analyze the source code and generates a XML description. This way of working fits seamlessly OpenCF-R as the description of the work used is also based on XML. At most it would be necessary XSLT transformation of the output generated by the library to be translated into the schema used by OpenCF-R.

The objective of adapting OpenCF to run interpreted languages scripts was only the beginning of a long way to go. Adding R a test platform was obtained for the dynamic generation of web services and other utilities (code execution uploaded by the user and offers the main functions of R as web services). The main problems encountered were:

- To identify the routines that would be exported as services. The list of services to be exported must be known in advance.
- To identify the types of data associated with the services. End users must enter the arguments of the service to run through the web interface of the client. A generic specification is needed to develop the service interface, and simple enough to be manageable in practice by a non expert user.
- It is very difficult to guess input and output arguments to a routine by analyzing only the header of the routine. We could develop a heuristic based on the use of `const`, pointers, etc.. but it should be adjusted to each type of code. Another solution is to write down the codes, for example, using the syntax given in the comments to describe each of the arguments (eg: based on javadoc).
- To generate the service interface, the help information for the user and the binary code to provide new services.

4.1 Main R Functions as a Services

Using a Python script and the library *rpy* (integrating python and R), we parsed th R doc and, for every function, was generated a XML file that describes it as a web service (i.e., specifying its input arguments, output, description and service name). A executable wrapper was written in perl to run a call to selected function of R with the arguments passed by the web service. This web service is added as some of the functions of the R base package and the package utils. Not all functions of R can be offered as a web service (eg, arithmetic operators, functions with no generic argument types, etc). To avoid incorrect services, a list of unsupported features as web services is given to avoid the incorrect XML file generation.

4.2 R Script Launching in OpenCF

A key objective of the project is that the user can launch their own scritps to machine computation. Running R scripts provided by the user in isolation was achieved as a web service. To do this, we have created a XML file describing the service in question, which requires the name of the main script file to execute and a R or zip file with the scripts necessary for the execution. Arguments of script are read from a R workspace file sent as input argument. Running the R script provided could generate several file results so, as a result of the service, the user downloads a zip containing all the files of the service.

4.3 Dynamic Web Services Generation

The objective of the dynamic web services generation in OpenCF is to provide to the servers more offering

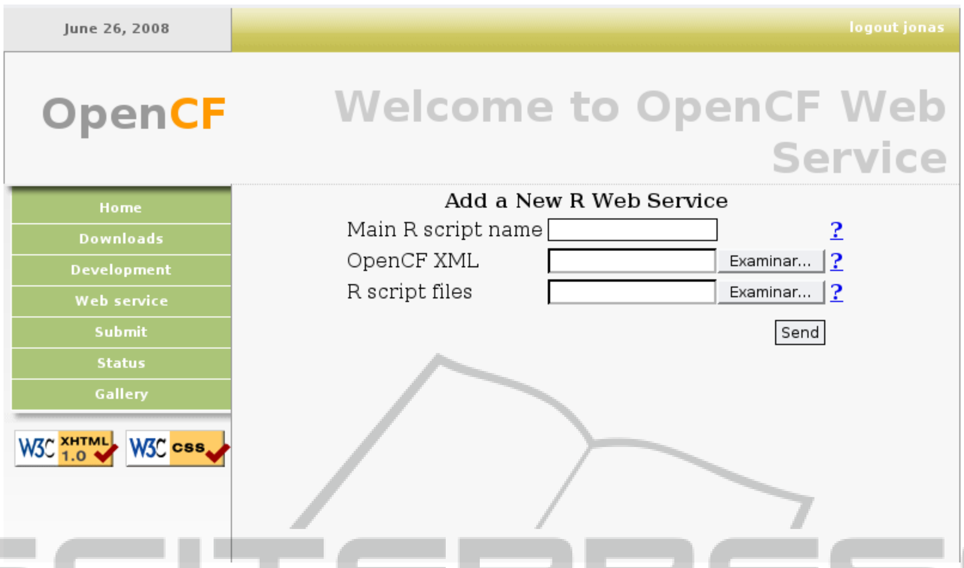

Figure 2: To add a new service dynamically.

services in a semi-automatically way. This service allows an user to define a new web service that runs a R script uploaded to the server for it. To add a new service, you need the script (or scripts) in R and a XML file describing the work, as shown in figure 2.

With this we obtain a new web service present on the server that launched the execution ready to be called by clients of OpenCF.

5 CONCLUSIONS

Web services-based technologies have emerged as a technological alternative for computational web portals. Facilitating access to distributed resources through web interfaces while simultaneously ensuring security is a major goal in most of the existing tools and development environments. OpenCF-R, the open source computational development environment that we have developed, shares these objectives and adds others, such as portability, generality, modularity and compatibility with a wide range of systems of High Performance Computing.

With the incursion of the interpreted languages we have added an extra value to OpenCF-R, expanding the chances for end users and making the platform even more independent and flexible as regards the management of tasks by system managers and the execution of them by users. With the ability to add web services dynamically achieves a greater degree of freedom and participation by users, further streamlining the process to run a task into the HPCS.

REFERENCES

Chen, Y.-Y. (2011). Software framework for solving hyperbolic partial differential equations.

Das, S., Sismanis, Y., Beyer, K. S., and McPherson, J. (2010). Ricardo: integrating r and hadoop. In *SIGMOD'10*, Indianapolis, Indiana, USA.

Eddelbuettel, D. (2011). High-performance and parallel computing with r.

Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems. *IFIP International Conference on Network and Parallel Computing*, 3779:2–13.

Kepner, J., Gokhale, M., Minnich, R., Marks, A., and DeGood, J. (2000). Interfacing interpreted and compiled languages to support applications on a massively parallel network of workstations (mp-now). *Cluster Computing*, 3:35–44.

Mark Baker, R. B. and Laforenza, D. (2002). Grids and grid technologies for wide-area distributed computing. *Softw., Pract. Exper.*, 32(15):1437–1466.

Newhall, T. and Miller, B. P. (1998). Performance measurement of interpreted programs. *Lecture Notes in Computer Science*, 1470:146–156.

Santos, A., Almeida, F., and Blanco, V. (2007). Lightweight web services for high performace computing. *European Conference on Software Architecture ECSA2007*, 4758.

Sempolinski, P. and Thain, D. (2010). A comparison and critique of eucalyptus, opennebula and nimbus. *Critique*, November:417–426.