CELAR: Automated Application Elasticity Platform

Ioannis Giannakopoulos*, Nikolaos Papailiou*, Christos Mantas*, Ioannis Konstantinou*,
Dimitrios Tsoumakos† and Nectarios Koziris*
*National Technical University of Athens, School of ECE
{ggian,npapa,cmantas,ikons,nkoziris}@cslab.ece.ntua.gr
†Department of Informatics, Ionian University, Corfu, Greece
dtsouma@ionio.gr

Abstract*—One of the main promises of the cloud computing paradigm is the ability to scale resources on-demand. This feature characterizes the cloud era, where the overhead of early expenditure for infrastructure is eliminated. Innovative services are thus able to enter the market quicker and adopt faster to new challenges and user demand. One of the main aspects of this on-demand nature is the concept of elasticity, i.e., the ability of autonomously provision and de-provision resources by reacting to changes in the incoming load. An elastic service is able to operate with an optimal cost by expanding and contracting its used resources at runtime and according to demand. This does not only minimizes running cost, but also avoids disruptive outages due to spikes in service usage. While the various layers comprising a cloud service can be scaled, this does not happen in a unified manner. The vision of CELAR is to provide a fully integrated software stack that manages resource allocation for cloud applications in an autonomous, efficient and generic manner. In order to achieve that, CELAR incorporates novel methodologies for describing cloud applications, monitoring the use of various resources, evaluating cost, taking informed decisions and interacting with the underlying cloud infrastructure. Our goal is two-fold. On the one hand is developing the methodologies for achieving multi-grained, automatic elasticity control on both application and infrastructure level. On the other hand is developing the open-source tools that implement those methods in an integrated manner. Hereby we present an overview of the *CELAR* platform, explaining its architectural components and some basic workflows that show how they interact in order to achieve the core functionalities.

***Keywords*—elasticity; application deployment; decision making**

I. Introduction

Cloud computing embodies the vision of handling computing resources as a utility service [1]. This "pay-as-you-go" model represents a major step in the IT industry that is moving away from privately owned and managed infrastructure. The business model of IT companies is swifted towards creating more innovative applications, capable of tackling the challenges of large scale in the Big Data era [2]. A Cloud Service developer is offered access to a shared pool of configurable computing resources that can be utilized on-demand [3]. This tackles the challenge of creating the computing infrastructure hosting a service, when one enters the market, and maintaining it afterwards. For new and innovative applications the traditional model of investing for a privately owned infrastructure is inefficient. This is not only because of the related acquisition cost, but also because of the time and effort needed to actually handle such an infrastructure.

On the contrary, the opportunity is presented to save cost and utilize no more than the sufficient amount of resources required to meet the application goals, by leveraging the on-demand nature of the resources supplied in the Cloud. One can avoid overspending for a service that does not meet the expected user engagement as well as being unable to handle a service that gets popular quickly. In order to help alleviate those problems, Cloud Infrastructure (IaaS) Providers offer :

- Virtually infinite computing resources being available when needed. Cloud users (i.e hosted applications) need not predict the maximum amount of needed resources beforehand. They can allocate new resources on-the-fly.
- The possibility of short term allocation of computing resources, in an hour-level granularity. Cloud users taking advantage of this short term allocation can make fine grain adjustments to the resources they are using, thus achieving a cost efficiency over time.
- A wide variety of resource types and capacities. Be it, processing, storage, network or other, the offered units are virtualized. As a result, the client has the ability to choose the exact size of resource required for any given task, further increasing cost efficiency.

Those features are the building blocks for applications, in order to be "elastic" in their use of cloud resources.

According to the cloud computing definition [3], being elastic means being able to autonomously adapt to workload changes and manage the available resources so that they match the current demand as closely as possible. Achieving this property is crucial in not only cost optimization by not over-provisioning resources, but also in avoiding service outages, when demand peaks, by not under-provisioning. Such cases [4] can prove to be very disruptive and costly for a client-facing service. Most cloud services are built on a layered architecture where on the bottom layer lies a distributed database system like Cassandra [5] and/or a file system modeled after GFS [6] (most notably, HDFS [7]). Other layers may include a web server cluster, and data processing frameworks like Hadoop [7]. Each one of those components can scale on demand. Research [8] suggests that achieving automation in scaling even the most complex of the building blocks of a Cloud Application -namely NoSQL databases- is feasible. However there is no available middleware that integrates automated cost-aware control with multi-layer resource provisioning. Towards this direction, the CELAR vision is the following:

- Research and develop the methodologies needed to achieve an autonomic, multi-grained, multi-layered resource management platform.
- Develop a complete, open-source software stack that efficiently implements elastic control of cloud applications in order to achieve user-set goals.

