An Overview of Elastic Cloud Applications

Technical University of Vienna,
Advanced Internet Computing,
(Jan 2013)

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, explain a summary of working experiment with CloudScale, as a middleware to build elastic applications on Cloud IaaS. We also provide the comparison result of deployment the same application, twitter-based sentiment analysis, on top of an IaaS (Amazon EC2[1]) by using CloudScale and without it 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 [9] 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 **??**, which is the main focus of the paper, discusses the state of the art of elastic cloud applications. Section 3 presents the experiences of the authors with deploying an application on Amazon EC2 by using CloudScale as a middleware versus on Google AppEngine directly. Finally, Section 4 concludes the paper.

2. STATE OF THE ART

In the following, we summarize the most significant works related to 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. The main focus on this paper is on the research work.

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

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

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 elasticity on the level of cloud IaaS or PaaS.

Service horizontal scaling can be devided to two sub-categories, whether dding 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 being CloudScale in this case)

2.1 Container-level Scalability

This category is like CloudScale approach (description of this kind of approach is described in [12].

Aneka [13]: For high loads, and to avoid overprovisioning of resources, it would be useful to be able to federate clouds so components can be run in external/public clouds if needed. Aneka is able to deploy containers and run users applications in several IaaS providers [12].

AppScale [3]: Platforms will require container replicas to be deployed or released dynamically to handle load variations. AppScale can scale the VMs used to host containers depending on actual application demand, automatically configuring the load balancers [12].

We can consider two main categories for the WR:

2.2 Related Research (scientific approaches)

(1) paper [10] (MDA): Enough about standardization, let's build cloud applications

(2) paper [6]:(cite CloudScale) A topology-aware adaptive deployment framework for elastic applications

(2-1) **Ubuntu Juju** [6]: Basic services are described as predefined charms

(2-2) **RIM. Carina Environment Manager** [7]: It automate and speed up the deployment of services onto the OpenNebula private clouds

(2-3) CA 3Teras[8]http://www.ca.com/us/cloud-platform.aspx: 3Teras AppLogic automates complex application deployment.

(2-4)**Aeolus** /citedi2012towards: AeolusâĂŹs component model specifies compositions of services to automate deployments, planning of day-to-day activities such as software upgrade planning, service deployment, elastic scaling

(3)Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (IaaS) cloud applications [1]

(4)Component-based scalability for cloud applications [5]

(5)Look up!: your future is in the cloud [7]

(6)Orleans: cloud computing for everyone [2]

(7)Lightweight Resource Scaling for Cloud Applications [4]

Aneka is a .NET-based platform with a focus on enabling hybrid cloud applications. Unlike CloudScale, Aneka is more akin to traditional Grid computing middleware, providing a relatively low-level abstraction based on the message passing interface (MPI). In general, Aneka seems suitable for building scientific computing applications, but less so for enterprise applications [8]

AppScale, on the other hand, specifi- cally (and exclusively) targets Online Transaction Processing (OLTP) style enterprise applications [8].

another platform with a focus on data processing has been proposed by the BOOM research project [4]. BOOM builds on a custom, declarative programming model and is clearly geared towards data analytics [8]..

The main contribution of CloudScale over all of these models is that Cloud- Scale has amore general claim, and is able to handle a wide variety of elastic application types, including data-intense, processing-intense and OLTP style web applications [8]..

The European commission funded project **Contrail** [2] is currently building another PaaS for hosting relatively flexible elastic applications, called ConPaaS [28] (Contrail PaaS). ConPaaS supports web and high-performance applications built in Java or PHP, and can be used in combination with

[3]http://aws.amazon.com/autoscaling/

[4]http://www.manjrasoft.com/products.html

[5]http://www.appscale.com

[6]https://juju.ubuntu.com/

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

[8]

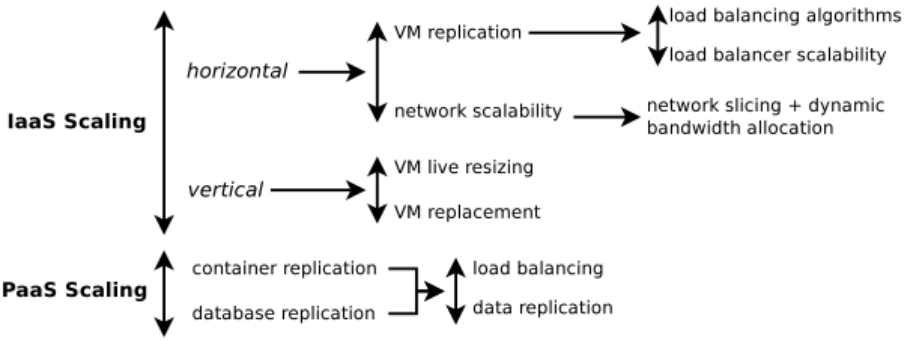

Figure 1: Possible mechanisms to support elasticity on Cloud IaaS/PaaS [12]

