A Hybrid Grid/Cloud Distributed platform : a case study

Mohamed Ben Belgacem, Haithem Hafsi, and Nabil Abdennadher

University of Geneva
Mohamed.Benbelgacem@unige.ch
National School of Computer Science (ENSI), Tunisia
Haithem.Hafsi@gmail.com
University of Applied Sciences, Western Switzerland, hepia Geneva
Nabil.abdennadher@hesge.ch

Abstract. The scene of the computational sciences has considerably changed during the last years. Today, New emerging Desktop grid and Cloud e-infrastructure have a considerable potential to be adopted and used in large scale to exploit thousands of CPUs power to run both scientific and commercial applications. This paper targets scientists and programmers who need to accelerate their scientific research by running their applications on distributed Grid/Cloud infrastructures. We present a hybrid Grid/Cloud platform used to deploy a phylogeny application called MetaPIGA. The aim is to combine the advantages of Grid and Cloud architectures in order to set up a robust, reliable and open platform.

Keywords: distributed computation, Grid and Cloud computing, MetaPIGA

1 Introduction

The concept of grid Computing was born in the mid of 1990s as an answer to the increased demand of high performance computing that required more computing power than a single cluster could provide [1]. According to [2], Grid Computing has three characteristics:

- decentralized resource control,
- non-guaranteed qualities of services : latency, throughput, and reliability,
- standardization: Grid middleware is based upon open and common protocols.

Simultaneously with Grid Computing, a second alternative emerged. It consists of executing high performance applications on anonymous connected computers by using their available resources. This concept is called Volunteer Computing (VC). The most known systems are BOINC [3] and XtremWeb [4]. In the remainder of this paper, Grid will also include volunteer computing.

Despite the number of research projects carried out in the domain of Grid, these technologies were rarely commercialized. The development of Grid Computing and its standards was mainly driven by scientific communities.

For Cloud Computing, there is no established definition yet. According to [5], “a Cloud is a pool of virtualized computer resources”. The same paper considers Clouds to complement Grid environments by supporting resources management. Clouds allow the dynamic scale-in and scale-out of applications by the provisioning and de-provisioning of resources.

This paper proposes a “hybrid” platform composed of a volunteer computing infrastructure, called XtremWeb-CH (XWCH: www.xtremwebch.net), and a Cloud infrastructure, used as provisioning system. The platform is used to develop, deploy and execute a high performance phylogenetic application called MetaPIGA [6]. As stated by [7] and [5], Clouds are a “useful utility that you can plug into your Grid”. Our vision is to:

– combine the reliability of Cloud infrastructures and the “openness” of Grid environments,
– allow users deploying their applications on a reliable platform composed of a heterogeneous infrastructure: Grid, Cluster and Cloud.

This document is organized in 6 sections. After the introductory section 1, section 2 gives an overview of Grid vs. Cloud. Section 3 presents the Venus-C European project that aims at implementing a development environment for e-sciences applications on Cloud Infrastructure. The concepts proposed by Venus-C are used as guidelines in our research. Section 4 presents the hybrid solution developed in the framework of our research. Section 5 gives some experimental results carried out in order to evaluate the proposed solution. Finally, section 6 gives some perspectives of this research.

2 Grid vs. Cloud

This section compares Grid and Cloud [8] within 7 criteria detailed below:

1. *Resource localization*: while Grid Computing is defined by its geographically dispersed and decentralized resources, Cloud Computing seems to be a step back towards centralizing IT in data centers.
2. *Virtualization*: few research projects have integrated virtualization in grid projects. In Cloud, virtualization is one of the cornerstones; it allows the dynamic scale-in and scale-out of applications by the provisioning and de-provisioning of resources.
3. *Type of applications*: Contrarily to Grid, Clouds are not limited to e-sciences “batch” applications, but also support “interactive applications” such as Web and three-tier architectures.
4. *Development of applications*: the approach of how to develop applications is very different in Grids and Clouds. In Grids, the user typically needs to generate a binary for his application. This binary is then transferred to and executed on the remote resources in the Grid. Clouds allow a fundamentally different approach to software development. For instance, the Cloud provider offers “ready-to-use” components, the user can then dynamically assemble these existing functionalities to construct his Cloud-native application.

5. *Access & ease of use*: access to Grid resources is realized via a specific and often complex middleware. In contrast, interaction with resources in the Cloud is established via standard Web protocols, facilitating the access for the users. The lightweight accessibility and ease of use is one key factor that helped Cloud vendors succeed to convince non-academic customers to deploy their applications on their Cloud in a relative short period of time.
6. *Business model and SLAs*: as stated previously, business model, pricing and SLAs are one of the cornerstones of Cloud. These concepts are completely absent in Grid.
7. *Switching cost*: Through standardization, a Grid user can easily switch from the resources of one Grid provider to another. Due to the lack of standards, this is not possible in Cloud environment. Typically, Cloud providers have no interest in participating and implementing standards enabling potential customers to switch easily.

