*Proceedings of the 10th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2010
27–30 June 2010.*

Introducing High Performance Software Tools in Cloud Computing with PyACTS-PyOpenCF

F. Almeida[1], V. Blanco[1], V. Galiano[2], H. Migallón[2] and A. Santos[1]

[1] *Dpto. Estadstica, I.O. y Computacin, Universidad de La Laguna, S/C de Tenerife, Spain*

[2] *Dpto. Fsica y Arq. Computadores, Universidad Miguel Hernndez, Elche, Spain*

emails: `falmeida@ull.es`, `vblanco@ull.es`, `vgaliano@umh.es`, `hmigallon@umh.es`, `asmarre@ull.es`

Abstract

Many computational applications rely heavily on numerical linear algebra operations. Part of these applications are data and computation intensive that need to run in high performance computing environments. On the other hand, Cloud Computing is emerging as a new computing paradigm which aims to provide reliable, customized and QoS guaranteed dynamic computing environments for end users. For research groups, while the ACTS Collection brings robust and high-end software tools to the hands of application developers, cloud Computing provides convenient access to reliable, high-performance clusters and storage without the need to purchase and maintain sophisticated hardware. In this paper we propose to join these two paradigms of scientific computing in a framework that allows executing high performance tools included in ACTS in a heterogeneous and dynamic system.

Key words: Cloud Computing, high performance, Software Tools, ACTS, Python, Web Services, instructions

1 Introduction

Scientists and engineers in virtually every field are turning to high performance parallel computers to simulate and solve some of their problems. One reason for this trend resides in the fact that the models, algorithms, and phenomena are becoming more complex. Unfortunately, parallel architectures are expensive and hard to configure and administrate. Only major research centers have the necessary financial and human resources to manage a center of High Performance Computing.

In last two years, a new concept is emerging: Cloud Computing [2]. Clouds are hinting at a future in which we will not compute on local computers, but on centralized facilities operated by third-party computing and storage utilities. We can relate it to other similar technologies, especially grid-computing, but there are significant differences show in Table 1. In resume, Cloud computing describes a new supplement, consumption and delivery model for IT services based on Internet, and it typically involves the provision of dynamically scalable and often virtualized resources as a serviceover the Internet.

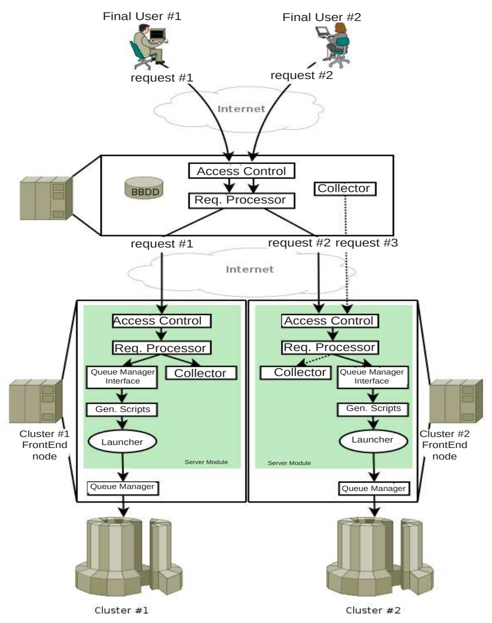
	Grid	Cloud
Underlying Concept	Utility Computing	Utility Computing
Main Benefit	Solve computationally complex problems	Provide a scalable standard environment for network-centric application development, testing and deployment
Resource distribution /allocation	Negotiate and manage resource sharing schedulers	Simple user Provider model pay-per-use
Domains	Multiple domains	Single domain
Character/history	Non-commercial, publicly funded	Commercial

Figure 1: Main differences between Cloud and Grid Computing

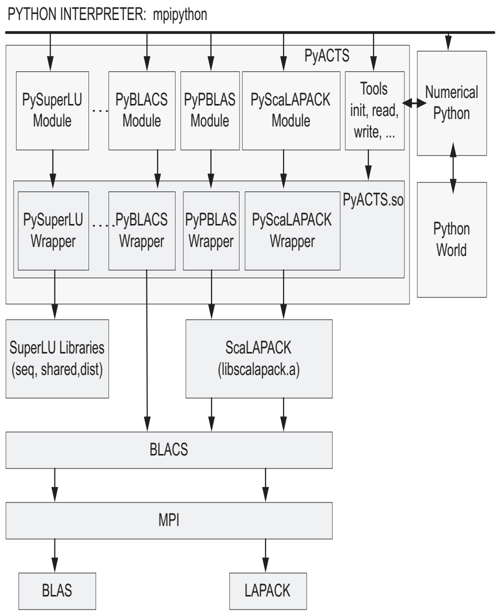
In previous work, OpenCF [8, 11] has been presented as a framework on a Cloud Computing infrastructure, where users can access to the computing facilities on demand according to their needs. A significant difference between OpenCF and others Cloud Computing projects is that OpenCF does not revolve around the creation of virtual machines as a way to offer services to users. Rather, it allows for direct instances of applications to be run on the computing servers(SaaSapproach).This, which at first glance might seem like a disadvantage, is proposed as a way to facilitate access to this type of tool to end users with limited knowledge of programming or high performance computing.

