CodeCloud: A platform to enable execution of programming models on the Clouds

Miguel Caballer*, Carlos de Alfonso, Germán Moltó, Eloy Romero, Ignacio Blanquer, Andrés García

Instituto de Instrumentación para Imagen Molecular (I3M), Centro mixto CSIC – Universitat Politècnica de València – CIEMAT, Camino de Vera s/n, 46022 Valencia, Spain

**A R T I C L E   I N F O**

Article history:
Received 7 October 2013
Received in revised form 7 February 2014
Accepted 7 February 2014
Available online 17 February 2014

**Keywords:**
Cloud computing
Virtual infrastructures
Elasticity

**A B S T R A C T**

This paper presents a platform that supports the execution of scientific applications covering different programming models (such as Master/Slave, Parallel/MP, MapReduce and Workflows) on Cloud infrastructures. The platform includes (i) a high-level declarative language to express the requirements of the applications featuring software customization at runtime, (ii) an approach based on virtual containers to encapsulate the logic of the different programming models, (iii) an infrastructure manager to interact with different IaaS backends, (iv) a configuration software to dynamically configure the provisioned resources and (v) a catalog and repository of virtual machine images. By using this platform, an application developer can adapt, deploy and execute parallel applications agnostic to the Cloud backend.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Scientific computing has long been devoted to close the gap between scientists, who require executing resource-starved models, and computer scientists, who can deliver the power required to solve challenging computational problems. In the last decades, different computing infrastructures have been used to provide computing power to the scientific community. Moreover with the commoditization of hardware, clusters of PCs became a suitable platform for scientists to execute their workloads. However, scientific problems required computational resources far beyond the capacity of a single cluster of PCs. The increase in network bandwidth made utility computing possible. Organizations started sharing computational power and storage (among other resources) in the so-called Grids, where scientific users could execute large experiments. Grid computing has proved to be a valuable tool not only to foster collaboration among research but also to aggregate enough computing power to tackle challenging problems that could not have been solved before (Jacq et al., 2008).

However, the Grid also exposed some drawbacks for scientists, since they had to adapt their applications to fit the requirements of the underlying computing platforms (in terms of Operating System, software libraries, etc.). Although the Scientific Gateways made progresses toward abstracting the usage of the Grid (Wilkins-Diehr et al., 2008), the diversity of platforms and software configurations made the approach a compile-once run-anywhere difficult. This hindered the massive adoption of the Grid out of the academic environment, which typically remained composed by Virtual Organizations with pre-configured environments and tools.

With the advent of Cloud computing, the idea of utility computing is reconsidered. The use of virtualization and its ability to customize the underlying infrastructure to the requirements of the applications (not the other way round, as in Grid computing) opened new opportunities. Cloud computing enabled users to migrate clustered based applications to Cloud-computing resources without modifying the existing resources. The usage of virtualized infrastructures on top of Cloud infrastructures enables to dynamically deploy suitable computing platforms, such as a virtual cluster. In this way users are provided with larger computing capabilities, but with the software and environment they are familiar. Therefore scientists can run their unmodified codes on modern computer resources without investing time in porting their applications to new computer designs.

For that, this paper describes an architecture and the implemented platform (called CodeCloud) to perform the execution of scientific applications on Cloud computing infrastructures, supporting different programming models (currently Master/Slave, Parallel/MP, Workflow and MapReduce). The scientists just need to provide a high-level description of the jobs to be executed,
the programming model required, and the computing and execution environment requirements. Then, the system performs the automatic provision of the virtual infrastructure that satisfies the aforementioned requirements, executes and monitors the jobs, including data management. The platform features both horizontal and vertical elasticity capabilities to dynamically allocate and deallocate resources from both on-premise clouds (such as OpenNebula and OpenStack) and public cloud providers (including Amazon Web Services). This paves the way for scientists to easily access vast computing resources on-demand with minimal investment in application porting.

After Section 1, the remainder of the paper is structured as follows. First, Section 2 describes the related work in this area. Next, Section 3 provides a high-level overview of the CodeCloud architecture and its components. Then, Section 4 describes the CJDLL domain specific language that enables the user to define the jobs to be executed providing a common abstraction layer that insulates the user from knowing the internal details of the Cloud deployment. Later, Section 5 defines the architecture and details the software components developed, it also shows the elasticity capabilities of the platform. Section 6 describes a study case to demonstrate the functionality of the developed platform. Finally Section 7 summarizes the paper and points to future work.