3 The Venus-C European project

3.1 Project Overview

Venus-C [9] is a European project funded with the purpose to provide a new friendly-user Cloud solution for the scientific research domain in Europe. The target end-users are mainly individuals and researchers group that never have had access to high performance computing resources and are content with their desktop machines to run their applications. The objective of Venus-C project is to make it possible for researchers community to run easily their applications on a large Cloud computing infrastructure in order to accelerate their scientific researches. Several scientific applications from several domains have been ported on the Venus-C platform.

Technically, the Venus-C is an interface between the Cloud providers and the end-users. It aims to provide a Platform as a Service (PaaS) with a set of tools and APIs to easily develop e-sciences applications and execute jobs that requires an execution coordination and a platform elasticity features.

One of the potential Cloud resources providers of theVenus-C project is the Microsoft Windows Azure infrastructure, which is based on Windows operating system. In what follows, we will interest on one of the programming model in Venus-C project: the “Generic Worker”.

3.2 Generic Worker concept

The Venus-C project comes up with the Generic Worker (GW) concept, an intermediate layer between the Azure platform and the end-users that shields them from technical complexity of the steps to use cloud computing resources. The main role of the GW is to facilitate the creation of the VM instances on the Cloud infrastructure and the application execution. Figure 1 depicts the GW component and its features. In its simple form, the GW is composed of a .Net based package and an API used to start VM instances on the Azure platform. Several types of the VM are supported: small, medium large and extra-large. The type of a VM is mainly determined by the number of cpus and the memory size. To start VM instances through the Azure web portal, the user should upload the GW package with his XML configuration that mainly determines:

– the number of VM instances
– data access and certificate credentials.

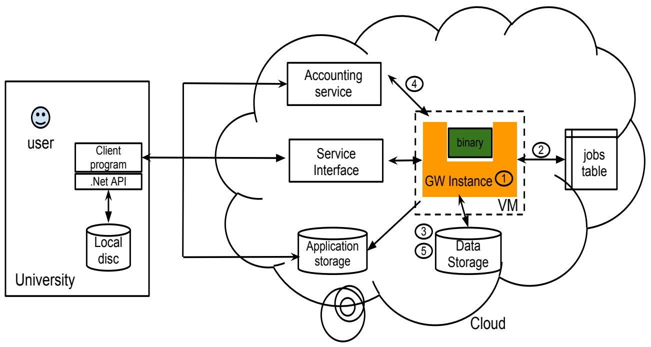


Fig. 1: Venus-C architecture overview

A Venus-C application can be composed of a workflow of jobs, where dependencies are based on input files: a job cannot be started unless its input files exist in the storage domain. Each running VM contains a GW instance that handles the execution of a given job (1). All the submitted jobs are stored in an Azure database table, and, then, scheduled by a service monitoring to the available GW instances. Periodically, each GW instance reads the database and retrieves its scheduled job (2). To execute the job, the GW instance should check the existence of its job's input file in the Cloud storage domain, then, loads them with the binary and any necessary libraries files to its local machine disc (3). Accordingly, the status of the job is tracked during its execution in the database (4). When the execution ends, the GW instance stages out the job result in the user storage domain (5). Besides, the GW provides a web service interface that allows the user through its client program to perform the scaling, notification and job management services. It worth noticing here that the number of the GW instances can be efficiently scaled on demand through the client program.

4 Hybrid Solution

The main idea behind the hybrid solution is to combine the XtremWeb-CH volunteer computing platform (XWCH: www.xtremwebch.net) with:

– Cloud infrastructures such as Amazon Elastic Cloud Compute (EC2) [10] and Azure,
– high performance oriented Cloud platforms such as Venus-C Generic Worker (GW) package.

The goal is to create a scalable and reliable large scale distributed platform used to deploy and execute the phylogenetic MetaPIGA application [6]. In what follows, we present, first, a brief description of the XWCH platform. Then, we describe how the hybrid solution is elaborated.

4.1 The XWCH platform

The XWCH platform consists of three components: a coordinator, workers and warehouses. The coordinator schedules jobs and pre-assigns them to the workers. An XWCH worker is a small Java daemon that runs on a user or institute machine. Periodically, a worker reports itself to the coordinator, asks for a job, retrieves job's input files and stages out computation results in the warehouses. Since the workers could be fire-walled and could not communicate with each other to retrieve files for their jobs, the warehouses are used as file repositories to ensure file communication between the jobs within the same workflow. If the coordinator does not receive signal from a worker which is executing a job, it simply removes it from the workers list and assign its job to another available worker. A flexible API allows users to submit and monitor jobs according to their needs. Several applications have been ported on the XWCH platform [11].

