# 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

(lots of mistakes and ambiguities) Cloud computing provides access to a virtually unlimited amount of resources. 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 of the cloud paradigm and makes it different to an "advanced outsourcing" solution. However, there are various challenges to understand the elasticity requirements of a given application and several approaches try to tackle these issues. In this paper, we investigate the state of the art of elastic cloud applications and describe the requirements for supporting elasticity in a cloud environment. 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

(lots of mistakes and ambiguities) More and more enterprises decide to migrate business applications to a cloud environment to utilize the core features of cloud computing, which are, among others: The ability to dynamically increase or decrease computing power on demand as well as a more flexible pay-by-usage model that helps reducing costs associated with running and maintaining a private data/server center. Those costs are significant especially if e.g., high availability and short response times are a requirement. Public cloud providers face serious challenges in order to understand the elasticity characteristics of applications and workloads while having to consider the required capacity of their cloud platforms.

Ideally, a cloud platform is infinitely and instantaneously elastic, meaning that infinite computing resources are available and that the scaling up of an application can be done instantaneously. Based on this assumption, an application can be scaled out indefinitely with increasing load without increasing response times [1].

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 3, 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

In general, IaaS or PaaS automatic elasticity in a cloud environment is typically achieved by using a set of provider-defined rules that govern how and when the service should scale up or down to adapt to a varying application load [12]. These rules are a set of conditions that when met trigger some actions on the infrastructure or platform in order to support dynamic scaling. Existing approaches differ greatly in the abstraction level of this process, the customization of rules and the degree of automation.

While some approaches allow the user to specify only simple conditions by using fixed predefined set of metrics such as CPU and memory usage, other approaches offers service level metrics (e.g., cost-to-benefit ratio) in order to allow the user to specify more complex conditions that may further be combinations of simple rules. Existing approaches furthermore differ in the way they behave when the supported conditions are met. Figure 1 [12] depicts possible

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

mechanisms of elasticity support on the level of cloud IaaS or PaaS.

Scaling can roughly bew divided in two approaches, called horizontal and vertical scaling. Horizontal scaling is done by either adding new server replicas and load balancers to distribute the load among more servers, or through (what does the following mean? Network scaling is no where defined.) dynamic bandwidth allocation by supporting network scaling. Vertical scaling can be achieved by changing the instances on-the-fly[3] either by resizing (e.g. dedicating more physical resources such as CPU and memory to a running virtual machine) or replacing (what?). However, on-the-fly changes of resources dedicated to a virtual machine instance are 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, which allows for their automatic deployment and escalation (what?)depending on the service status (not only on the infrastructure). Their proposed abstraction layer sits on top of different cloud providers, hence mitigating a potential lock-in and allowing for a transparent federation of clouds.

Apart from server scalability, load balancing is another major issue for scalability. A load balancer is required to distribute load as evenly as possible among servers or VMs. As a commercial public Cloud IaaS provider, Amazon already has some strategies for load balancing for replicated VMs via the Elastic Load Balancing capabilities [4]. Therefore, having several servers and the mechanisms to distribute load among them is a necessary step towards scaling a cloud application [11].

However, network scalability (according to Fig. 1, this is only an issue for IaaS) is a somewhat neglected element that should also be considered [14] for cloud datacenters in order to be able to support application elasticity. Because several VMs share the same network, potentially producing a huge increase in the required bandwidth, the network too has to be scalable. Regarding to Figure 1, the aforementioned CloudScale middleware as well as Aneka[5] and AppScale[6], which we will introduce in Section 3.2, fall into the "container replication" category (what is this?) in the platform layer (the container is CloudScale here) (how is cloudscale connected to the Aneka and AppScale?).

# 3. RELATED WORK

In the following, we provide an overview of significant work dealing with elastic cloud applications, where both scientific research and commercial products, technologies and tools are covered.

## 3.1 Scientific research

Keller et. all [7] propose a framework contributes by describing necessary interfaces, functionalities, and data exchanges to deploy complex application across several Cloud IaaS, such as Amazon EC2 in a dynamic and adaptive way (what??). In the other world (which? PaaS?), 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 (what??).

