Future Generation Computer Systems 55 (2016) 200–212

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Automated configuration support for infrastructure migration to the cloud

Jesús García-Galán [a,*], Pablo Trinidad [a], Omer F. Rana [b], Antonio Ruiz-Cortés [a]

[a] Escuela Técnica Superior de Ingeniería Informática, Avda. Reina Mercedes s/n, 41012, University of Seville, Seville, Spain

[b] School of Computer Science and Informatics, Cardiff University, Queen's Buildings, Newport Road, Cardiff CF24 3AA, UK

HIGHLIGHTS

- We support the decision making in migration planning to the cloud.
- We use Feature Models to describe the configuration space of an IaaS.
- We automate the search of the most suitable IaaS configuration.
- Our approach improves the results of commercial applications on Amazon EC2.

ARTICLE INFO

Article history:
Received 30 June 2014
Received in revised form
13 January 2015
Accepted 9 March 2015
Available online 19 March 2015

Keywords:
EC2
Automated analysis
Cloud migration
Feature model
IaaS

ABSTRACT

With an increasing number of cloud computing offerings in the market, migrating an existing computational infrastructure to the cloud requires comparison of different offers in order to find the most suitable configuration. Cloud providers offer many configuration options, such as location, purchasing mode, redundancy, and extra storage. Often, the information about such options is not well organised. This leads to large and unstructured configuration spaces, and turns the comparison into a tedious, error-prone search problem for the customers. In this work we focus on supporting customer decision making for selecting the most suitable cloud configuration—in terms of infrastructural requirements and cost. We achieve this by means of variability modelling and analysis techniques. Firstly, we structure the configuration space of an IaaS using feature models, usually employed for the modelling of *variability-intensive* systems, and present the case study of the Amazon EC2. Secondly, we assist the configuration search process. Feature models enable the use of different analysis operations that, among others, automate the search of optimal configurations. Results of our analysis show how our approach, with a negligible analysis time, outperforms commercial approaches in terms of expressiveness and accuracy.

1. Introduction

The clear benefits of cloud-based infrastructures are increasing the number of companies that are migrating their private and expensive data centres to the cloud. An *Infrastructure as a Service (IaaS)* enables the dynamic provisioning of computational and data resources (often on-demand), reducing costs (for short term workloads), speeding up the start-up process for many companies, and decreasing resource and power consumption (among other benefits).

Deciding the most suitable provider is often challenging, as each provides a number of possible configurations. As an example of the dimension of this problem, there are over 100 public cloud providers [1], and just for *Elastic Compute Cloud (EC2)* [2], the *Amazon Web Services (AWS)* computing service, we have identified 16,991 different configurations.[1] As each user/company that plans to use a cloud computing infrastructure is likely to have their own specific requirements, it is necessary to identify the most relevant provider and subsequently the most suitable configuration. Identifying such configuration within a large potential search space is a tedious and error-prone task that requires for an automated support.

* Corresponding author. Tel.: +34 659108324.
E-mail addresses: jegalan@us.es (J. García-Galán), ptrinidad@us.es (P. Trinidad), ranaof@cardiff.ac.uk (O.F. Rana), aruiz@us.es (A. Ruiz-Cortés).

[1] The basis for this number is explained further in this paper.

http://dx.doi.org/10.1016/j.future.2015.03.006
0167-739X/

In recent times, software tools and research contributions have emerged to support this decision process, but we have found these to have limitations, providing either incomplete configuration spaces or/and imprecise results. On the commercial side, providers such as Amazon or Rackspace provide tools that suggest specific configurations for migrating an on-premise infrastructure [3,4]. However, such tools ignore some configuration options, forcing for example in the case of Amazon to choose Linux as the only operating system. Other companies, like CloudScreener [5], provide their own comparators to decide which provider and configuration best fit user needs. Nonetheless, such tools lack information about the configuration space they work with, and some tests have revealed false positives in their optimal results. Recent academic work [6,7] also suffers from similar concerns, as they generally only consider a small subset of the available configurations of services like EC2 or Azure virtual machines. In order to overcome these limitations, any approach must ensure for each provider to model its complete configuration space.

In this work, we assist the customer in determining the most suitable configuration of an IaaS. For that purpose, we present the case study of AWS EC2. As outlined previously by us [8], AWS is one of the most variable and complex providers in terms of configuration options and pricing. Indeed, understanding EC2 configuration space can be challenging, as it is scattered across several pages, tables and paragraphs. We believe that modelling a complex provider eases the task of modelling simpler providers. Additionally, AWS is one of the most widely used IaaS providers, being present in all the current configuration tools. For focusing on one provider, enables us to check their precision and compare to our approach. Among all the different services AWS offers, we focus on EC2 and *Elastic Block Storage (EBS)*—additional disk for computing instances, which are considered as core infrastructure services.

