PaaS Federation Analysis for Seamless Creation and Migration of Cloud Applications

Daniel A. Rodríguez Silva, Lilian Adkinson-Orellana, Dolores M. Núñez-Taboada and F. Javier González-Castaño
Gradiant, Network and Applications Area, Ed. Citexvi, Campus Vigo, 36310 Vigo, Spain

Keywords: Cloud Federation, Platform as a Service, Modular Programming, Provider Lock-in, Interoperability.

Abstract: A major concern regarding the development of cloud applications is provider lock-in. There are no standard solutions to this problem with the exception of some attempts of achieving interoperability at the infrastructure level. This paper presents an analysis focused on the federation at PaaS level. Furthermore, the proposed schemes are intended to simplify the design of cloud applications by means of a declarative programming based on a set of modules available in all supported platforms. Hence, designed applications can migrate between platforms easily, enabling the creation of new fault-tolerant architectures.

1 INTRODUCTION

The Platform as a Service (PaaS) paradigm offers a simple way of building cloud applications, hiding the complexity of the underlying infrastructure (Lawton, 2008). However, when a developer selects a platform, its applications usually get tied to the underlying technologies, hindering the migration to other PaaS (Louridas, 2010). Google App Engine, Microsoft Azure or Heroku are popular platforms, highly heterogeneous in terms of supported languages (Python, Java, .NET...) and programming environments (NetBeans IDE, Eclipse IDE, Visual Studio...), which illustrates the strong dependence inherent to the adoption of a particular solution. Moreover, the complexity of transferring large amounts of data between cloud providers makes migration between platforms particularly difficult. Therefore, users have no option but accepting unexpected changes on the conditions of the services (Bradshaw, Millard and Walden, 2010).

For this reason, an abstraction layer over different PaaS, which will allow easy development and seamless deployment of applications in any of them, is of great interest. PaaS federation minimizes the problems aforementioned, as an intermediate layer that guarantees interoperability between PaaS providers. In this paper we introduce this concept. Section 2 reviews related work. Section 3 analyzes the features of the federated PaaS and presents three different approaches for its architecture. Finally, section 4 concludes the paper.

2 RELATED WORK

Even though the federation of cloud resources is an extended concept at IaaS level (Rochwerger, Breitgand, Levy, Galis and Nagin, 2009), it does not have a clear equivalent at the platform level. CumuLogic (http://www.cumulogic.com) is a service that allows creating customized PaaS for Java applications. However, the migration of its developed applications to other existing PaaS is complex. There are other open source solutions like Cloud Foundry (http://www.cloudfoundry.com) and the mOSAIC FP7 project (Di Martino, Petcu, Cossu, Goncalves, and Máhr, 2011) that provide platforms deployable in several public or private clouds. Nevertheless, they focus on federation at IaaS level. In (Paraiso, Haderer, Merle, Rouvoy and Seinturier, 2012) a federated multi-cloud PaaS infrastructure is presented. However, it considers some infrastructure services for managing both the multi-cloud PaaS and its SaaS applications, so it also relies on the IaaS layer and depends on Java. In (Gonçalves, Cunha, Neves, Sousa and Barraca, 2012) present a cloud service broker that allows managing applications through a PaaS and an IaaS manager. These managers support multi-provider and multi-cloud

Rodriguez-Silva D., Adkinson-Orellana L., Nuñez-Taboada D. and Javier González-Castaño F..
PaaS Federation Analysis for Seamless Creation and Migration of Cloud Applications.
DOI: 10.5220/0004407201560159
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 156-159
ISBN: 978-989-8565-52-5

environments. Nevertheless, the framework does not make the selection of implementation technologies transparent, limiting the target PaaS.

3 PaaS FEDERATION ANALYSIS

3.1 Federated PaaS Requirements

A federated PaaS is a high level platform that supports a set of traditional PaaS to provide an abstraction layer simplifying the creation and deployment of cloud applications. In order to build a federated PaaS it is important to consider some requisites and functionalities.

3.1.1 Ease of Application Design

The development of applications in the federated PaaS is reduced to a simple process of selecting and configuring a set of predefined modules available in all the underlying platforms. Modules should be created beforehand and mapped to the corresponding implementations in each supported PaaS. Following the modular declarative programming approach, application development becomes a simple choice of the appropriate modules to obtain the desired functionality at lower effort. Figure 1 illustrates the concept proposed. Modules are represented by boxes connected through the different cloud layers. Each module can work autonomously and has a well-defined interface to facilitate interoperability.

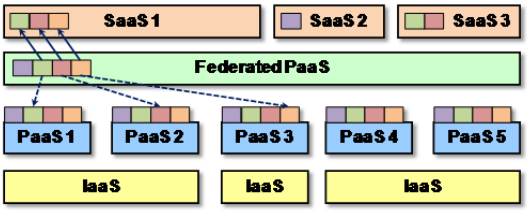

Figure 1: General architecture of the proposed system.

The federated PaaS will also provide mechanisms to add modules to its repository according to the needs of the SaaS developers.

3.1.2 Monitoring

The federated PaaS will include a monitoring module to verify that the quality parameters of interest stay within the established thresholds. The measurable parameters are standard ones: input and output bandwidth (GB), consumed RAM memory (MB), consumed CPU time (CPU hours) and stored data (GB/month). This data will allow billing based on average resource consumption during a time window, besides the verification of SLA fulfilment.

3.1.3 Efficient Resource Provisioning

Regarding resource provisioning, the main goal is the creation of an intelligent delivery system capable of determining the most appropriate PaaS fitting application requirements and SLAs. This process, transparent to developers, will minimize costs by migrating applications to other platforms when their performance improves. However, developers can choose the PaaS to deploy applications manually. The federated PaaS will hide the complexity of dealing with underlying PaaS. Another interesting feature is the possibility for an application of using modules from different PaaS, in order to obtain the best price or performance combining all of them.

3.1.4 Security and Privacy

The federated PaaS should provide mechanisms to build secure applications, a secure communication with the underlying PaaS, an adequate authentication mechanism and protection for sensitive information, among other. Moreover, applications developed on the federated PaaS are intrinsically fault-tolerant: in case of PaaS failure, applications will automatically migrate to another PaaS satisfying their constraints.

3.2 Federated PaaS Approaches

The first proposed approach to build a federated PaaS is shown in Figure 2. In the design phase developers can create applications by choosing the components from a catalogue and connecting them. The result is a configuration file describing how the application must be built. This file is processed by a component (PaaS selector), in charge of choosing the most suitable PaaS attending to SLA constraints and the status of the underlying PaaS. Once the target PaaS is chosen, the builder component creates the application combining the appropriate modules from the repository and deploying them on the selected platform. Hence, each application created in the federated PaaS will have a unique equivalence with an application deployed on the target PaaS. Besides, all modules belonging to an application will always be deployed on the same platform. The main benefit of this model is that it offers the possibility of easily exploiting the monitoring and accounting capabilities of each underlying PaaS. As all the modules of an application are deployed on the same platform, it is straightforward to obtain these values.

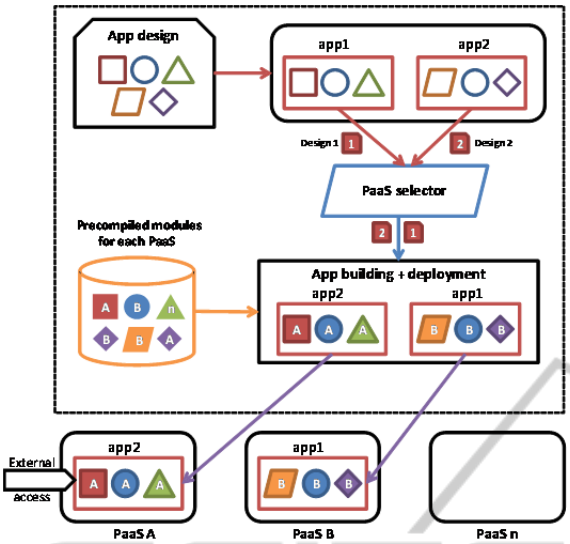

Figure 2: Model A.

However, this design has the drawback that, as each deployment on a PaaS needs to be registered, it is necessary to automatize the registration of applications in each platform.

The second approach (model B) is shown in Figure 3. In an initial stage all the predefined modules are deployed in each supported PaaS. Therefore, when a new application is created there is no real deployment, but only instantiation and configuration of the corresponding modules.

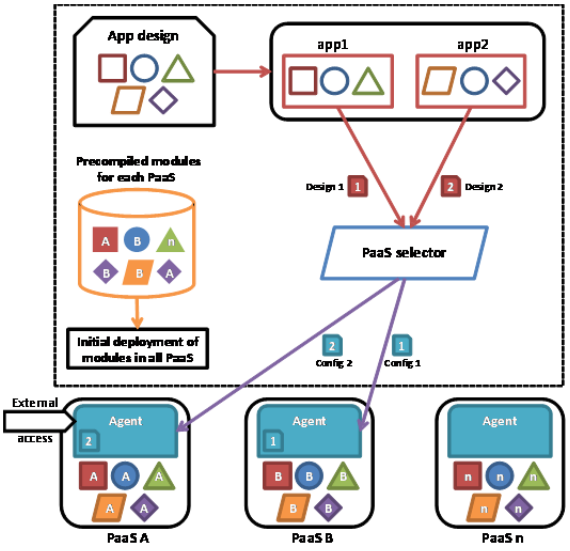

Figure 3: Model B.

To support this scheme, there is an agent in each PaaS in charge of orchestrating the modules according to the configuration file of the design phase. This agent receives the specific rules from the PaaS selector to assemble the modules dynamically and acts as the entry point for external access.

The main advantage of this approach is that deployment time is almost null. In addition, it is possible to combine modules deployed in different platforms to compose a federated PaaS application, which allows developers to benefit from the most suitable option in each case. The main disadvantage is that, as all the modules are shared by all the developers, it is necessary to develop a specific federated PaaS monitoring system to evaluate and register the resources consumed by each application in each shared module.

