The Journal of Systems and Software 110 (2015) 239–252

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A programming-level approach for elasticizing parallel scientific applications

Guilherme Galante[a], Luis Carlos Erpen De Bona[b]

[a] Computer Science Department, Western Parana State University (UNIOESTE), Cascavel, PR, Brazil
[b] Informatics Department, Federal University of Parana (UFPR), Curitiba, PR, Brazil

ARTICLE INFO

Article history:
Received 8 May 2015
Revised 24 August 2015
Accepted 27 August 2015
Available online 8 September 2015

Keywords:
Cloud computing
Elasticity
Parallel applications

ABSTRACT

Elasticity is considered one of the fundamental properties of cloud computing. Several mechanisms to provide the feature are offered by public cloud providers and in some academic works. We argue these solutions are inefficient in providing elasticity for scientific applications, since they cannot consider the internal structure and behavior of applications. In this paper we present an approach for exploring the elasticity in scientific applications, in which the elasticity control is embedded in application source code and constructed using elasticity primitives. This approach enables the application itself to request or to release its own resources, taking into account the execution flow and runtime requirements. To support the construction of elastic applications using the presented approach, we developed the Cloudine framework. Cloudine provides all components necessary to construct and execute elastic scientific applications. The Cloudine effectiveness is demonstrated in the experiments where the platform is successfully used to include new features to existing applications, to extend functionalities of other elasticity frameworks and to add elasticity support to parallel programming libraries.

1. Introduction

Elasticity is considered one of the fundamental properties of the cloud (Han et al., 2014). It can be defined as the ability to adaptively scale resources up and down in order to meet varying application demands. It implies that the resources that compose the virtual environment may be added or removed on-the-fly and without service interruptions. Ideally, for the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time (Badger et al., 2011). This feature is suitable for dynamic applications, whose resources requirements cannot be determined exactly in advance, either due to changes in runtime requirements or in application structure (Jha et al., 2011).

Considering the importance of elasticity, several mechanisms to provide this feature are offered by public cloud providers, such as, Amazon EC2,[1] GoGrid[2] and Rackspace,[3] and by several academic works (Galante and Bona, 2012). These mechanisms were originally developed for dynamic scaling server-based applications, such as web, e-mail and database servers, to handle unpredictable workloads, and enabling organizations to avoid the downfalls involved with non-elastic provisioning (over and under-provisioning) (Chieu et al., 2009; Armbrust et al., 2010).

Most of current elasticity mechanisms are based in the monitoring of the external requests or in resources usage (processor load, memory usage, I/O requests) which can vary widely over time (Vaquero et al., 2011). The monitoring data are employed by an elasticity controller that makes decisions on whether the resources must be scaled or not, taking into account a set of conditions that when satisfied triggers some actions over the underlying cloud.

Although these mechanisms are used successfully in dynamic resources provisioning for server-based applications, we argue that these solutions may be inefficient in providing elasticity for scientific applications (Wang et al., 2012; Galante and Bona, 2013). First: Scientific application workloads are not defined by external requests, disabling the use of monitoring of external requests. Second: monitoring systems generally collect information from VM, neglecting the behavior and structure of the application that runs in the VM. Third: the consumption of resource in scientific applications is different from that presented in server-based applications. The former tends to consume all resources assigned, independently from the amount provided and the latter consumes the resources according to workload variation.

Moreover, we must take into account that there are several applications models (e.g., serial, multithread, single program multiple data, master-worker, MapReduce, etc.), each one with its own im-

E-mail address: guilherme.galante@unioeste.br (G. Galante).

[1] http://aws.amazon.com/. Last Access: Mar. 2015.
[2] http://www.gogrid.com/. Last Access: Mar. 2015.
[3] http://www.rackspace.com/. Last Access: Mar. 2015.

http://dx.doi.org/10.1016/j.jss.2015.08.051
0164-1212/© 2015 Elsevier Inc. All rights reserved.

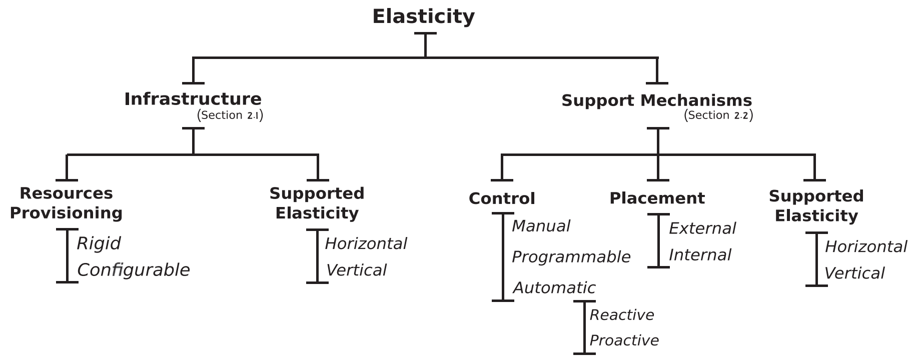

Fig. 1. Elasticity mechanisms classification.