For this, CELAR incorporates various subsystems handling different parts of the application workflow. CELAR monitors the usage or resources and the performance of the application's components. It can take informed, precise decisions and orchestrate them both in the infrastructure and the application level. It can also profile the application beforehand in order to observe its behavior under various scenarios. The project also includes a novel Application Management Platform handling the tasks of describing the cloud service on a high level, submitting it for deployment and informing the user of its current and historic state.

In this work we present CELAR's modular architecture, we briefly describe its components and we outline some basic workflows that depict their interaction with the application and underlying infrastructure.

II. System Architecture

In Figure 1 we provide the architecture of CELAR. The system is deployed within an IaaS cloud provider, into a dedicated virtual machine (VM) and its major components are depicted in the figure: the CELAR Server and the CELAR Application Orchestrator.

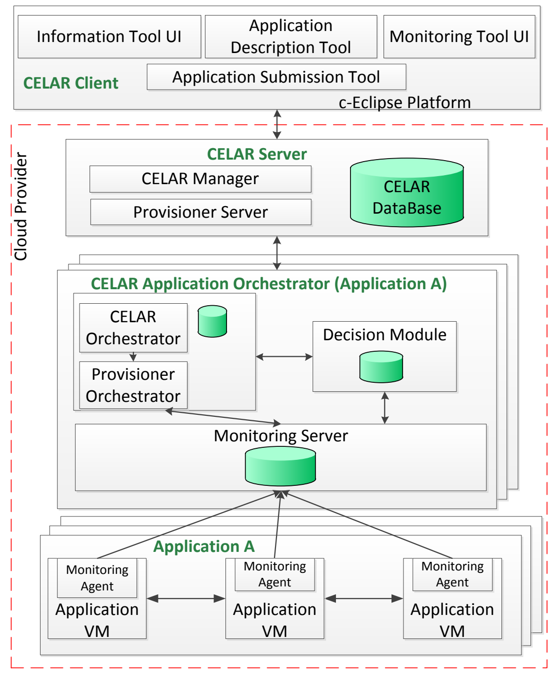

Fig. 1. CELAR System Architecture

The CELAR Server acts as an endpoint between the platform and other services. Specifically, the services executed on the CELAR Server enable users to interact with the platform through a REST API, issue new application descriptions, deploy applications and obtain statistics regarding their applications. The CELAR Manager operates as the endpoint; the Provisioner Server on the other hand is responsible for the deployment and the enforcement of the resizing actions to an application instance. The CELAR DataBase is the main repository of the system: a highly available database in which all the deployment related information is stored. There exists a unique CELAR Server instance within a cloud provider: every time a user describes and deploys a new application, one needs to issue the request to the appropriate CELAR Server instance (physically located inside the cloud they want to deploy their application to) and manage their deployment through it.

Every time a new application is deployed, CELAR creates a new VM in which all the core modules of the platform, responsible for the provisioning and the orchestration of the application, are executed. In Figure 1 this VM is referred to as the CELAR Application Orchestrator. The modules located into the CELAR Application Orchestrator are the following:

- The CELAR Orchestrator along with the Provisioner Orchestrator, which are responsible for the enforcement of the resizing actions, both in resource (i.e., allocate/de-allocate resources) and in application level (i.e., execute the necessary scripts so that the application utilizes the newly allocated resources).
- The Decision Module, which is responsible to evaluate the current status of the application and take elastic decisions in order to improve its performance with respect to a user defined policy.
- The Monitoring Server, which is responsible to report application metrics. This is achieved using agents injected inside the application VMs during their initialization. The agents report the metrics for each VM separately and the Monitoring Server then aggregates the metrics and forwards them to the Decision Module.

Each component exports its functionality to the rest of the modules through a REST API; This enables the platform to properly function with different subsystems, as long as they respect this API. For example, a user might prefer a specific monitoring system to monitor their application or a specific cloud deployment tool. By default, CELAR uses SlipStream [9] for the application orchestration, JCatasopia [10] as a monitoring system and rSybl [11] as the decision making module. All of the above are open source projects, under the Apache License, and they are freely distributed through github.

Finally, to enable the users to describe and deploy applications with CELAR, c-Eclipse [12], a plugin to the popular Eclipse IDE has been used. The users describe their application with c-Eclipse, using the TOSCA [13] specification language and manage their deployments. For further deployment monitoring and statistics, the Information System module provides real-time information on the performed resizing actions, historical data representing former states of the application and other high-level statistics such as deployment costs, performance data, etc.

III. System Workflows

In this Section, we present the functionality of the CELAR platform by describing the major workflows that are involved throughout the lifetime of a managed cloud application.