We interpret an IaaS as a *variability-intensive* system, so we can rely on variability modelling and analysis techniques to support the configuration process. In particular, we propose the modelling of IaaS – and EC2 in concrete – as *Feature Model (FMs)*, a kind of model widely used for variability-intensive systems. In this way, first we represent the configuration space in a complete, structured and compact manner, and second we provide the user with a model to ease the configuration process. This modelling enables the use of the so-named *Automated Analysis of Feature Models (AAFM)*, a set of analysis operations that extracts information from the models, which we subsequently use to assist decision-making. We use some of them to verify the validity and completeness of the FM with respect to the service configuration space, and to determine which configuration is the most suitable for any given requirements. We interpret the most suitable configuration as the one that meets customer's requirements and optimises the cost. Our approach presents two main benefits: first, we consider the complete configuration space, so that the real optimal solution is obtained for given customer requirements; second, assisting the configuration process for such a highly-configurable service like AWS EC2 enables the same approach to be used for other providers, such as Azure or Rackspace.

For evaluation purposes, we (i) verify our proposed model, (ii) compare our approach to existing commercial applications in term of expressiveness and accuracy, and (iii) check and improve the performance of our approach. To verify our model, we describe the EC2 FM using a plain-text language, extract the list of configurations within our model, and check that it matches exactly the available configurations of EC2.[2] The validation of the analysis is performed by means of two different implementations: FaMa Framework – a well-known tool for the AAFM – and a reasoner based on the IBM CPLEX solver. We compare the performance of both approaches, where the latter implementation shows improved and negligible execution times when calculating the most suitable configuration. We also compare the obtained results with the output of CloudScreener, which can be improved by the use of our approach.

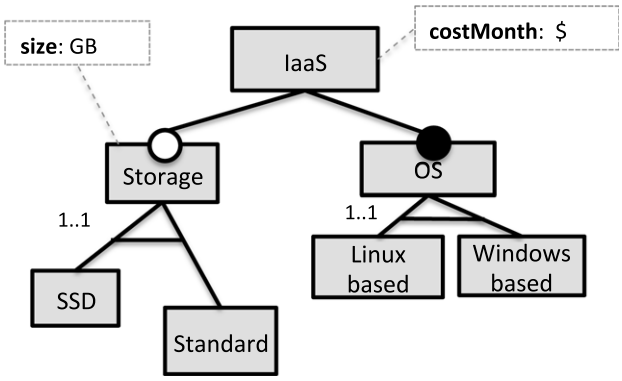

Fig. 1. Example of a FM.

This paper extends our previous work [8] in several ways. In particular, we provide (i) an explicit description of the configuration problem, (ii) a modelling methodology to describe the configuration space of an IaaS as a FM, (iii) a verification of the configuration space represented by the FM by means of analysis operations and (iv) an evaluation of the expressiveness, accuracy and performance of our approach.

The rest of the paper is structured as follows: Section 2 briefly describes the state of the art in variability modelling and analysis. Section 3 states the problem we tackle in this work. Section 4 describes our modelling methodology for the configuration space of an IaaS, while Section 5 presents a modelling case study with Amazon EC2. Section 6 presents our analysis approach for the search of the optimal configuration, and explains the details of the analysis operations. We present an implementation of our approach in Section 7, and we evaluate it in Section 8. Related work is described in Section 9. Finally, Section 10 discusses our proposal and proposes future directions in our research.

2. Feature models

Feature Models (FMs) [9] are used to represent all the possible products that can be built in variability-intensive systems such as *Software Product Lines (SPLs)*. FMs are tree-like data structures where each node represents a product feature. Fig. 1 shows a FM that represents general features of a fictional IaaS provider. Features are bound by means of hierarchical (mandatory, optional and set) and cross-tree relationships. These relationships define how features can be combined in a product, defining the configuration space of the system. In a FM, a feature does not necessarily represent a specific functionality but can be used as abstract features [10] which represent domain decisions such as `Linux based` feature (Fig. 1). `IaaS` is the root feature that represents the overall functionality of the system. It has two children, an optional feature (white circle) named `Storage`, and a mandatory feature (black circle) named `OS`. Both features present set relationships whose cardinality indicates the number of child features that can be chosen at the same time.

FMs can also have attributes that represent non-functional properties, leading to attributed FMs. These attributes are linked

[2] We exclude spot instances and micro instances since they are not intended to be persistent, and EBS optimised instances because we do not consider IOPS provisioning for EBS.