Managing and Monitoring Elastic Cloud Applications

Demetris Trihinas, Chrystalla Sofokleous, Nicholas Loulloudes, Athanasios Foudoulis, George Pallis, Marios D. Dikaiakos

Department of Computer Science, University of Cyprus
{trihinas,stalosof,loulloudes.n,afoudo01,gpallis,mdd}@cs.ucy.ac.cy

Abstract. Next generation Cloud applications present elastic features and rapidly scale their comprised resources. Consequently, managing and monitoring Cloud applications is becoming a challenge. This paper showcases the functionality and novel features of: (i) c-Eclipse, a framework for describing Cloud applications along with their elasticity requirements and deploying them on any IaaS provider; and (ii) JCatascopia, a fully-automated, multi-layer, interoperable Cloud monitoring system. Particularly, we demonstrate how a user can manage the full lifecycle of a three-tier web application and observe, in real-time, how an elasticity management platform automatically scales the application based on various user-defined elasticity requirements, workloads and performance metrics.

1 Introduction

Cloud computing offers organizations the opportunity to reduce IT costs and improve the efficiency of their services. The next generation of Cloud applications present elastic features which allow them to expand or contract their allocated resources in order to meet their exact demands. However, managing and monitoring elastic Cloud applications is not a trivial task. For instance, organizations with large-scale distributed web applications (e.g. online video streaming services) require a deployment comprised of multiple virtual instances, which often have complex inter-dependencies. Despite the fact that current elasticity management platforms such as Amazon Auto Scaling[1] and Rackspace Auto Scale[2] can automatically and seamlessly scale large Cloud applications, these systems are proprietary and limit the application to operate only on specific Cloud platforms. Portability imposes a level of complexity and additional effort, from the Cloud consumer perspective, to *move* an application from one IaaS provider to another. Another downside of the aforementioned systems is that they only handle simplistic boolean requirements based on a limited number of low-level metrics (i.e. CPU, memory usage, etc.) and only support fine-grained elasticity actions (e.g. add/remove virtual instances). Tiramola [1] on the other hand, is an open-source alternative which succeeds in accommodating complex elasticity requirements based on application-level metrics but it is limited to only scale NoSQL databases.

Furthermore, elasticity management requires the monitoring of elastic applications, a challenge that is left unaddressed by many of the existing monitoring tools (i.e. Ganglia[3], Nagios[4]). The complex nature of this task requires for the monitoring system

[1] http://aws.amazon.com/autoscaling/
[2] http://www.rackspace.com/cloud/monitoring/
[3] http://ganglia.sourceforge.net/
[4] http://www.nagios.org/

to run in a fully automated manner, detecting configuration changes in the application topology which occur due to elasticity actions (e.g. a new VM is added) or when allocated resource-related parameters change (e.g. new disk attached to VM).

To address the aforementioned challenges which occur when managing and monitoring elastic Cloud applications, we present two powerful open-source tools:

c-Eclipse: a client-side Cloud application management framework which allows developers to describe, deploy and manage the lifecycle of their applications in a clean and graphical manner, under a unified environment. c-Eclipse is built on top of the well-established Eclipse platform. It adheres to two highly desirable Cloud application features: portability and elasticity. Current frameworks such as the ServiceMesh Agility Platform[5] limit users to describe and deploy their applications to a small number of Cloud platforms for which connectors are available and when users want to *move* their application to another Cloud, the description must be altered. Additionally, current frameworks offer limited or no elasticity support by only allowing the definition of fine-grain elasticity requirements. c-Eclipse ensures application portability by adopting the open TOSCA specification[6] for describing the provision, deployment and application re-contextualization across different Cloud platforms. In contrast to other frameworks which also adopt the TOSCA specification [2], c-Eclipse does not require for its users to have any prior knowledge of the TOSCA specification since users describe their application through an intuitive graphical interface which automatically translates the description into TOSCA. Finally, c-Eclipse facilitates the specification of complex, multi-grained elasticity requirements via the SYBL [3] directive language.

JCatascopia [4]: a fully-automated, multi-layer, interoperable Cloud monitoring system. JCatascopia addresses the aforementioned challenges by being able to run in a non-intrusive and transparent manner to any underlying virtualized infrastructure. Current monitoring systems which provide elasticity support [5] [6], require for special entities at the physical level or depend on the current hypervisor to detect topology changes. In contrast to these systems, JCatascopia uses a variation of the publish and subscribe protocol [7] to dynamically detect, at runtime, when monitoring agents have been added/removed from the overall system due to elasticity actions. This is accomplished without any human intervention, special entities or dependence on the underlying hypervisor, allowing users to even monitor applications distributed over multiple Cloud platforms. In addition, JCatascopia provides filtering capabilities to reduce the overhead for metric distribution and storage, and generates high-level application metrics dynamically at runtime by aggregating and grouping low-level metrics.

2 Elasticity Management Platform

In this section we focus on describing the components which comprise an elasticity management platform which incorporates c-Eclipse and JCatascopia.

A developer, at first, uses the c-Eclipse **Application Description Tool** to graphically describe the application topology, software dependencies and elasticity requirements. The graphical description is translated, on the fly, into TOSCA. Then, the developer

[5] `http://www.servicemesh.com`

[6] `http://docs.oasis-open.org/tosca/TOSCA/v1.0/`

[7] Monitoring Agents (metric producers) subscribe to Monitoring Servers instead of the other way around, allowing for them to (dis-)appear dynamically [4]

selects a Cloud provider and via the c-Eclipse **Submission Tool**, the description is submitted to the **Cloud Manager** for deployment. Subsequently, the Cloud Manager parses the portable and platform independent TOSCA description, and initiates the deployment of the Cloud application via the **Orchestrator**. The Orchestrator consists of two subcomponents. The first component is the **Cloud Orchestrator**, which is the interface that interacts with the IaaS provider to (de-)allocate the requested Cloud resources. The second component is the **App Orchestrator**, which performs the execution of application specific scripts and ensures the configuration and deployment correctness.