2. Related works

The execution of scientific applications on the Cloud involves (i) the provision of Cloud computing resources, mainly computational nodes and storage, (ii) the deployment of the applications and their dependent services, and (iii) the dynamic adaptation of the resources provisioned to the variable requirements of the applications at runtime.

Most of these aspects are shared by applications that successfully migrated to the Cloud or those that were developed for the Cloud leaning their developments on some PaaS (Platform as a Service, Zhang et al., 2010). A PaaS solution creates an environment for developers to access Cloud resources from a high-level perspective, without dealing with the infrastructure details. The developer is provided with an API (or SDK) to compose the services offered by the platform but no direct interaction with the infrastructure (the Virtual Machines) is performed. Therefore, an abstraction for the execution of services and applications is provided.

2.1. Platform as a service solutions

There are some well-known commercial PaaS solutions such as Google App Engine (GAE), Microsoft Azure and Amazon Web Services (AWS). They provide support for their own commercial Cloud infrastructures and services allowing the development of applications in several languages like .NET, Java or Python.

Another commercial solution is PaaS Manjrasoft Aneka (Vecchiola et al., 2009), a software platform that provides a runtime environment and a set of APIs that allow developers to build .NET applications that leverage their computation on either public or private Clouds. Aneka provides special support to coordinated tasks (using a similar interface to the one used for programming threads) and Map/Reduce operations.

There also exist free PaaS platforms. For example, Heroku can deploy applications using a large set of languages and environments such as Ruby, Node.js, Clojure, Java, Python, and Scala. It also provides a large set of third party services like databases, caching, monitoring, performance management, etc. to enhance the applications. CloudFoundry supports the application development frameworks Spring, Ruby on Rails, Ruby and Sinatra, Node.js and Grails. It also provides a set of services to the application developers such as some relational database management systems (i.e., MySQL, PostgreSQL). Redis for a key-value NoSQL database and RabbitMQ as a messaging service. AppScale (Chohan et al., 2010) is an open source implementation of the GAE PaaS Cloud technology. As a new development over AppScale, Neptune (Bunch et al., 2011) is a domain specific language over Ruby that automates the configuration and deployment on multiple nodes of applications based on MPI or Hadoop MapReduce. Besides its simple user interface, it is noteworthy the support for launching on multiple IaaS at the same time and recycling unused nodes in order to reduce the final per-hour cost. However, like the rest, it does not enable to customize the VMs with user requirements.

ConPaaS (Pierre et al., 2011) is a runtime environment to run applications in the Cloud supporting OpenNebula and Amazon EC2 Cloud deployments. It currently includes a Web hosting service supporting PHP and Servlets, a MySQL database service, a batch processing service and a MapReduce service. In ConPaaS, an application is defined as a composition of one or more services. The availability of a custom budget-constrained scheduler for the batch processing service (Oprescu and Kielmann, 2010) is specially interesting. However, there are still details that remain unsolved. Scientific users do not have to deal with the set-up of their local environments. Clouds never require users to do so.

In Table 1 the comparative between the mentioned platforms is extended considering aspects such as

1) the software license type;
2) the software interface of the product;
3) the capability of provisioning from public IaaS;
4) whether a custom VM can be used to deploy the VMs and in that case if a configuration is required previously;
5) if an automatic management of applications is supported;
6) if rules can be defined to dynamically resize the provisioned resources, and
7) if it is done considering heuristics to minimize the final cost and
8) the fulfillment of Quality of Service (QoS) and Service Level Agreement (SLA) specifications;
9) the programming models supported;
10–15) if some abstraction is offered to identify and move files using different protocols; and
16) if there is some support to control the use of the resources by applications and
17) users.

2.2. Related tools for Cloud computing

In addition to the aforementioned approaches, which completely manage the lifecycle of application execution in the Cloud, many tools have been developed only focused on specific steps in this lifecycle.

For example, in Cloud resources provisioning there are tools like boto 1 or AWS to access the Amazon Web Services, and other IaaS agnostic like DeltaCloud, 2 Libcloud 3 and fog. 4 In addition, Nimbus is an open source IaaS system offering EC2, S3 and WSRF interface. It also provides tools to automate the deployment and configuration of a virtual cluster (Keeney and Freeman, 2008), and to manage the elasticity (Marshall et al., 2012).

Other tools focus on automating the installation, configuration and contextualization of applications, sometimes referred as

---

1 http://boto.readthedocs.org/en/latest/
2 http://deltacloud.apache.org/about.html
3 http://libcloud.apache.org/about.html
4 http://fog.io
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات