ConPaaS: an Integrated Runtime Environment for Elastic Cloud Applications

Guillaume Pierre, Ismail El Helw, Corina Stratan, Ana Oprescu, Thilo Kielmann
Vrije Universiteit Amsterdam
{gpierre,ielhelw,cstratan,amo,kielmann}@cs.vu.nl

Thorsten Schütt, Jan Stender
Zuse Intitut Berlin
{schuett,stender}@zib.de

Matej Artač, Aleš Černivec
XLAB
{matej.artac,ales.cernivec}@xlab.si

1. INTRODUCTION

Most Cloud applications are re-enactments of traditional enterprise applications such as Web applications, content delivery and e-commerce [1]. The advantages of the Cloud are well-known: access to a near-infinite number of resources, ability to adjust an application's capacity on demand, pay-as-you-go pricing model. However, Cloud application developers need to pay attention to new topics: building custom VM images, making applications elastic and scalable, controlling performance, fault-tolerance, etc.

ConPaaS is an open-source platform-as-a-service middleware which aims at simplifying the deployment of applications in the Cloud. In ConPaaS, an application is defined as a composition of one or more *services*. Each service is an elastic component dedicated to the hosting of a particular type of functionality. A service can be seen as a standalone component of a distributed application. Examples of services include a Web hosting service supporting PHP and Servlets, a MySQL database service, and a MapReduce service.

Each ConPaaS service is self-managed and elastic: it can deploy itself on the Cloud, monitor its own performance, and increase or decrease its processing capacity by dynamically (de-)provisioning instances of itself in the Cloud. Services are designed to be composable: an application can for example use a Web hosting service, a database service to store the internal application state, a file storage service to store access logs, and a MapReduce service to periodically compute statistics from these logs. Application providers simply need to submit a manifest file describing the structure of their application and its performance requirements.

2. ARCHITECTURE

Application providers interact with ConPaaS using a Web GUI (Figure 1) or a command-line interface. The interface allows users to create and delete services, upload application code and data to them, monitor their health and performance, and control their resource usage by changing the number of virtual machines used by each service.

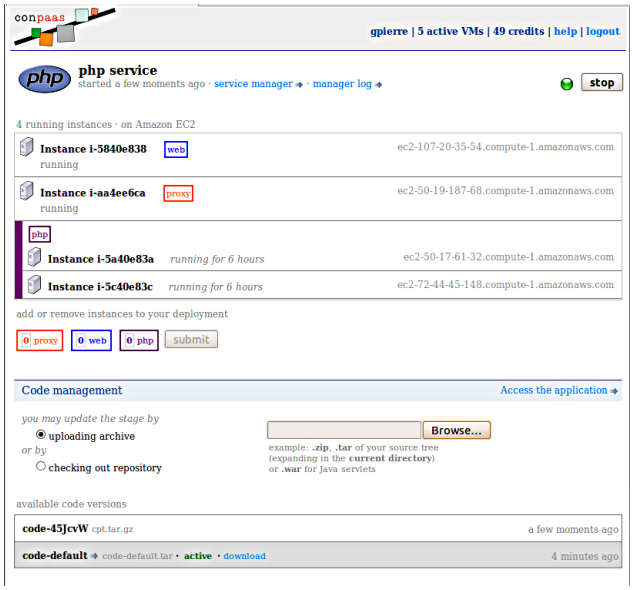

Figure 1: ConPaaS dashboard

As shown in Figure 2, each service is under the control of one "manager" VM. This VM does not run the application itself. It is in charge of executing all management requests, centralizing health and performance monitoring data, and controlling the allocation of resources assigned to this service. The actual application traffic is not addressed to the manager VM. Requests from end users willing to access the application must be directed directly to the worker VMs.

When a service manager receives a request to change the number of VMs assigned to the service, it is in charge of requesting or releasing VMs from the underlying Cloud infrastructure, uploading the service code or data to the new VMs, and coordinating the reconfiguration of other VMs to take the change into account. Reconfigurations are carefully organized such that the application remains available during the entire reconfiguration process. Service managers also make sure to keep the first worker VM active at all times so that applications remain bound to a stable address, even after multiple reconfigurations.

Middleware Posters'2011, December 12th, 2011, Lisbon, Portugal.