This paper proposes the use of scripts such as computing service to users of cloud computing. That is, in addition to providing access to applications compiled on computing servers, the user can program interpreted Python code programming its own high performance application to be executed on any platform in the cloud. Users can make use of precompiled libraries in high performance computing servers through distributions like PyACTS [5]. Thus, the code can make use of high performance libraries to the lowest level and to be independent of the platform on which to run.

This paper is arranged as follows. Section 2 and 3 introduces OpenCF and PyACTS respectively. In section 4, we present the framework that integrates both architectures and we show several examples that illustrates the advantages of these new paradigm.



(a) OpenCF Architecture



(b) PyACTS Architecture

Figure 2: OpenCF and PyACTS's models

2 OpenCF

This section describes the architecture of OpenCF. OpenCF software architecture is shown in Figure 2(a). As introduced in [9], OPenCF is highlighted by a modular design: module server and client module. The modules can be replaced independently and even extended to provide new functionality without disturbing the rest of the system components. The client and server implement the three lower layers of the stack that describes the Web service: Description of Services, XML Messaging and Transport. The fourth level, Service Discovery has not been implemented for security reasons. Therefore, system administrators still control access by of customers to parallel platforms through traditional techniques authentication.

The client provides an interface for the end user and translates the requests queries to the server. The server receives requests from clients Authenticated and transforms them into jobs for the queue manager. These modules, in turn, are also modularized. The module `Control Access`, `Submission Process` and `Collector` can be found on both the server and client. The client also maintains a database to manage information generated by the system. The server includes elements for generating scripts and work release the queuing system.Briefly, we should describe the features of the modules listed.

1. Client Module: The client is the interface between the end user and the system, where users are registered through a form. Below is a list of sub-modules.

- The `Database` stores information about users, servers, work, input and output files, etc.. It has been implemented as a base MySQL relational data [6] and is accessed through PHP scripts.
- The `Request processor` is via a Web interface which the user can access the list of available applications. Each entry in the list shows a short description of the routine. Tasks grouped according to the servers that support them. It also manages dynamically generate XHTML form input data according to the job description.
- The `Collector` manages the output generated by launched work on the server. We can also check the status of submissions through the Web interface, and download the results.

2. Server Module: The server handles all matters related to the work, making them available through the service and monitor its status and implementation.

    - The `Request Processor` consists of a set of PHP scripts that are responsible for analyzing the requests received from the client. In addition, it is also responsible to generate and export the Web service, to maintain updated the WSDL document [13] (*Web Service Description Language*) encapsulated with the protocol messages SOAP generated by NuSOAP [12].
    - The `Queue Manager Interface` manages the queue of the HPC system. The server needs to know how to run a work and how to check its status on the server is installed. Additionally, we need an XML description of each of the routines available to specify the job. In section 4 we will show an integrated example with the PyACTS library. In current version of OpenCF, once the user sends a job request, the server executes the code binary associated with the supplied arguments. In this way, we can incorporate new services by adding the file with the XML description with compiled code in the server module. This is the main difference with the proposed system in 4, where the application are programmed in a scripting language and not need to be compiled.
    - The `Script Generator` produces the necessary scripts for execution of work in different systems of queues. It is composed of a set of templates. We need a different template for each one of the queue managers supported.
    - The `Launcher` is the interface between OpenCF and operating system. For security reasons, you need a non-privileged user created to run the OpenCF code.

3. The `collector` is the interface that delivers the output data produced by execution of a job. Once work is complete, the queuing system automatically sends an email to the user, and moves output files to a temporary directory until they are downloaded by the `client collector`.

`PyOpenCF` born from the idea of developing OpenCF in a single programming language, so that it is more portable and independent as well as more efficient in upgrading the platform. For this, the language used was *Python* [7]. Python is an scripting language widely used by scientific and academic community. Same functions and characteristics of OpenCF have been implemented in this version. In this way, PyOpenCF offers a platform to submit precompiled jobs to the computing servers. However, if we would program our algorithm we could do it with an scripting language like Python, and submit this script with PyOpenCF. The main disadvantage is the bad performance in scripting languages, but according to [3], an application can be written in Python but the hard computational tasks can be executed in tunned libraries to each HPC system, without significant penalty in performance. In this sense, we will introduce PyACTS concepts in the next section.

