An Overview of Elastic Cloud Applications

Technical University of Vienna,
Advanced Internet Computing Lecture,
(Jan 2014)

Soodeh Farokhi*
1228800

Gajo Gajic
0828150

Martin Kalany
0825673

Jia Wei
0035204

ABSTRACT

Cloud computing enables access to an almost unlimited amount of resources, but in order to realize this feature, the Cloud provider should be able to support the elastic deployment of applications. Elasticity, the ability to rapidly scale resources up and down on demand, is one of the main advantages brought by the cloud paradigm and makes it different to an âĂIJadvanced outsourcingâĂİ solution. However, there are various challenges to understand the elasticity requirements of a given application and several approaches try to tackle this issues. In this paper, we are aim to investigate the state of the art of elastic cloud applications in one hand and on the other hand, describe the requirements for supporting elasticity on Cloud. It is because the authors had a working experiment with CloudScale [10], [9] in order to compare the process of deployment an application, twitter-based sentiment analysis, on top of an IaaS (Amazon EC2[1]) by using CloudScale and without using it directly on a PaaS (Google AppEngine [2]).

1. INTRODUCTION

Nowadays more and more enterprises decide to migrate to Cloud environment, in order to save the expenses of maintaining a private data center and utilize the core features of cloud computing in which resources are dynamically increased and decreased on demand, and charging is consumption based. However, public IaaS Cloud providers face serious challenges in order to to understand the elasticity characteristics of applications, and workloads while considering the capacity of their own Cloud platform.

Ideally a cloud platform is infinitely and instantaneously elastic. Based on this consumption, an application can be scaled out indefinitely with increasing load, and this can happen as fast as the load increases with no degradation of response times [1].

Therefore, in this picture of an ideal Cloud, resources are available instantly and the application is immediately deployed and available to be used. However, supporting this scenario, is not easy to be tackled by Cloud IaaS or PaaS provider. In this paper, we introduce the existing approaches for this problem by considering CloudScale [10] features, advantages and disadvantages over other works as a focus of this work. We will also talk briefly about an experimental comparison between the deployment of a twitter-based sentiment analysis application on Amazon EC2 as a public Cloud IaaS by CloudScale with the deployment of it on Google AppEngine as a public Cloud PaaS without utilizing the CloudScale features as a middleware. It is worth mentioning that, although elasticity is interpreted as the capability of both scaling up and down, while scalability more is used for scaling up, in this paper we used them interchangeable,.

The rest of paper is organized as follows. Section 2 discusses the possible ways to provide elastic applications on Cloud and the essential features to support it is explained. Then in Section **??**, which is the main focus of the paper, the state of the art of elastic cloud applications will be presented in two categories, research work and commercial tools and technologies. In Section 4 the differences between CloudScale, as a tool which authors had experienced with, and other similar approaches is introduced briefly. Finally, Section 5 concludes the paper.

2. ELASTICITY REQUIREMENTS

Talking generally, based on [12] IaaS or PaaS automatic elasticity is typically achieved in Cloud by using a set of provider-defined rules that govern how and when the service should scales up or down to adapt to a variable deployed application load. Actually, these rules are a set of conditions that when met, triggers some actions on the infrastructure or platform in order to support automatic elasticity. The abstraction level of this process for the user, the customization of rules and the degree of automation are different among existing approaches.

For example, some approaches make it possible for users to build some simple conditions by using fixed infrastructure or platform metrics such as CPU, memory, while other approaches offers service level metrics such as cost to benefit ratio in order to support more complex conditions which are combinations of simple rules. Another aspect that existing approaches are different in is the way they behave when the supported conditions are met. Figure 1[12] simply shows possible mechanisms of how existing approaches support the

*In alphabetical order
[1]http://aws.amazon.com/ec2/
[2]https://developers.google.com/appengine/

elasticity on the level of cloud IaaS or PaaS.