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(what??).

Work presented in [1] introduces an elasticity mechanism 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(this doesn't really say much).

The OSGi-inspired component framework COSCA [6] automatically manages the elastic deployment of component-based applications 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.

Han et. all [5] adopt 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 resources up and down in order to meet the given QoS requirements while reducing the cloud providersâĂŹ costs.

## 3.2 Commercial approaches, technologies and tools

Several application provisioning solutions exist, enabling developers and administrators to declaratively specify deployment requirements and dependencies to support repeatable and managed resource provisioning. Examples are *Opscode Chef*[7], *Puppet*[8] and *juju*[9]. 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 (what?).

Aneka [13] is a .NET-based platform that focuses on enabling hybrid cloud applications (what??) by employing a specialized programming model. It is able to deploy containers and run user applications on several IaaS providers. Aneka is 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 than enterprise applications.

*AppScale* [3] is an open source extension to the Google App 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. To provide elasticity, it scales the VMs used to host containers depending on actual application demand, automatically configuring the load balancers. It targets Online Transaction Processing

[3] without rebooting the machine
[4] http://aws.amazon.com/autoscaling/
[5] http://www.manjrasoft.com/products.html
[6] http://www.appscale.com
[7] www.opscode.com/chef/
[8] http://puppetlabs.com
[9] http://juju.ubuntu.com

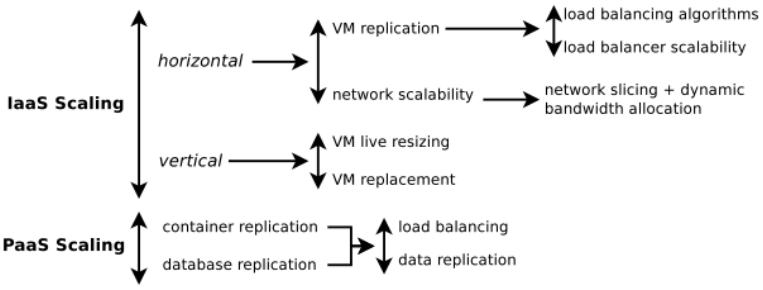


**Figure 1: Possible mechanisms to support elasticity on Cloud IaaS/PaaS [12]**

(OLTP) style enterprise applications.

*Carina Environment Manager*[10] automates and speeds up 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 (which?) 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.

CA AppLogic[11] is another commercial tool to automate complex application deployment. It scales applications without changing code or architecture. Indeed, it provides 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 a programming model that encourages the use of simple, easy to understand (what??) and employs 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[12] is a service which provides continuous integration and an elastic platform for hosting Enterprise Java Beans (EJB) applications.

(many, many tools listed, but I don't think we capture the essance)

# 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 which can not been retained using a PaaS. Another advantage of CloudScale is that it provides an abstract layer so that applications are not bound to any specific cloud providers. Thus they areeasy to migrate, work well in the context of private or hybrid clouds while still providing an abstraction comparable to commercial PaaS solutions. Compared to similar approaches for elastic applications, CloudScale strives to be a more general tool, which is able to handle a wide variety of elastic application types, including data-intense, processing-intense and OLTP[13] style web applications[9].

Based on [10] the main difference between CloudScale and the introduced approaches in Section 3 is that by using CloudScale, developers retain full control over their application. Although CloudScale hides some scalability-related issues from developers, they are still free to customize the way CloudScale works to their own needs, either by implementing custom scaling 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 disadvantages of the other approaches are that they imply a significant loss of control for the developer and typically require the usage of a given public cloud (usually provided by the same vendor). Furthermore they may 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 applications research which include both current scientific research as well as commercial approaches. We also investigate neccassary requirements for supporting elasticity in a cloud environment. Advantages and disadvantages of CloudScale are discussed, based on the experience gained by the authors by deploying a twitter-based sentiment analysis application both using CloudScale on the Amazon EC2 cloud (IaaS) and the PaaS Google AppEngine.

# 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.

[10] https://github.com/blackberry/OpenNebula-Carina

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

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

[13] Online Transaction Processing

[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.

[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.