4.2 Hybrid platform

The "global" challenge behind bridging XWCH and Cloud is to scale up the XWCH infrastructure with Cloud resources. Let's remind here that XWCH workers are volunteer based, they belong to universities and/or individuals. Resources (CPU, cores number, memory, software tools) which are available on these volunteer workers are similar to those available on "off-the-shelf" computers.

The main idea can be simply described as follow: when resources requested by the MetaPIGA application are not available on the volunteer XWCH infrastructure, the system creates its "private" resources on the Cloud according to the needs (processor performance, main memory, etc.) of the application. These resources are created "on the fly" on the Cloud, used by MetaPIGA jobs and released as soon as the execution ends. In this paper, we consider two scenarios for resources scaling.

In the first scenario(figure 2), MetaPIGA jobs are submitted to the XWCH coordinator (1). When the requested resources are not available on the Volunteer infrastructure, the XWCH-coordinator creates a "private" XWCH worker supporting these resources (2). This private worker will then execute the job for which it was created. In this scenario, Cloud resources are considered as part of the XWCH infrastructure. The user uses only one developing environment :

XWCH API. This scenario was tested with two Cloud infrastructures: Amazon and Azure.

In the second scenario (figure 2), when XWCH infrastructure is unable to provide the necessary resources, the MetaPIGA application creates itself the requested resources and submits directly its jobs to the Cloud (1). In this case, Cloud resources are not considered as part of the XWCH platform. MetaPIGA jobs use the Cloud storage to retrieve their input files. After execution, the job's results are stored in the Cloud storage in order to be retrieved by the metaPIGA application. The developer uses two different APIs to submit his jobs: XWCH API and Cloud API. He also manages data flow between jobs running on the Cloud and those running on XWCH platform. This scenario was tested with Venus-C GW platform.

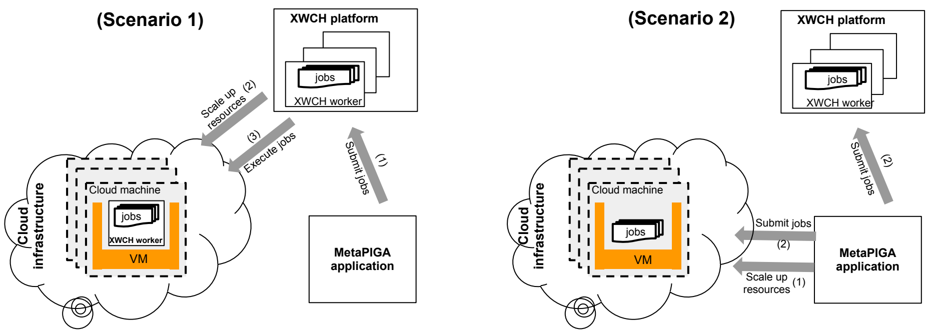


Fig. 2: Hybrid platform

5 Experiments

The two proposed scenarios are used to deploy and execute the MetaPIGA application, developed at the University of Geneva, over large hybrid computing infrastructure.

5.1 MetaPIGA application

The Java based MetaPIGA [6,12] application consists of a robust implementation of several stochastic heuristics for large phylogeny inference (under maximum likelihood), including a simulated annealing algorithm, a classical genetic algorithm, and the metapopulation genetic algorithm (metaGA) together with complex substitution models, discrete Gamma rate heterogeneity, and the possibility to partition data. Heuristics and substitution models are highly customizable through manual batch files and command line processing.

MetaPIGA is a CPU time consuming application. For instance, one big dataset needs in general 500 CPU hours. Assuming that 200 analyses are launched every year, the total number of CPU hours needed per year is equal to 100'000.

MetaPIGA is well suited for parallelization since several populations can be run in parallel and can therefore be sent to different machines.