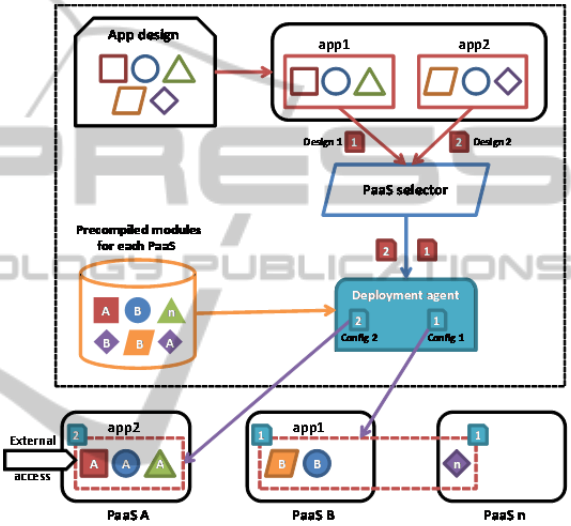

Figure 4: Model C.

The third approach (Figure 4) is a hybrid solution from model A and model B. The configuration file generated in the design phase is processed by a deployment agent in charge of deploying all the required modules in the corresponding underlying PaaS. Each module is configured before being deployed following the federated PaaS application rules and has logic inside for interacting with other modules depending on its configuration. Thus, the modules of each application in the federated PaaS are not shared and several modules of the same type can be deployed in the same PaaS for different applications. Moreover, the same application can use several modules from different PaaS.

With this model, the federated PaaS can exploit the monitoring capabilities of each PaaS. However, since the modules are deployed on-demand, it is necessary to register and deploy them programmatically.

3.3 Discussion

The three models presented have different characteristics depending on how the federated PaaS interact with the different underlying PaaS. Table 1 compares four important features to be considered, because there is not always direct control over public PaaS supported by the federated PaaS

Table 1: Analysed models comparison.

	Model		
Feature	A	B	C
Need of dynamical application registration	Yes	No	Yes
Monitoring capability exploited	Yes	No	Yes
Applications can combine modules from different PaaS	No	Yes	Yes
Instant deployment of applications	No	Yes	No

Public PaaS usually provide an API to access specific functionalities programmatically. However, in some cases the registration of new applications must be manual, via a web interface. For that reason model B is the only valid option when dynamic registration is not available. On the other hand, public PaaS provide mechanisms to monitor resource consumption in order to verify their billing. This feature can be exploited in models A and C, but model B requires implementing a specific monitoring system.

Regarding combination of modules deployed in different PaaS, model A is the only one that cannot support this, because applications are specifically built for the PaaS where they are going to be deployed. Combining modules from different PaaS in one application brings flexibility, for example when some module is only available for one PaaS. However this can be dangerous in the sense the final application cannot always migrate to another PaaS. Finally, model B is the fastest option to deploy applications as modules are only instantiated and configured.

To sum up, the option that has less dependency with regard to the underlying PaaS is model B, despite a monitoring system has to be implemented. If all supported PaaS have a complete API to manage applications, model C is the most flexible option.

4 CONCLUSIONS

Due to the variety of current PaaS solutions, it is not easy to unify criteria for the development of applications, so further work is needed to map the functionalities of the provided modules –at federated PaaS level– for each supported platform. The current state of the art indicates that this problem is still not solved. Application development independently of the underlying PaaS technologies mitigates the problem of provider lock-in, which hinders the migration to other PaaS and, as a consequence, provides fault tolerance across different vendors. The three architectures analysed give an idea of the difficulties involved in federation at PaaS level. Developers must pay the price of lower freedom for creating applications, since they will have to rely on a set of predefined modules extendable on demand.

ACKNOWLEDGEMENTS

This research has been supported by the CloudMeUp grant (IDI-20101357), funded by CDTI, Spain.

REFERENCES

Bradshaw, S., Millard, C. and Walden, I. (2010). Contracts for clouds: A comparative analysis of terms and conditions for cloud computing services. In *Queen Mary School of Law Legal Studies Research (Paper No. 63/201), London.*

Di Martino, B., Petcu, D., Cossu, R.. Goncalves, P., Máhr, T. and Loichate, M. (2011). Building a Mosaic of Clouds. In *Euro-Par 2010 Parallel Processing Workshops, Lecture Notes in Computer Science, vol. 6586/2011, pp.571-578.*

Gonçalves, C., Cunha, D., Neves, P., Sousa, P. and Barraca, J. (2012). Towards a Cloud Service Broker for the Meta-Cloud. In *CRC 2012, Construction Research Congress*, West Lafayette, IN, U.S.A.

Lawton, G. (2008). Developing Software Online with Platform-as-a-Service Technology. *Computer, vol. 41, no. 6, pp. 13-15.*

Louridas, P. (2010). Up in the Air: Moving Your Applications to the Cloud. *Software, IEEE, p. 6-11.*

Paraiso, F., Haderer N., Merle P., Rouvoy R. and Seinturier L., (2012). A Federated Multi-Cloud PaaS Infrastructure. In *CLOUD 2012, 5th IEEE International Conference on Cloud Computing,* Honolulu, Hawaii, EEUU.

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente I. et al. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development, vol. 53, no. 4.*