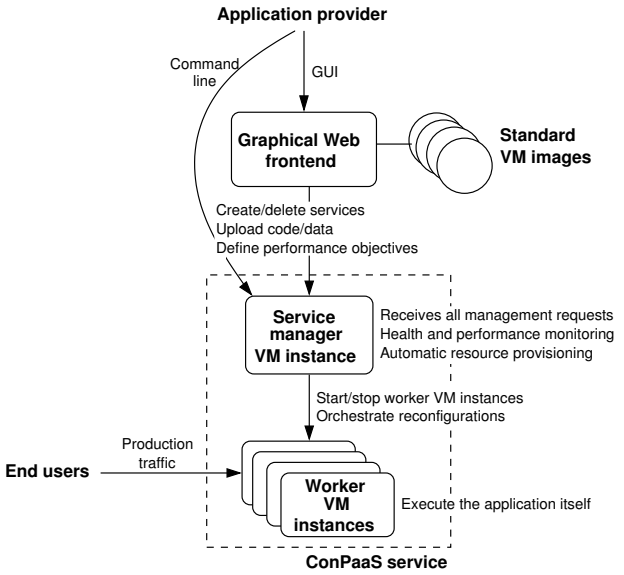

Figure 2: Organization of one ConPaaS service

Depending on the nature of the service, worker VMs can be fully symmetric or further specialized into a number of roles. For example, all nodes of a NoSQL data storage service are identical, and can process any request indifferently. Conversely, a Web hosting service is decomposed into a load balancer (running in the first VM acting as the external contact address of the service), and any number of Web servers serving static content and application servers processing dynamic document generation requests.

3. RESEARCH CHALLENGES

The work on ConPaaS makes us address a number of difficult research challenges.

3.1 Complex application composition

A ConPaaS application can be composed of several independent services. Deploying it automatically in the Cloud means that a different set of machines is used every time. However, each part of the application must be able to invoke other components at a well-known address. Rather than updating the client-side addresses at runtime, ConPaaS dynamically establishes and maintains a VPN between all virtual machines involved in a composed application. Each ConPaaS service can thus choose a local DNS name which gets resolved within the VPN. The VPN is dynamically updated each time a machine instance is added or removed from the application. Besides facilitating service composition, this organization also helps securing the network traffic internal to the application.

3.2 Performance guarantees

A Cloud environment allows application programmers to design elastic applications capable of dynamically varying their processing capacity to follow workload variations. For example, a Web application may add resources when load is high, and later release some of these resources to save costs. However, in a composed application a new issue arises: which individual element should be (de-)provisioned such that performance is guaranteed at minimum cost?

Current Cloud platforms (Amazon's AutoScale, RedHat's OpenShift etc.) allow one to give a performance objective to each service (web server tier, database tier, etc.), and provision each service individually. However, a better approach consists of giving a performance objective only to the front-end services, and making the other components *negotiate* the most efficient use of resources so that the global performance is maintained at minimum cost [2]. ConPaaS will exploit its global knowledge of each application's service composition to guarantee performance using as few resources as possible.

3.3 Cloud heterogeneity

The performance of virtual instances provided by current Clouds is largely heterogeneous, even when requesting the exact same type of instance each time [3]. This has a strong impact on resource provisioning strategies: each time ConPaaS requests a new virtual instance from the Cloud, it must measure the individual performance profile of this particular instance before being able to decide what is the most efficient use of this resource.

3.4 Extensibility

ConPaaS comprises six services: (i) Web hosting; (ii) MapReduce; (iii) Bags-of-Tasks [4]; (iv) MySQL; (v) Scalaris, a strongly consistent NoSQL database [5]; (vi) XtreemFS, a large-scale distributed file system [6]. Although these services are general enough to support a wide range of applications, we aim to make the platform easily extensible to new types of services such as workflows, data streaming, etc.

4. CONCLUSION

ConPaaS is a new open-source Platform-as-a-Service environment which aims at facilitating the deployment, administration and performance control of applications in the Cloud. A first version is available at `http://www.conpaas.eu/`. An open testbed is also available so potential new users can test the technology without having any software to install.

5. ACKNOWLEDGEMENTS

This work is partially funded by the FP7 Programme of the European Commission in the context of the Contrail project under Grant Agreement FP7-ICT-257438.

6. REFERENCES

[1] Amazon Web Services, "Case studies." `http://aws.amazon.com/solutions/case-studies/`.

[2] Jiang D., G. Pierre and Chi C.-H., "Autonomous resource provisioning for multi-service web applications," in *Proc. WWW*, 2010.

[3] Jiang D., G. Pierre and Chi C.-H., "EC2 performance analysis for resource provisioning of service-oriented applications," in *Proc. NFPSLAM-SOC*, 2009.

[4] A.-M. Oprescu and Thilo Kielmann, "Bag-of-Tasks Scheduling under Budget Constraints," in *Proc. CloudCom*, 2010.

[5] F. Schintke, A. Reinefeld, S. Haridi and T. Schütt, "Enhanced paxos commit for transactions on DHTs," in *Proc. CCGrid*, 2010.

[6] F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti and E. Cesario, "XtreemFS – a case for object-based file systems in grids," *CCPE* 20(8), 2008.