After successfully deploying a Cloud application, users are able, via c-Eclipse, to monitor the deployment, acquire aggregated monitoring metrics from **JCatascopia**, and configure the deployment by refining the elasticity requirements. The **Cloud Manager** (Fig. 1) constantly checks the user-defined elasticity requirements and when a violation is detected, resizing actions are issued. Specifically, the Cloud Manager requests from the **Cloud Orchestrator** to add/remove instances depending on the application demands and from the **App Orchestrator** to configure the application accordingly.

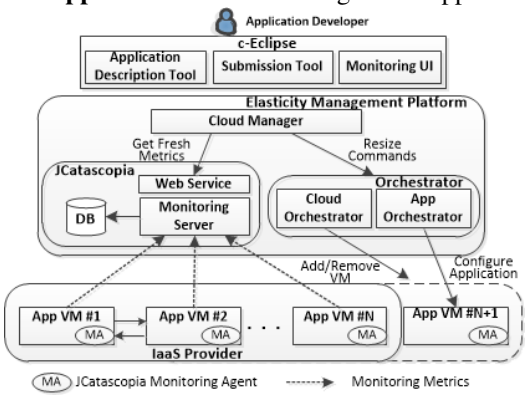

Fig. 1: Elasticity Management Platform

3 Demonstration Description

This demonstration showcases[8] the functionality of the proposed platform by managing the full lifecycle of an elastic Cloud application. Specifically, we will: (i) describe, via c-Eclipse, a Cloud application's topology, software dependencies and relationships between its tiers; (ii) define, via c-Eclipse, elasticity requirements for the elastic components comprising the application; (iii) select a Cloud platform and submit the generated TOSCA description to the Cloud Manager; (iv) monitor both the Cloud resources allocated for the application and its performance by utilizing JCatascopia (Fig. 2); and (v) scale the application, via the Cloud Manager, based on collected metrics and the user-defined elasticity requirements. It must be noted that both the Cloud Manager and Orchestrator are simplistic components developed only to showcase the full potential of c-Eclipse and JCatascopia. Furthermore, attendees may configure the deployment by refining the elasticity requirements. Finally, users will observe real-time graphs for each collected metric, configure monitoring parameters (i.e. sampling rate) and generate graphs by aggregating metrics originated from multiple instances.

[8] Screenshots of c-Eclipse and JCatascopia can be found at: http://linc.ucy.ac.cy/CELAR/icwe2014

Use case scenario: we consider a three-tier online video streaming service comprised of: (i) an HAProxy[9] *load balancer* which distributes client requests (i.e. download, upload video) across multiple application servers. (ii) An *application server tier*, where each instance is an Apache Tomcat[10] server containing the video streaming web service. Aggregated tier metrics such as the average *number of connections* and/or *request throughput* can be used to decide when a scaling action should occur; (iii) a Cassandra[11] NoSQL *distributed data storage backend*. Similarly, aggregated metrics such as the average *CPU utilization* and/or *query latency* can be used to scale the Cassandra ring. To stress the video service, we have developed a *workload generator* where the workload form (i.e. sinusoidal, linear), type (i.e. read-heavy, write-heavy, combination) and parameters (i.e. intensity, max execution time) are all configurable.

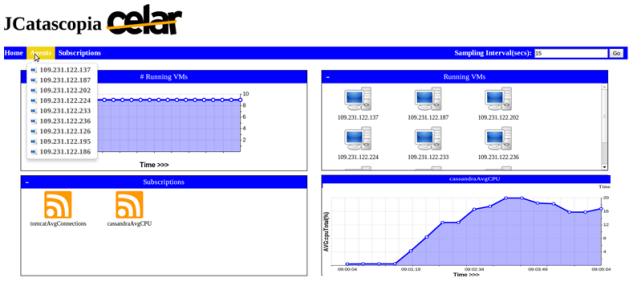

Fig. 2: Screenshot from JCatascopia while running the demo scenario

4 Acknowledgements

This work was partially supported by the European Commission in terms of the CELAR 317790 FP7 project (FP7-ICT-2011-8).

References

1. Tsoumakos, D., Konstantinou, I., Boumpouka, C., Sioutas, S., Koziris, N.: Automated, Elastic Resource Provisioning for NoSQL Clusters Using TIRAMOLA. IEEE International Symposium on Cluster Computing and the Grid (2013) 34–41
2. Kopp, O., Binz, T., Breitenbcher, U., Leymann, F.: Winery a modeling tool for tosca-based cloud applications. In: Service-Oriented Computing. Volume 8274 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 700–704
3. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications. In: 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. (2013) 112–119
4. Trihinas, D., Pallis, G., Dikaiakos, M.D.: JCatascopia: Monitoring Elastically Adaptive Applications in the Cloud. In: 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. (2014)
5. Clayman, S., Galis, A., Mamatas, L.: Monitoring virtual networks with lattice. In: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP. (2010) 239–246
6. de Carvalho, M.B., Granville, L.Z.: Incorporating virtualization awareness in service monitoring systems. In: Integrated Network Management, IEEE (2011) 297–304

[9] http://haproxy.1wt.eu/
[10] http://tomcat.apache.org/
[11] http://cassandra.apache.org/