plementation details and execution behavior particularities. Thus, to estimate accurately the applications demands, it would be necessary a specialized controller for each class of application. Trying to address this issue, a couple of academic researches have developed solutions to enable the development of elastic scientific applications in specific models, e.g., workflows (Byun et al., 2011), MapReduce (Chohan et al., 2010; Iordache et al., 2012), Message Passing Interface (MPI) (Raveendran et al., 2011) and master–slave applications (Rajan et al., 2011).

In this paper, we present an approach for enabling the development of elastic parallel scientific applications in several models and independent from monitoring systems and user interaction. In our proposal, the elasticity control is embedded in application source code and the elasticity actions (allocation and deallocation of resources) are performed by application itself, relying on its internal structure and behavior. The development of the embedded elasticity control is based in the concept of *elasticity primitives*, which are functions that provide support for representing and reflecting dynamic demand and translating it into dynamic requests of resources.

Using the elasticity primitives it is possible to construct new elastic applications as well as to elastify existing applications without having to rebuild them using cloud frameworks. The elastic applications are able to adjust their own resources according to runtime requirements or due to changes in execution flow (e.g., different solvers with different processing demands). It also enables the construction or adaptation of parallel processing libraries and APIs for transparently support elasticity.

To support the construction of elastic applications using the presented approach, we developed Cloudine, a framework for building and execution of elastic applications in IaaS clouds. Our framework provides a platform, that controls the resources provisioning and an application-programming interface (API), which provides the elasticity primitives set. The Cloudine effectiveness is demonstrated in the experiments where the platform is successfully used to include new features to existing (or legacy) applications, to extend functionalities of other elasticity frameworks and to add elasticity support to well-known parallel programming libraries.

The remainder of the paper is organized as follows. Section 2 presents the state-of-art related to cloud elasticity. After analyzing the current elasticity solutions, Section 3 points out some research opportunities in the area. In Section 4 we present the programming level elasticity approach. Section 5 introduces the Cloudine framework and presents its architecture and operation. Section 6 shows some experiments. Finally, Section 7 concludes the paper.

2. Cloud elasticity: state-of-art

Elasticity is defined as the ability of a system to dynamically add or remove computational resources used by an application or user to match the current demand as closely as possible (Herbst et al., 2013). Resources can include everything from single virtual processors (VCPU) to a complete virtual cluster. The concept could also be extended to applications. An elastic application is able to adapt itself to handle changes in resources or to request or release resources according to demands. To be able to take advantage of elasticity, it is necessary that both architecture and application support the feature in some form.

Recently, several elasticity solutions have been developed by public providers and by academy. In this section we present a classification of existing solutions and establish the state-of-the art of elasticity in computational clouds. This classification was created after analyzing 6 public clouds infrastructures, 3 private cloud platforms and 28 elasticity mechanisms, and later, extracting their main characteristics. Fig. 1 present the proposed classification.

At the first level, the solutions are separated into two groups: (1) *elastic architectures* and (2) *elasticity support mechanisms*. In the first group we analyze the support offered by the elasticity of the cloud infrastructure and in the second group we consider the characteristics of the mechanisms used to provide elasticity to applications that run on IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) clouds.

2.1. Elastic infrastructures

The cloud computing model emerged as an attractive alternative to the acquisition and management of infrastructure resources, allowing users requesting, using and releasing resources with a flexibility not found in other computational models.

The elasticity provided by clouds infrastructures are inherent to the use of virtualization techniques and to the availability of a large amount of physical resources. However, the manner it is provided to the user varies for each cloud platform according to how resources are offered and which elasticity type is supported.

Resources can be provided in two different modes: *fixed* or *configurable*. In fixed mode, virtual machines (VMs) are offered with a predefined configuration of CPU, memory and I/O (called *instance types* by Amazon and *server sizes* in GoGrid and Rackspace). The problem in providing resources in such way occurs when users cannot map their specific demands into one of the configurations offered by the provider. In configurable mode users can customize VM resources according to their needs. Although this model is the more appropriate to the cloud concept, the configurable mode is available in few cloud provides, such as Profitbricks[4] and CloudSigma.[5]

Depending on how the cloud implements the provisioning of resources we can classify its elasticity as horizontal or vertical (Vaquero et al., 2011). In the horizontal approach, the number of instances (VMs) is increased or decreased. On the other hand, the vertical

4 https://www.profitbricks.com/. Last Access: Mar. 2015.
5 http://www.cloudsigma.com/. Last Access: Mar. 2015.