Amazon EC2 or an OpenNebula private cloud. However, ConPaaS seems to operate on a lower level of abstraction than CloudScale. This PaaS environment consists of elastically hostedWeb services, such as web or database servers, and mostly integrates existing stand-alone components [8].

2.3 Related tools and commercial approaches

A service particularly related to our work is **CloudBees RUN@Cloud** [9], which provides continuous integration and an elastic platform for hosting Enterprise Java Beans (EJB) applications [8].

Other PaaS of- ferings that support the Java programming language include the well-known GAE, AmazonâĂŹs Elastic Beanstalk services, or MicrosoftâĂŹs Azure [8].

Outside of the Java world, **Heroku** is gaining traction as a provider of PaaS for dynamic scripting languages, such as Ruby or Python[10] [8].

AppScale can be also considered as a commercial product and it allows users to build their own GAE compliant PaaS on top of any private or public IaaS service. We consider this system a significant step forward in comparison to other vendors [8].

Many core features and terms of CloudScale, as introduced in Section 3.1, are akin to well-known Java remote procedure call technology, e.g., Java RMI or EJB. However, note that the main contribution of CloudScale is not in enabling remoting (as RMI and EJB do), but rather in enabling elasticity. Hence, the actual conceptual overlap between CloudScale and these frameworks is not particularly large. In this paper, we more focus on the first category by investigating the scientific state of the art in elastic cloud applications, while just introduce some existing tools and technologies for the second category [8].

2.4 Other aspects

1. efficiently deploying applications in the cloud

2. distributed deployment and hosting of (Java) applications: most replicating and clustering of **Enterprise Java Beans** (EJB) implementations lack the dynamism of CloudScale, and do not provide resource management strategies or transparent code distribution [9].

3. Specialized programming models: **Aneka** is a platform for deployment of .NET-based applications, employing a specialized programming model. The BOOM initiative at UC Berkeley aims at simplifying declarative programming for the cloud. Whereas BOOM is mostly suited for data analytics, Cloud- Scale is targeted at general-purpose programming and deployment of arbitrary Java applications [9].

4. PaaS and SaaS: From a technical viewpoint, these providers do little more than allowing scalable remote access to existing software developing platforms, enriched with rich user interface and Web 2.0 experience; The main difference between CloudScale and those commercial SaaS and PaaS solutions is that, when using CloudScale, developers retain full control over their application. That is, even though 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 COs in the application manually [9].

5. Java-based remoting: e.g., **Java RMI, Enterprise Java Beans** (EJB) or CORBA. These frameworks, while technically similar, provide only limited support for automated scaling. How- ever, the ideas behind the CloudScale middleware could be implemented on top of these technologies, enhancing, e.g., the CORBA remoting model with automated load balancing in the cloud.

3. CLOUDSCALE FEATURES AND EXPERIENCES

CloudScale provides an abstraction that makes elastic applications running on top of an IaaS cloud seem like regular, non-distributed Java applications.

1. CloudScale, on the other hand, allows application developers to retain full control over their application. CloudScale applications are not bound to any specific cloud provider, 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 [8]..

2. The main disadvantage of all of those systems 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 propri- etary APIs, and restrict the types of applications that are supported [8].

3. CloudScale middleware for transparently scaling appli-

[9]http://www.cloudbees.com/
[10]https://www.heroku.com/

cations using an IaaS cloud.

4. The CloudScale approach provides a middle ground between common 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. [9]

4. CONCLUSION

In this paper we present the existing research and technologies for elastic Cloud application as well as the experiences that authors had with working with CloudScale as a middleware for building scaling Cloud applications transparently.

5. 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] J. Miranda, J. Guillén, J. M. Murillo, and C. Canal. Enough about standardization, let's build cloud applications. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, pages 74–77. 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.