Future Generation Computer Systems 78 (2018) 176–190

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications

Rodrigo da Rosa Righi [a,*], Vinicius Facco Rodrigues [a], Gustavo Rostirolla [a], Cristiano André da Costa [a], Eduardo Roloff [b], Philippe Olivier Alexandre Navaux [b]

[a] Applied Computing Graduate Program, UNISINOS, Brazil
[b] Parallel and Distributed Processing Group, UFRGS, Brazil

HIGHLIGHTS

- We are proposing a lightweight plug-and-play elasticity service for self-organizing resource provisioning.
- Based on the TCP (Transmission Control Protocol) congestion control, we propose an algorithm named Live Thresholding (LT).
- The results highlight performance competitiveness in terms of application time (performance) and cost (performance × energy) metrics.
- This article presented the Helpar model, which can be seen as an elasticity service for HPC applications.

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form
8 December 2016
Accepted 12 February 2017
Available online 16 February 2017

Keywords:
Cloud elasticity service
High-performance computing
Live Thresholding
Resource management
Self-organizing

ABSTRACT

Today cloud elasticity can bring benefits to parallel applications, besides the traditional targets including Web and critical-business demands. This consists in adapting the number of resources and processes at runtime, so users do not need to worry about the best choice for them beforehand. To accomplish this, the most common approaches use threshold-based reactive elasticity or time-consuming proactive elasticity. However, both present at least one problem related to the need of a previous user experience, lack on handling load peaks, completion of parameters or design for a specific infrastructure and workload setting. In this context, we developed a hybrid elasticity service for master–slave parallel applications named Helpar. The proposal presents a closed control loop elasticity architecture that adapts at runtime the values of lower and upper thresholds. The main scientific contribution is the proposition of the Live Thresholding (LT) technique for controlling elasticity. LT is based on the TCP congestion algorithm and automatically manages the value of the elasticity bounds to enhance better reactiveness on resource provisioning. The idea is to provide a lightweight plug-and-play service at the PaaS (Platform-as-a-Service) level of a cloud, in which users are completely unaware of the elasticity feature, only needing to compile their applications with Helpar prototype. For evaluation, we used a numerical integration application and OpenNebula to compare the Helpar execution against two scenarios: a set of static thresholds and a non-elastic application. The results present the lightweight feature of Helpar, besides highlighting its performance competitiveness in terms of application time (performance) and cost (performance × energy) metrics.

1. Introduction

Cloud computing has been gaining increasing adherence particularly because of it presents an easier way to auto-scale systems [1]. The elasticity service is better represented on Web-based business-critical systems, which must satisfy certain service level agreement (SLA), *e.g.*, upper bounds on user perceived response time [2–4]. Besides acting on performance and energy saving issues, horizontal and/or vertical elasticity actions are also especially pertinent on dynamic environments, where human intervention is becoming more and more difficult or even impossible [5]. In addition to the aforementioned client–server target, cloud elasticity is being more and more perceived as a key service to support the execution of HPC (High Performance Computing) applications. Traditionally, these kind of applications run on clusters or even in grid architectures: both

* Corresponding author.
E-mail address: rrrighi@unisinos.br (R. da Rosa Righi).

http://dx.doi.org/10.1016/j.future.2017.02.023
0167-739X/

have a fixed number of resources that must be maintained in terms of infrastructure configuration, scheduling tool and energy consumption. Besides hiding these procedures from programmers, an elasticity service when integrated with parallel applications can reveal one of its main contributions for the HPC context: self-organizing the number of resources (and processes or threads, consequently) in accordance with application's demands. Defining the exact number of processes is decisive on getting better performance, being sometimes estimated by hand through several tuning executions [6]. First, either short or large value will not efficiently explore the distributed system. Second, a fixed value cannot fit irregular applications, in which the workload varies during execution and/or occasionally is not predictable in advance.

Deciding the right amount of cloud computing resources to execute a parallel application is a double-edged sword, which may lead to either under-provisioning or over-provisioning situations [7,8]. These application-resource mappings are results of saturation or waste of resources, and are among the most significant challenges cloud elasticity clients are faced with. Today, most of the elasticity control strategies can be classified as either being reactive or proactive (also named by some authors as predictive) [5,7,9,10]. For the first case, typically users define an upper bound t_u and a lower bound t_l in an ad-hoc manner over a target metric (*e.g.*, CPU utilization level, throughput and average response time) to trigger, respectively, the activation and deactivation of a certain number of resources [6]. This threshold-based technique is the most used in Web-based commercial auto-scaling systems: its simplicity and intuitive nature drive this trend [3,11]. On the other hand, a proactive approach employs prediction techniques to anticipate the behavior of the system (its load) and thereby decide the reconfiguration actions. This capability in turn will enable the application to be ready to handle the load increase when it actually occurs. To accomplish this approach, it is common to use machine learning algorithms including Neural Network, Linear Regression, Support Vector Machine, Reinforcement Learning and Pattern Matching techniques [5]. To improve their accuracy on forecasting load values, they are commonly combined with popular Time-Series-based prediction techniques, such as Exponential Smoothing, Moving Averages and Autoregressive models [7].