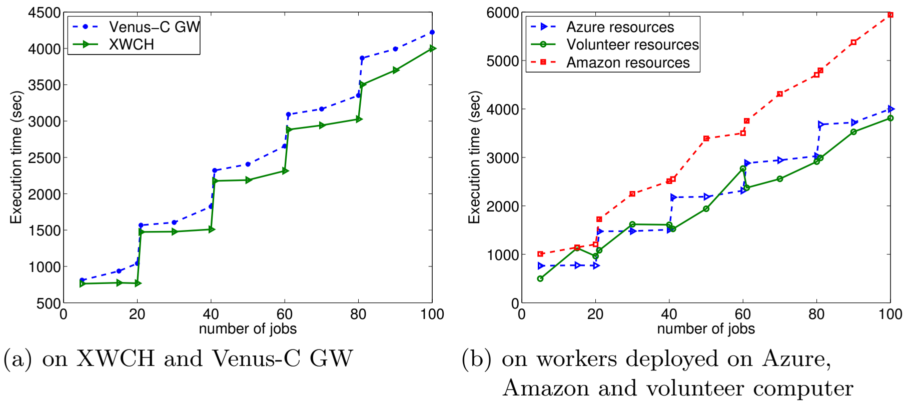
5.2 Measurements

Figure 3(a) compares the overhead generated by the two API : XWCH and Venus-C GW. Since the native Venus-C GW API is only implemented on Azure, we use this platform as a hardware infrastructure.

The results show that the overhead generated by XWCH API is slightly inferior to Venus-C GW API. In this figure, the number of available XWCH workers (resp. GW instances) is equal to 20.

Figure 3(b) compares the performances of MetaPIGA when executed on an XWCH platform using three "types" of workers:

– volunteer, non dedicated, workers,
– workers deployed on Azure,
– workers deployed on Amazon.



(a) on XWCH and Venus-C GW (b) on workers deployed on Azure, Amazon and volunteer computer

Fig. 3: Time execution of MetaPIGA

For both Amazon and Azure workers, we have used small VM instances. An Amazon VM instance runs Ubuntu operating system and has a 1 ECU ($EC2$ $Compute$ $Unit \simeq 1.0 - 1.2$ GHz) of CPU speed and 1.7 GB of memory. An Azure VM instance runs Windows operating system and has a 1.6 GHz of CPU speed and 1.75 GB of memory. Regarding the XWCH workers (Linux and Windows), they are installed on student machines having each an average CPU speed of 2.2 GHz and 3 GB of system memory.

Results show that the execution time of the MetaPIGA application on Azure workers is slightly higher comparing to the Amazon ones. Besides, the non-stairs shape obtained in the XWCH curve can be explained by the volatility aspect of the XWCH platform, i.e that the number of connected XWCH workers on the platform can vary during execution.

6 Conclusion

This paper presents two scenarios to bridge the XWCH Grid platform with Cloud infrastructure. Cloud is used as a provisioning system which allows users to "rent" resources not supported by the Grid. Two scenarios were proposed: In the first case Cloud resources are not seen by the user, they are managed by the Grid itself. Contrarily to this approach, the second scenario assumes that the user submits himself the jobs to the Cloud. It therefore obliges the developer to use two different developing environments.

The two scenarios have been tested in the case of MetaPIGA application with Amazon, Azure and Venus-C Cloud platforms. The next step will be to generalize this approach to other applications and develop a "generic" toolkit environment that supports other Cloud infrastructures.

References

1. Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.
2. Ian Foster. What is the Grid? - a three point checklist. *GRIDtoday*, 1(6), July 2002.
3. BOINC. `http://boinc.berkeley.edu/`.
4. XtremWeb. `http://www.xtremweb.net/`.
5. Boss G, Malladi P, Quan S, Legregni L, Hall H. IBM high performance on demand solutions. Technical report, IBM developerWorks, 2007.
6. MetaPIGA 2 - Large phylogeny estimation. `http://http://www.metapiga.org/`.
7. Wolfgang Gentzsch, DEISA. Grids are Dead! Or are they? `http://www.hpcinthecloud.com/hpccloud/2008-06-16/grids_are_dead_or_are_they.html`, 2008.
8. Weinhardt C., Blau B., Meinl T., Stößer J. Cloud Computing - A Classification, Business Models, and Research Directions. *Business & Information Systems Engineering journal*, 1:391 – 399, 2009.
9. VENUS-C European project. `http://www.venus-c.eu/`.
10. Amazon Elastic Cloud Compute. `http://aws.amazon.com/ec2/`.
11. Nabil Abdennhader, Mohamed Ben Belgacem, Raphaël Couturier, David Laiymani, Sébastien Miquée, Marko Niinimaki, and Marc Sauget. Gridification of a radiotherapy dose computation application with the xtremweb-ch environment. In *Proceedings of the 6th international conference on Advances in grid and pervasive computing*, GPC'11, pages 188–197, Berlin, Heidelberg, 2011. Springer-Verlag.
12. Mohamed Ben Belgacem, Nabil Abdennadher, Marko Niinimaki. *Desktop Grid Computing*, chapter The XtremWebCH Volunteer Computing Platform (3). Numerical Analysis and Scientific Computing Series. Chapman and Hall/CRC, June 25, 2012.