A. *Application Description*

The first step for a user to utilize the CELAR platform is the description of her cloud application. During this phase, the user interacts with c-Eclipse [12], a user friendly, informative eclipse plugin that serves as a front-end GUI for CELAR. Cloud application portability is of great importance in the realm of cloud infrastructure provisioning. Therefore, c-Eclipse and the CELAR platform utilize the open, non-proprietary OASIS TOSCA specification [13] for describing the resource provisioning, deployment and re-contextualization of cloud applications. This ensures the portability of application descriptions across different IaaS platforms and application deployment tools. In addition, CELAR enables the user to specify application level elasticity actions that can be used to modify resources as well as elasticity policies that need to be enforced throughout the deployment lifetime.

B. Application Profiling

To automatically decide on the best deployment configuration, a system needs to obtain knowledge about the behavior over different resource provisioning and load scenarios. This knowledge can be generated both from on-line learning, during deployment and resizing, as well as from offline profiling that automatically deploys and monitors the application using different deployment and load configurations. The offline profiling process can help the Decision Module have a steeper learning curve and avoid initial errors (and thus financial costs) in scaling decisions. Profiling is an optional functionality provided by the CELAR platform that can be executed before a new deployment. During profiling, a number of different configurations are selected, deployed and monitored in order to identify the relationship between a specific configuration and the application's behaviour. All monitoring metrics are measured and evaluated in order for CELAR to generate a knowledge base that can be used to facilitate the Decision making process. The main challenge for the Profiling module is to intelligently choose the set of profiled configurations. Each deployment has a respective cost both in time and money in order to be sufficiently profiled. Therefore, the Profiler attempts to tackle the problem of generating the most accurate profile using a user specified budget.

C. Elastic Application Deployment

After the user application is described and optionally profiled it can be deployed using one of the available IaaS platforms that are connected with the CELAR platform. The user can issue a deployment request using the c-Eclipse GUI. The request is sent to the CELAR Server accompanied by an initial resource configuration along with elasticity policies that need to be enforced by the Decision Module throughout the deployment lifetime. Elasticity policies are expressed using the user defined metrics and can express complex rules that depend on high level application metrics like cost, performance, availability e.t.c..

As previously stated, when the CELAR Server receives a request for a new deployment, it allocates a new CELAR Orchestrator VM that hosts all the CELAR related components and is responsible for the deployment, orchestration, monitoring and elastic scaling. When the application is deployed monitoring metrics are gathered by our Monitoring system and are utilized by the Decision module in order to decide on the elasticity actions required. Elasticity actions are enforced by the CELAR and the Provisioner orchestrators. The effect of resizing actions on monitoring metrics is stored in the CELAR Database and is used by the Decision Module in order for CELAR to achieve a fine tuned elasticity control. The user can monitor her application using our Information System web interface that graphically depicts monitoring metrics, configuration changes and decisions taken. Using the Information System interface the user can also compare deployments in different cloud providers and decide on the most beneficial according to her requirements.

IV. Conclusion

In this paper we presented *CELAR*, a platform able to automatically scale applications deployed over a cloud infrastructure. CELAR includes tools for describing a cloud service, deploying it on an IaaS provider, profiling it, monitoring its performance, scaling its resources and informing the user of its progress. CELAR will allow Cloud developers to build services that can be efficient and achieve high performance by *elastically* managing the use of available resources.

V. Acknowledgments

This work was partially supported by the European Commission in terms of the CELAR 317790 FP7 project (FP7-ICT-2011-8). Nikolaos Papailiou has received funding from IKY fellowships of excellence for postgraduate studies in Greece - SIEMENS program.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big Data: The Next Frontier for Innovation, Competition, and Productivity," 2011.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," 2011.

[4] "Amazon gets b́lack eyeŕom cloud outage," http://www.computerworld.com/article/2507903/cloud-computing/amazon-gets–black-eye–from-cloud-outage.html.

[5] A. Lakshman and P. Malik, "Cassandra: a Decentralized Structured Storage System," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[7] D. Borthakur, "The Hadoop Distributed File System: Architecture and Design," *Hadoop Project Website*, vol. 11, p. 21, 2007.

[8] D. Tsoumakos, I. Konstantinou, C. Boumpouka, S. Sioutas, and N. Koziris, "Automated, Elastic Resource Provisioning for NoSQL Clusters Using Tiramola," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 34–41.

[9] "SixSq. SlipStream," http://sixsq.com/products/slipstream.html.

[10] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "JCatascopia: Monitoring Elastically Adaptive Applications in the Cloud," in *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2014.

[11] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "Multi-level Elasticity Control of Cloud Services," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds., 2013, vol. 8274.

[12] C. Sofokleous, N. Loulloudes, D. Trihinas, G. Pallis, and M. D. Dikaiakos, "c-Eclipse: An Open-Source Management Framework for Cloud Applications," in *Euro-Par 2014 Parallel Processing*. Springer International Publishing, 2014, pp. 38–49.

[13] "OASIS: TOSCA Version 1.0." https://www.oasis-open.org/committees/tosca/.