Although the word automatic is used on both autoscaling mechanisms, current implementations commonly require some kind of user input, preliminary configuration and/or use of APIs (Application Programming Interface) to adjust resources as workloads change [1,10]. For the reactive approach, in particular, the tasks of choosing t_l and t_u and writing if-then rules are not trivial and sometimes require a deep knowledge about the behavior of the system over time [3,8,11]. This makes the accuracy of the policy subjective and prone to uncertainty: the same set of thresholds that fits fine a specific infrastructure/application possibly causes undesired emergent behaviors, such as instability and resource thrashing, on other settings [6,12]. In addition, other problem of using thresholds is related to the lack of reactivity. There are situations in which the cloud controller could anticipate the (de)allocation of resources, but the resource configuration remains the same due to bad choices on setting t_l and t_u. Although not needing thresholds, the proactive elasticity mechanism, on the other hand, is based on robust mathematical modeling and commonly classified adversely as time-consuming for sensitive performance-driven applications [7,13]. Besides runtime model tuning, there is also the need of training the predictive technique and previous execution of the application to optimize and select parameters [5]. Finally, Netto et al. [6] affirm that proactive elasticity strategies focus only on method accuracy and ignore cloud technical limitations, besides being very much dependent on workload characteristics and precise prediction models.

Considering the background, we are working with the following problem statement in mind: *how to provide a totally automatic elasticity service for parallel applications to bypass the aforementioned drawbacks related to proactive and reactive approaches*? To answer it, we designed an elasticity model called Helpar (Hybrid Elasticity Model for Parallel Applications). As a blending elastic approach, Helpar acts as a resource provisioning service at the PaaS (Platform as a Service) level of cloud joining the threshold-based lightweight feature from the reactive approach and the prediction and feedback control from the proactive way. In our understanding, a cloud scenario of "Complete Computing" must combine the keywords automatic, effortless, proactiveness, performance and intelligence. More precisely, our idea is to provide a plug-and-play parameterless model that on-the-fly adapts t_l and t_u in accordance with the application's demand to enhance the reactiveness of the system. The current version of Helpar addresses iterative master–slave parallel applications in such a way that users must only compile their applications with the middleware developed as a product of the Helpar model. Consequently, this middleware is in charge of transforming a non-elastic application in an elastic one without imposing any API or previous execution. Helpar provides a controller and a framework to manage resource (de)allocation and process (dis)connection without blocking the application while elasticity actions take place. Regarding the Helpar's scientific contributions, it brings the following additions in the state-of-the-art:

(i) A modeling of closed control-theoretic [14] infrastructure to support the hybrid elasticity behavior on parallel cloud-based applications;

(ii) Based on the TCP (Transmission Control Protocol) congestion control, we propose an algorithm named Live Thresholding (LT) to handle both application load projection and t_l and t_u adaptivity.

We designed LT to provide both elasticity reactiveness and application performance, but not neglecting energy consumption. In other words, it is not pertinent to reduce the application time by the half but spending four times more resources to accomplish this. In this way, Helpar evaluation analyzes performance (*time*) and energy consumption (*resource*), but also the cost metric ($time \times resource$) in comparisons with non-elastic and pure reactive elasticity approaches. We modeled and developed a numerical integration parallel application that was executed against the three aforesaid scenarios when varying the input workload pattern as follows: Ascending, Descending, Wave and Constant. In our understanding, despite using a single application, the strategy of adopting multiple evaluation metrics, scenarios and mainly different workloads was essential to discuss the resource provisioning proposal.

The remainder of this article will first introduce related studies in Section 2. Section 3 presents the Helpar model, revealing how we developed the aforesaid contributions. The evaluation methodology and the discussion of the results are described in Sections 4 and 5. Finally, Section 6 expresses the final remarks, highlighting the contributions with quantitative data.

2. Motivation and related work

Resource provisioning and cloud elasticity are topics of a vast number of research and scientific articles. Here, we present a set of initiatives both for Web and HPC applications that guide our motivation and research gaps.