Journal of Network and Computer Applications 95 (2017) 26–41

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Auto-scaling web applications in clouds: A cost-aware approach

Mohammad Sadegh Aslanpour[a,*], Mostafa Ghobaei-Arani[b,*], Adel Nadjaran Toosi[c]

[a] Department of Computer Engineering, Jahrom Branch, Islamic Azad University, Jahrom, Iran
[b] Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran
[c] Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia

ARTICLE INFO

Keywords:
Auto-scaling
Resource provisioning
Cloud resource
Cost-aware
Web application
Service level agreement (SLA)

ABSTRACT

The elasticity feature of cloud computing and its pay-*per*-use pricing entice application providers to use cloud application hosting. One of the most valuable methods, an application provider can use in order to reduce costs is resource auto-scaling. Resource auto-scaling for the purpose of preventing resource over-provisioning or under-provisioning is a widely investigated topic in cloud environments. The Auto-scaling process is often implemented based on the four phases of MAPE loop: Monitoring (M), Analysis (A), Planning (P) and Execution (E). Hence, researchers seek to improve the performance of this mechanism with different solutions for each phase. However, the solutions in this area are generally focused on the improvement of the performance in the three phases of the monitoring, analysis, and planning, while the execution phase is considered less often. This paper provides a cost saving super professional executor which shows the importance and effectiveness of this phase of the controlling cycle. Unlike common executors, the proposed solution executes scale-down commands via aware selection of surplus virtual machines; moreover, with its novel features, surplus virtual machines are kept quarantined for the rest of their billing period in order to maximize the cost efficiency. Simulation results show that the proposed executor reduces the cost of renting virtual machines by 7% while improves the final service level agreement of the application provider and controls the mechanism's oscillation in decision-making.

1. Introduction

With the rapid development of cloud computing, nowadays, instead of purchasing computing infrastructure, many application providers (APs) tend to host their applications on cloud resources offered by cloud providers (CPs). Cloud providers such as Amazon EC2 (EC2) offer resources to the AP in the form of Virtual Machines (VMs) with the scalability feature and pay-*per*-use charging model (Lorido-Botran et al., 2014; Coutinho et al., 2015; Qu et al., 2016). The AP, the application, and application users can be a webmaster, online store website, and end users, respectively.

Since the AP, in particular, the Web application provider is aware of the dynamics of the Web environment and end users requests, static resource provisioning is not efficient. The reason is that in static resource provisioning, with increased rate of incoming user requests, resource under-provisioning occurs which consequently results in interruption or delayed response to user requests. On the other hand, in the period of reduced traffic, the issue of resource over-provisioning occurs and as a result increased AP costs arise (Qu et al., 2016; Arani and Shamsi, 2015). Therefore, considering the various pricing models in the cloud (Qu et al., 2016; Shen et al., 2014), the AP usually prepays a minimum number of resources for its permanent and long-term needs to receive a discount for this type of rental (for example, reserved instances in EC2 receive a discount of up to 75%). Consequently, with load fluctuations, the AP seeks to use the short-term rental model to cover its temporary needs (for example on-demand machines in the form of pay per hourly use). However, this approach is not enough as it requires a mechanism capable of automatically determining the capacity and the number of rented on-demand resources proportional to the incoming load (Amiri and Mohammad-Khanli, 2017).

Presentation of an efficient auto-scaling mechanism is a research topic which is mainly faced with the challenge of maintaining a balance between cost reduction and the Service Level Agreement (SLA). IBM proposes a model for autonomous management of auto-scaling mechanism in the form of MAPE (Monitor-Analyze-Plan-Execute) loop as a reference model (Computing, 2006). The MAPE loop model can be applied to implement a cloud web application system which knows its state and reacts to its changes. Therefore, the majority of auto-scaling mechanisms are based on the MAPE loop (Lorido-Botran et al., 2014; Qu et al., 2016; Mohamed et al., 2014; Ghobaei-Arani et al., 2016;

* Corresponding authors.
E-mail addresses: aslanpour.sadegh@gmail.com (M.S. Aslanpour), mostafaghobaye@yahoo.com, m.ghobaei@qom-iau.ac.ir (M. Ghobaei-Arani), anadjaran@unimelb.edu.au (A. Nadjaran Toosi).

http://dx.doi.org/10.1016/j.jnca.2017.07.012
Received 18 February 2017; Received in revised form 13 July 2017; Accepted 17 July 2017
Available online 18 July 2017

Weingärtner et al., 2015). MAPE-based mechanisms constantly repeat the four general processes of the monitoring, analysis, planning, and execution, in a way that a monitor iteratively gathers information about resources, for example, the status of resource utilization. After monitoring, the auto-scaling mechanism indicates the analysis process (Qu et al., 2016) to start which can be simple or complicated; simple analysis is the use of raw information obtained by the monitor, while complex analysis discovers knowledge from information using methods such as artificial intelligence or machine learning (Amiri and Mohammad-Khanli, 2017; Ghobaei-Arani et al., 2017a). Afterward, by matching obtained analyses to a series of rules predefined by the AP, the planner makes scale-up or down decisions (rule-based planner (Qu et al., 2016)). The final phase of the MAPE cycle is the execution of the decision by the executor. This is when the auto-scaling mechanism needs to send a request for instantiation of a new VM or release of the one of the VMs previously rented from the CP. This research focuses on improving the performance of the executor in the resource auto-scaling mechanism with a cost-aware approach.