Service horizontal scaling can be divided to two sub-categories, whether adding new server replicas and load balancers to distribute load among all available replicas, or dynamic bandwidth allocation by supporting network scaling. Vertical scaling can be achieved by changing the instances on-the-fly (without rebooting) whether by resizing (e.g. dedicated more physical resources such as CPU and RAM to a running virtual machine) or replacing. However, on-the-fly changes on the available resources of a virtual machine is not supported by the most common operating systems. Some work like [11] tried to facilitate this process by proposing a new abstraction layer closer to the lifecycle of services that allows for their automatic deployment and escalation depending on the service status (not only on the infrastructure). Their proposed abstraction layer sits on top of different cloud providers, hence mitigating the potential lock-in problem and allowing the transparent federation of clouds for the execution of services.

Apart from supporting server scalability, some other characteristics need to be considered that affect the overall application scaling potential such as load balancers and related algorithms. Load balancer is needed to support the aggregation of new servers (typically in the form of new VMs) to be able to distribute load among several servers. As a commercial public Cloud IaaS provider, Amazon already have some strategies for load balancing of replicated VMs via the Elastic Load Balancing capabilities [3]. Therefore, having several servers and the mechanisms to distribute load among them is a certain step towards scaling a cloud application [11]. However, network scalability is to be said a neglected element that should also be considered [14] for Cloud datacenters in order to be able to support application elasticity. It is because several VMs share the same network, potentially producing a huge increase in the required bandwidth. Regarding to Figure 1, the aforementioned CloudScale middleware as well as as well as Aneka[4] and AppScale[5], while will be introduced later, fall into "container replication" category in the platform layer (the container is CloudScale here).

3. STATE OF THE ART

In the following, we summarize the most significant works which support elastic Cloud applications, based on two main categories of the existing approaches. At the first category, we exploit the research work in this subject, while the next part the most significant commercial products, technologies and tools regarding to elastic Cloud application will be introduced.

3.1 Related Research (scientific approaches)

Authors of [7] proposes a framework contributes by describing neces- sary interfaces, functionalities, and data exchanges to deploy complex application across several Cloud IaaS, such as Amazon EC2 in a dynamic and adaptive way. In the other world, they answer the question of "How to deploy elastic applications?" by presenting a flexible framework that supports high-level interfaces for an adaptation plug-in. These interfaces simplify the retrieval of necessary input data for all placement algorithms support state-full applications, or complex application architectures.

In [4] Aeolus component model is proposed to capture similar scenario from realistic cloud deployments, and specifies compositions of services to automate deployments, planning of day-to-day activities such as software upgrade planning, service deployment, and elastic scaling.

Work presented in [1] introduces an elasticity mechanisms of a typical Cloud IaaS platform (inspired by Amazon EC2) and presents a Service Oriented Performance Modeling method and tool to model and predict the elasticity characteristics of three realistic applications and workloads on this cloud platform. They compare the pay-as-you-go instance costs and end-user response time for these three elasticity scenarios. Their proposed model is able to predict the elasticity requirements (in terms of the maximum instance spin-up time) for the working scenarios.

The OSGi-inspired component framework COSCA presented in [6] auto- matically manages elastic deployment of component-based application by isolating components of different applications and hides distribution using a virtualized and distributed OSGi-like framework. It eases the usage of cloud resources and scalability for component-based applications.

The authors of [5] adopts a lightweight approach along with its algorithm to enable cost-effective elasticity for cloud applications. The proposed approach operates fine-grained scaling at the resource level (CPUs, memory, I/O) in addition to VM-level scaling to efficiently scale cloud applicationâĂŹs resources up and down in order to meet the given QoS requirements while reducing cloud providersâĂŹ costs.

3.2 Related tools and commercial approaches

Several application provisioning solutions exist, enabling developers and administrators to declaratively specify deployment requirements and dependencies to support repeatable and managed resource provisioning such as Opscode Chef [6], Puppet[7] and juju[8]. In juju basic services are described as predefined charms and it can fall into the category in which it supports elastic applications by providing help, hints, or triggers.

