Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Performance modelling and verification of cloud-based auto-scaling policies

Alexandros Evangelidis [*], David Parker, Rami Bahsoon

School of Computer Science, University of Birmingham, United Kingdom

HIGHLIGHTS

- Proposes a framework for formal reasoning about cloud-based auto-scaling policies.
- Presents a novel combination of performance modelling and formal verification.
- Validation is done using real experiments on Amazon EC2 and Microsoft Azure cloud.

ARTICLE INFO

Article history:
Received 14 June 2017
Received in revised form 4 November 2017
Accepted 26 December 2017
Available online xxxx

Keywords:
Auto-scaling
Markov models
Probabilistic verification

ABSTRACT

Auto-scaling, a key property of cloud computing, allows application owners to acquire and release resources on demand. However, the shared environment, along with the exponentially large configuration space of available parameters, makes the configuration of auto-scaling policies a challenging task. In particular, it is difficult to quantify, a priori, the impact of a policy on Quality of Service (QoS) provision. To address this problem, we propose a novel approach based on performance modelling and formal verification to produce performance guarantees on particular rule-based auto-scaling policies. We demonstrate the usefulness and efficiency of our techniques through a detailed validation process on two public cloud providers, Amazon EC2 and Microsoft Azure, targeting two cloud computing models, Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), respectively. Our experimental results show that the modelling process along with the model itself can be very effective in providing the necessary formal reasoning to cloud application owners with respect to the configuration of their auto-scaling policies, and consequently helping them to specify an auto-scaling policy which could minimise QoS violations.

1. Introduction

Cloud computing has become the most prominent way of delivering software solutions, and more software vendors are deploying their applications in the public cloud. In cloud computing, one of the key differentiating factors between successful and unsuccessful application providers is the ability to provide performance guarantees to customers, which allows violations in performance metrics such as CPU utilisation to be avoided [1]. In order to achieve this, cloud application providers use one of the key features of cloud computing: *auto-scaling*, which allows resources to be acquired and released on demand. While auto-scaling is an extremely valuable feature for software providers, specifying an *auto-scaling policy* that can guarantee no performance violations will occur is an extremely hard task, and "doomed to fail" [2] unless considerable care is taken. Furthermore, in order for a *rule-based* auto-scaling policy to be properly configured, there has to be an in-depth level of knowledge and a high degree of expertise, which is not necessarily true in practice [3,1].

Lately, public cloud providers such as Amazon EC2 and Microsoft Azure have increased the flexibility offered to users when defining auto-scaling policies, by allowing combinations of auto-scaling rules for a wide range of metrics. However, this "freedom" of being able to specify multiple auto-scaling rules comes at the cost of an extremely large configuration space. In fact, it is exponential in the number of performance metrics and predicates, making it virtually impossible to find the optimal values for the auto-scaling variables [4].

In addition, an auto-scaling policy consists not only of performance metrics thresholds, but also of *temporal parameters*, which often seem to be neglected, despite their significance in configuring a good auto-scaling policy. These parameters include the time interval that the auto-scaling mechanism looks back to determine whether to take an auto-scale action, and the duration for which it is prohibited from triggering auto-scale actions after a successful

[*] Corresponding author.
E-mail addresses: a.evangelidis@cs.bham.ac.uk (A. Evangelidis), d.a.parker@cs.bham.ac.uk (D. Parker), r.bahsoon@cs.bham.ac.uk (R. Bahsoon).

https://doi.org/10.1016/j.future.2017.12.047
0167-739X/© 2018 Elsevier B.V. All rights reserved.

auto-scale request (cool-down period). Since both of these parameters have to be specified by a human operator, it becomes a challenging task to understand the impact of these parameters on the performance metrics of the application running on the cloud. It is exactly this impact that we wish to quantitatively analyse.

As noted in [5], auto-scaling policies "tend to lack correctness guarantees". The ability to specify auto-scaling policies that can provide performance guarantees and reduce violations of Service Level Agreements (SLAs) is essential for more dependable and accountable cloud operations. However, this is a complex task due to: (i) the large configuration space of the conditions and parameters that need to be defined; (ii) the unpredictability of the cloud as an operating environment, due to its shared, elastic and on demand nature; and (iii) the heterogeneity in cloud resource provision, which makes it difficult to define reliable and universal auto-scaling policies. For example, looking at public cloud providers, one can observe that there is no guarantee on the time it will take for an auto-scale request to be served, nor whether the auto-scale request will receive a successful response or not.