3 PyACTS

The Advanced CompuTational Software (ACTS) Collection [4] is a set of software tools for computational sciences that helps programmers write high performance scientific codes for high-end computers. ACTS tools are mostly libraries (some are C libraries, some C++ class libraries, and some are Fortran libraries). They are primarily designed to run on distributed memory parallel computers. Portability and performance were both considerations in their design and implementation. The ACTS tools use the standard Message Passing Interface (MPI) [10] for communication. The computational model of ScaLAPACK, BLACS and PBLAS (included in the ACTS tools) consists of a one or two-dimensional process grid, where each process stores pieces of matrices and vectors. The prime beneficiaries of ACTS tools are developers of parallel engineering and scientific applications. Many areas of scientific computing are covered by ACTS tools, and can potentially make use of them. Nevertheless, parallel software can be more complex than serial software and significantly more expensive to implement.

In this context, we developed PyACTS as a set of modules which can be imported from the Python interpreter to enlarge the number of users that can make use of the routines included in ACTS. PyBLACS, PyPBLAS and PyScaLAPACK were our first steps to achieve our goal: an easy and integrated set of tools that can be used from Python but offers all the performance of the libraries in the original development environment (Fortran and C).

In Figure 3, we present a script used to test the interface to the PBLAS level 3 routine (*pvgemm*, $\alpha AB + \beta C,\ \alpha, \beta \in \mathbb{R},\ A, B, C \in \mathbb{R}^{n \times n}$). This example reads the data from three text files and stores them in *PyACTS Arrays*. Note that this reading is completed by a single process (usually, [0,0] in the process grid) and it sends the data to the rest of the processes using PyBLACS to obtain a two-dimensional block-cyclic distribution. After executing *Txt2PyACTS* in Figure 3, the variables a, b and c are *PyACTS Arrays* and can be used as parameters in PyACTS routines. Once the matrix multiplication is done, the routine *PyACTS2Text* collects the (distributed) results and writes them into the text file. It is interesting to compare this script of with a Fortran or

```
from PyACTS import *
import PyACTS.PyPBLAS as PyPBLAS
ACTS_lib=PyACTS.ScaLAPACK_ID      # ScaLAPACK ID
PyACTS.gridinit()                 # Grid initialization
alpha=Scal2PyACTS(1.2,ACTS_lib)   # Distribute scalars
beta=Scal2PyACTS(2,ACTS_lib)
a=Txt2PyACTS("data_a.txt",ACTS_lib) # Read Text files and
b=Txt2PyACTS("data_b.txt",ACTS_lib) # store in PyACTS Arrays
c=Txt2PyACTS("data_c.txt",ACTS_lib)
result=PyPBLAS.pvgemm(alpha,a,b,beta,c) # Call routine
PyACTS2Text("data_result.txt",result)   # Write results
PyACTS.gridexit()
```

Figure 3: Example of PyPBLAS: *pvgemm*

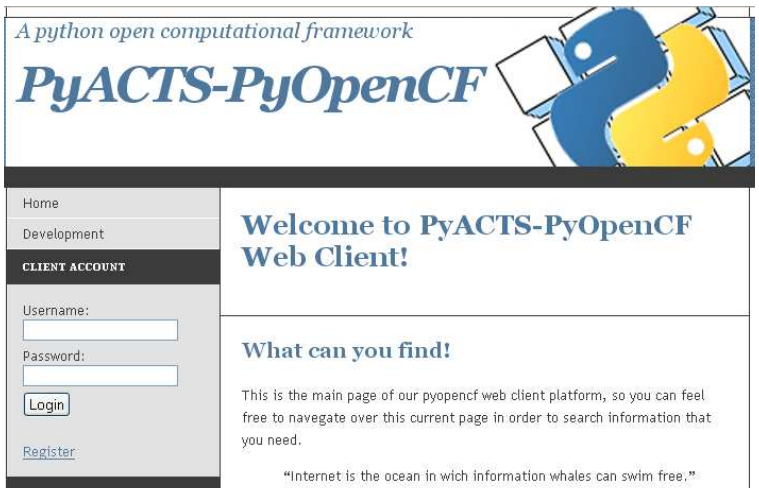
C implementation with same functionality. The implementation with python is usually more readable and easily, allowing faster development. Performance tests demonstrated that the Python interfaces do not involve a significant performance penalty. In sum, PyACTS is an intuitive, handy, and powerful tool to access ACTS tools from Python in a parallel setting.