Aneka [13] is a .NET-based platform that focuses on enabling hybrid cloud applications by employing a specialized programming model. It is able to deploy containers and run users applications in several IaaS providers. Aneka is more similar to Grid computing middleware, provides a relatively low-level abstraction based on the message passing interface (MPI). In general, Aneka seems more suitable for building scientific computing applications instead of enterprise applications.

AppScale [3] is an open source extension to the Google Ap- p Engine (GAE) PaaS that allows users to build their own GAE compliant PaaS on top of any private or public IaaS service. Indeed, it provides a framework to investigate the interaction between PaaS and IaaS systems. In order to support the elasticity, it scales the VMs used to host containers depending on actual application demand, automatically configuring the load balancers. It targets Online Transaction Processing (OLTP) style enterprise applications.

Carina Environment Manager[9] automates and speeds up

[3] http://aws.amazon.com/autoscaling/
[4] http://www.manjrasoft.com/products.html
[5] http://www.appscale.com
[6] www.opscode.com/chef/
[7] http://puppetlabs.com
[8] http://juju.ubuntu.com
[9] https://github.com/blackberry/OpenNebula-Carina

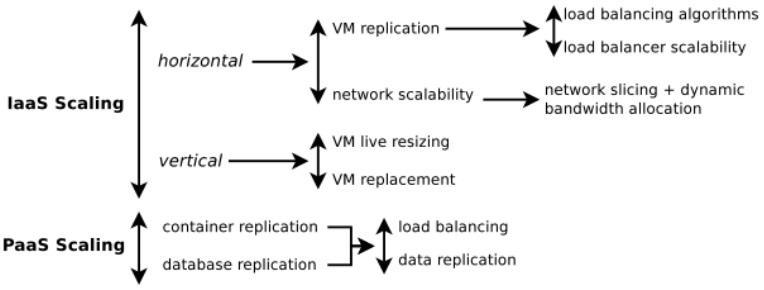

Figure 1: Possible mechanisms to support elasticity on Cloud IaaS/PaaS [12]

the deployment of services onto the OpenNebula IaaS platform. It supports the automated creation and run-time scaling of multi-VM application environments according to policies. It leverages the OpenNebula contextualization framework to setup clusters of VMs in a master-slave configuration or a set of workers with an IP load-balancer in front. Policies can be defined to control how VMs are added or removed based on manual, time of day, or application load-based triggers.

3Teras AppLogic[10] in another commercial tools to automate complex application deployment. It scales applications without changing code and architecture. Indeed, it provides an on demand scaling by assigning resources to the service as a single entity, rather than a collection of components.

Orleans, introduced in [8] and [2] is a software framework developed at Microsoft Research to build reliable, scalable, and elastic cloud applications. It includes programming model that encourages the use of simple, easy to understand and employ concurrency patterns. It is based on distributed actor-like components called grains, which are isolated units of state and computation that communicate through asynchronous messages. Orleans enables a developer to concentrate on application logic, while the Orleans runtime provides scalability, availability, and reliability.

CloudBees RUN@Cloud[11] is a service which provides continuous integration and an elastic platform for hosting Enterprise Java Beans (EJB) applications.

4. CLOUDSCALE FEATURES

In this section, very briefly, we enumerate the features of CloudScale and compare it with the previous introduced approaches in Section 3.

CloudScale is a middleware to build applications on top of Cloud IaaS. It provides an abstraction that makes elastic applications running on top of an IaaS seem like regular, non-distributed Java applications. Indeed, it places a middle layer between IaaS offerings, which provide great control over the application, but do so at the costs of high deployment effort, and PaaS offerings, which are easy to use, but provide little control [10]. It allows application developers to have full control over their application, while by using a PaaS, they cannot retain it. Another advantages of using CloudScale is providing an abstract layer by which applications are not bound to any specific cloud providers, and are easy to migrate, work well in the context of private or hybrid clouds, and support a wider variety of applications, while still providing an abstraction comparable to commercial PaaS solutions. CloudScale has a more general claim in comparision with similar approached for elastic applications, which is able to handle a wide variety of elastic application types, including data-intense, processing-intense and OLTP[12] style web applications[9].