In order to address the aforementioned challenges, we propose a novel approach based on performance modelling and formal verification. In particular, we use *probabilistic model checking* [6], which is a formal approach to generating guarantees about quantitative aspects of systems exhibiting probabilistic behaviour, and the tool PRISM [7]. Guarantees are expressed in quantitative extensions of temporal logic and numerical solution of probabilistic models is used to quantify these measures precisely. This approach provides a formal way of quantifying the uncertainty that exists in today's cloud-based systems and a means of providing performance guarantees on auto-scaling policies for application designers and developers. Another important novel aspect of our approach is the combination of probabilistic model checking with *Receiver Operating Characteristic (ROC)* analysis during empirical validation. This allows us not only to refine our original probabilistic estimates after collating real data and to validate the accuracy of our model, but also to obtain global QoS violation thresholds for the policies.

Our probabilistic model is a discrete-time Markov chain (DTMC), which we specify in the modelling language of the PRISM tool. It takes into account the stochasticity of auto-scale requests by transitioning between the different waiting times with a probability p, which can be specified by the user a priori, or left as a free parameter in order to probabilistically verify an auto-scaling policy under different values of p. We demonstrate the correctness and usefulness of this approach through an extensive validation, considering an Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) scenario running on the Amazon EC2 and Microsoft Azure cloud, respectively. We have made the data used to validate our model publicly available [8].

In order to validate the accuracy of our model, we perform *Receiver Operating Characteristic (ROC)* analysis, which is widely used in machine learning and data mining [9]. In a sense, we follow a similar approach to the validation of classification models, by treating the probability as a ranking measure which determines the likelihood of the event of interest, in our case the probability of a QoS violation. ROC can be used to help the decision maker select the appropriate classification threshold, by quantifying the trade-off between sensitivity and specificity. Additionally, it can be used to validate the accuracy of a classification model, by computing the AUC (Area Under Curve) metric, which is one of the most commonly used summary indices [10].

Our validation starts by ordering the probabilities that have been computed from our PRISM model for each auto-scaling policy. Then, we plot the respective ROC curve by computing the points for the respective thresholds [0..1]. Through this analysis, we are able to find the optimal threshold of discriminating between the auto-scaling policies that could result in a CPU/response time violation. Our criterion of optimality is the point which minimises the Euclidean distance between the ROC curve, and point (0,1), which is often called the point of "perfect classification". Also, this gives us the ability to refine our original violation estimates after we have seen the real data, and to obtain a global threshold for distinguishing between auto-scaling policies. After plotting the ROC curve, we compute the AUC, which in our case can be interpreted as the number of times our model can distinguish performance violations/non-violations of randomly selected auto-scaling policies. This is an "important statistical property" [9] of AUC, and one of the reasons that it has been used so widely for validating the performance of classifiers.

Our modelling and verification framework is intended to minimise the time and costs for cloud application owners who do not have the resources or desire to clone their existing applications in order to test them in a cloud-based environment. It can also provide valuable assistance in designing, analysing and verifying the auto-scaling policies of applications and services deployed on public clouds, and could help in providing robust performance guarantees.

Probabilistic model checking has previously been applied to a wide range of application domains, including aspects of cloud computing [11,5]. However, to the best of our knowledge, this is the first work to use probabilistic model checking to provide performance guarantees for auto-scaling policies from the perspective of a cloud application provider. We believe it is also the first case where the results of probabilistic verification have been validated using VMs running on major public cloud providers, Amazon EC2 and Microsoft Azure, and the first usage of ROC analysis to validate such results.

This paper is an extended version of our earlier work on performance modelling and verification of auto-scaling policies [12]. In particular, this paper investigates the generalisability of the approach to a second public cloud provider, Microsoft Azure, dealing with auto-scaling policies at the PaaS, rather than IaaS, level. It also extends the techniques to consider temporal properties of the auto-scaling policies, and how their variation can affect the performance guarantees that can be offered to users.

The remainder of the paper is structured as follows. In Section 2, we provide some background on auto-scaling policies and probabilistic verification. Then, in Section 3, we describe our approach to constructing a probabilistic model of the auto-scaling process. Sections 4 and 5 present the process for applying probabilistic verification to the model and validating the results, respectively. Section 6 discusses the results, and then we conclude by surveying the relevant literature and identifying directions for future work.

2. Preliminaries

2.1. Rule-based auto-scaling policies

An *auto-scaling policy* [4] defines the conditions under which capacity will be added to or removed from a cloud-based system, in order to satisfy the objectives of the application owner. Auto-scaling is divided into scaling-up/-down and scaling-out/-in methods, with the two approaches also being defined as *vertical* (add more RAM or CPU to existing VMs) and *horizontal* (add more "cheap" VMs) scaling. In this work, we focus on scaling-out and -in since it is a commonly used and cost-effective approach.

The main auto-scaling method that is given to application providers by all public cloud providers today (e.g., Amazon, Microsoft Azure, Google Cloud) is *rule-based*. The rule-based method is the most popular and is considered to be the *state-of-the-art* in auto-scaling an application in the cloud [13].

In a rule-based approach, the application provider has to specify an upper and/or lower bound on a performance metric (e.g., CPU