4 A well-matched couple

We present an evolution of both tools (and PyACTS PyOpenCF) which is precisely the union and interaction to achieve high performance platform for cloud computing philosophy. For this purpose, a new service called *PyOpenCF&PyACTS Web Client* was added to the PyACTS' distribution web (http://pyacts.umh.es). Thus, a user can log into the portal and submit their papers through the web browser, without need for compilation or libraries linking. The same code can be executed by Python different computing platforms without being rewritten. The work environment OpenCF management control processes and their results in the various computer servers as explained in section 2. The innovation introduced by the union of both tools are the programming flexibility and power in performance we achieved by making use of ACTS library routines from the Python language. In Figure 4, the pyacts.umh.es web client is shown.

The features of the application that has been developed in this version are the following:

- List of servers: the list of registered computing servers in the system, including server name and the address on the same.
- List of scripts: shows the user a list of previously existing scripts or stored on a remote system.



Welcome to PyACTS-PyOpenCF Web Client!

What can you find!

This is the main page of our pyopencf web client platform, so you can feel free to navegate over this current page in order to search information that you need.

"Internet is the ocean in wich information whales can swim free."

Figure 4: PyACTS-PyOpenCF Web Client

- Creation of new scripts / applications: allows user to define their own algorithms and programs using the Python scripting language.
- Launch a job from a script: From a Python uploaded code , user can launch a job selecting a computing server.
- Job Status: You can view the status of a particular job. It shows both the ID of the job, as the status in the remote server.
- Download results: if the job produces results as a file, we could it at any time by downloading from the web service.
- Deleting scripts: You can remove applications you no longer need to use.

In short, it seeks to achieve a more comfortable computing services work without worrying about the implementation of where or how the computation is done. Computing as a service is achieved with this architecture.

5 Conclusions

In this work we have presented a new web service which provides a framework to execute user applications in high performance servers in a comfortable and simple way. Python example has demonstrated that PyACTS is a user friendly interface that hides the challenges of parallel programming from non professional users, and PyOpenCF has illustrated a integrated framework for managing jobs in a set of remote servers. Both architectures allows users writing and submitting their own codes in available computing servers without worrying about compiling, linking, queue management, etc. The proposed web client is available for scientific community at `pyacts.umh.es`.

Acknowledgements

This research was partially supported by the Spanish Ministry of Science and Education under grant number TIN2008-06570-C04-04.

References

[1] Blackford LS, Choi J, Cleary A, D'Azevedo E, Demmel JW, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D, Whaley RC *ScaLAPACK User's Guide.* SIAM, Philadelphia, Pennsylvania(1997)

[2] Foster IT, Zhao Y, Raicu I, Lu S, *Cloud computing and grid computing 360-degree compared*, CoRR abs/0901.0131 (2009).

[3] L. A. Drummond and V. Galiano and V. Migallón and J. Penadés *Py-ACTS: A High-Level Framework for Fast Development of High Performance Applications* Lectures Notes in Computer Science Vol.4395: 417–425, 2007

[4] L. A. Drummond ,O. A. Marques *The ACTS Collection. Robust and high-performance tools for scientific computing: Guidelines for tool inclusion and retirement.* Tech. Rep. LBNL/PUB-3175, Computational research division, Lawrence Berkeley National Laboratory, Berkeley

[5] L. A. Drummond, V. Galiano, V. Migallón José Penadés, *PyACTS: A Python Based Interface to ACTS Tools and Parallel Scientific Applications*, International Journal of Parallel Programming, ISSN 0885-7458. DOI 10.1007/s10766-008-0083-4 (2008).

[6] MySQL Database Manager, *http://www.mysql.org/*

[7] Python Programming Languaje, *http://www.python.org/*

[8] A. Santos, F. Almeida , V. Blanco and J. C. Castillo, *Web services based scheduling in OpenCF*, The Journal of Supercomputing, ISSN 0920-8542. DOI 10.1007/s11227-009-0352-z (2009).

[9] A. Santos, F. Almeida, V. Blanco, *Lightweight Web Services for High Performance Computing", European Conference on Software Architecture* Madrid. Spain.(2007)

[10] Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J *MPI: The complete reference.* The MIT Press, Cambridge, MA (1998)

[11] A. Santos, F. Almeida, V. Blanco, *The OpenCF: an Open Source Computational Framework based on Web Services technologies* Seventh International Conference on Parallel Processing and Applied Mathematics PPAM2007, Gdansk, Poland.(2007)

[12] NuSOAP. Librera SOAP PHP, *http://dietrich.ganx4.com/nusoap/*

[13] WSDL (WS Description Language), *http://www.w3.org/TR/wsdl/*