Based on [10] the main difference between CloudScale and the introduced approached in Section 3 are by using CloudScale, developers retain full control over their application. Although CoudScale hides some scalability-related issues from developers, they are still free to customize the way CloudScale works to their own needs, either by implementing customscaling policies, adapting the CloudScale framework itself, or managing some so called Cloud-Objects (regular program-level objects that are abstractions of application logics, and should be distributed over a cloud) in the application manually. The main disadvantage of all of those approaches is that they imply a significant loss of control for the developer and also they typically require the usage of a given public cloud (typically provided by the same vendor), imply the usage of proprietary APIs, and restrict the types of applications that are supported [9].

5. CONCLUSION

In this paper we present the state of the art in elastic Cloud application which include both existing research work and technologies. We also took a look at requirements of supporting elasticity on Cloud. As the authors had experienced with deploying twitter-based sentiment analysis application on IaaS by interfering of CloudScale and without it directly on Google AppEngine, advantages of using CloudScale over existing approaches were discussed briefly as well.

6. REFERENCES

[1] P. C. Brebner. Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (iaas) cloud applications. In *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*, pages 263–266. ACM, 2012.

[2] S. Bykov, A. Geller, G. Kliot, J. R. Larus, R. Pandya, and J. Thelin. Orleans: cloud computing for everyone. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 16. ACM, 2011.

[10] http://www.ca.com/us/cloud-platform.aspx

[11] http://www.cloudbees.com/

[12] Online Transaction Processing

[3] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski. Appscale design and implementation. 2009.

[4] R. Di Cosmo, S. Zacchiroli, and G. Zavattaro. Towards a formal component model for the cloud. In *Software Engineering and Formal Methods*, pages 156–171. Springer, 2012.

[5] R. Han, L. Guo, M. M. Ghanem, and Y. Guo. Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644–651. IEEE, 2012.

[6] S. Kächele and F. J. Hauck. Component-based scalability for cloud applications. In *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, pages 19–24. ACM, 2013.

[7] M. Keller, M. Peuster, C. Robbert, and H. Karl. A topology-aware adaptive deployment framework for elastic applications. In *Intelligence in Next Generation Networks (ICIN), 2013 17th International Conference on*, pages 61–69. IEEE, 2013.

[8] J. R. Larus. Look up!: your future is in the cloud. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, pages 1–2. ACM, 2013.

[9] P. Leitner, Z. Rostyslav, W. Hummer, C. Inzinger, P. Leitner, W. Hummer, and C. Inzinger. CloudScale : Efficiently Implementing Elastic Applications for Infrastructure-as-a-Service Clouds. Technical report, 2013.

[10] P. Leitner, B. Satzger, W. Hummer, C. Inzinger, and S. Dustdar. Cloudscale: a novel middleware for building transparently scaling cloud applications. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 434–440. ACM, 2012.

[11] L. Rodero-Merino, L. M. Vaquero, V. Gil, F. Galán, J. Fontán, R. S. Montero, and I. M. Llorente. From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, 26(8):1226–1240, 2010.

[12] L. M. Vaquero, L. Rodero-Merino, and R. Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[13] C. Vecchiola, X. Chu, and R. Buyya. Aneka: a software platform for .net-based cloud computing. *High Speed and Large Scale Scientific Computing*, pages 267–295, 2009.

[14] H. Wu and B. Kemme. A unified framework for load distribution and fault-tolerance of application servers. In *Euro-Par 2009 Parallel Processing*, pages 178–190. Springer, 2009.