The motivation behind the improvement of the executor's performance lies in the following: Thanks to the possibility of selecting different types of VMs with various capacities, APs can rent a large number of VMs of different types simultaneously; considering the intense workload fluctuation in the Web environment, this is highly possible to happen (Singh and Chana, 2016). That said, if the auto-scaling mechanism makes a scale-down decision, the executor needs to select from a diverse set of rented VMs and release one. The basic question posed here is whether it matters which VM is selected? If the answer is yes, what policy is the best to be used for this selection? Unlike Amazon's auto-scaling policy that always selects the oldest VM for release (as default executor) and according to the dark spots seen in related research (Ghobaei-Arani et al., 2016; Islam et al., 2012; Huang et al., 2012; Bankole and Ajila, 2013; Ajila and Bankole, 2013; Herbst et al., 2014; Qavami et al., 2014; García et al., 2014; Singh and Chana, 2015; Fallah et al., 2015; de Assunção et al., 2016), this selection should be made cautiously and rigorously. This is because, firstly, the CP calculates partial billing as full billing (billing cycle) (Moldovan et al., 2016; Li et al., 2015). For example, in the EC2 service, billing is carried out on an hourly basis and the release of a VM for a duration of 4 h and 1 min would result in billing for 5 h. Therefore, policy making for minimizing the minutes wasted in the release of surplus VMs is an important economic matter for the AP. Secondly, due to unresolved load balancing challenges (Khan and Ahmad, 2016), candidate VMs are probably processing different workloads and the influence of releasing each VM on the SLA would vary. Hence, the first purpose of the present research is to employ novel policies, especially cost saving ones, in the selection of surplus VMs (professional executor).

A research gap can be still seen after the selection of the surplus VM and before its release. On the one hand, it is likely that the selector did not manage to find a VM with exactly X hours of use. In this situation, the release of that VM would impose extra costs. On the other hand, a scale-up decision may be made immediately after the release of the surplus VM; in this case, the delayed startup of the new VM is a challenge which negatively affects SLA (Lorido-Botran et al., 2014; Coutinho et al., 2015; Qu et al., 2016). Due to the unpredictability of the Web environment (Panneerselvam et al., 2014; Gholami and Arani, 2015) or maladjustment of scaling rules (Qu et al., 2016), it is highly likely that the mechanism to be affected by contradictory actions when the mechanism is in an oscillation condition (Lorido-Botran et al., 2014; Qu et al., 2016). As a result, the following hypothesis is put forward: If the selected surplus VM stays rented by the AP until the last minute of the bill, it can possibly be used for the improvement of the scaling mechanism's performance. Therefore, the other goal of this research is to offer an executor with the ability to quarantine the surplus VM until the billed hour is completed in order to resolve the challenge of delayed VM startup (super professional executor - Suprex). This is while to date, researchers merely considered benefits of vertical scaling or applying cooling time in the execution of the commands as the method for overcoming this challenge (Qu et al., 2016).

This paper presents a scaling mechanism equipped with a super professional executor (Suprex) with a cost-aware approach. We seek to show that the execution phase of the MAPE cycle can play an effective role in cost saving. We explain all four phases as they are required for the full implementation of the auto-scaling mechanism. This is also required for better understanding of the paper. The auto-scaling mechanism offered in this research is different from others where the focus is mainly on improving the mechanism's performance in the monitoring, analysis, and planning phases rather than the execution phase. The reason why the execution phase was overlooked lies in the fact the actions are often considered under the control of the CP and the CP is considered as a black box. However, by applying an architecture with full control (Casalicchio and Silvestri, 2013) from the AP's perspective, the power is granted to the AP in the execution of all scaling commands. The main contributions of this research are as follows:

1. We designed an auto-scaling mechanism based on the MAPE concept for Web applications,
2. We enhanced the effectiveness of the execution phase of the control MAPE loop with a cost-aware approach,
3. We provided an innovative solution for overcoming the challenges of delayed VM startup,
4. We designed an executor in order to mitigate oscillation and increase the stability of the mechanism, and
5. We conducted a series of experiments to evaluate the performance of proposed approach under real-world workload traces for different metrics.

The rest of the article is organized as follows: Section 2 provides the necessary background. Section 3 includes related work; Section 4 fully explains the proposed approach. Section 5 simulates and evaluates the performance of the Suprex executor. Section 6 discusses the experimental results in details. Finally, Section 7 presents conclusions and future work.

2. Background

This section provides a brief overview of autonomic computing and application adaptation.

2.1. Autonomic computing

The increasing complexity of computer systems makes it essential to handle them autonomically. IBM's Autonomic Computing Initiative has helped to define the four properties of autonomic systems: self-configuration, self-optimization, self-healing, and self-protection (Computing, 2006). Cloud computing providers manage their data centers in an efficient way, taking cues from well-established autonomic computing practices. Particularly, tasks like VM provisioning, disaster recovery, capacity management, etc. are performed autonomically (Dhingra, 2014). To effectively manage cloud deployed web applications, we need to react to regularly occurring or anticipated events in the system (Ghobaei-Arani et al., 2017b). In (Computing, 2006), IBM proposes a model for autonomous management in the form of an autonomic MAPE-K loop as a reference model. The MAPE-K loop model can be applied to implement a cloud web application system which knows its state and reacts to changes in it. This model details different components that allow an autonomic manager to self-manage properties, namely Monitor, Analyze, Plan, Execute and Knowledge (Rheddane, 2015). The MAPE-K loop model is depicted in Fig. 1 and discussed in the remaining part of this section.

The MAPE-K loop model is applied to auto-scaling of web applications in the cloud environment. The auto-scaling process of web