Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 97 (2016) 63 – 72

Procedia Computer Science

CLOUD FORWARD: From Distributed to Complete Computing, CF2016, 18-20 October 2016, Madrid, Spain

Multi-Cloud Platform-as-a-Service Model, Functionalities and Approaches

Ana Juan Ferrer[a,]*, David García Pérez[a], Román Sosa González[b]

[a]*ATOS, Av. Diagonal 200, Barcelona 08019, Spain*
[b]*ATOS, Subida al Mayorazgo, 24B, Planta 1*
38110 Tenerife, Spain

Abstract

Platform-as-a-Service (PaaS) is the Cloud area concentrating the major growth in the last years in terms of adoption and market share. At the same time Cloud models are evolving towards Hybrid Clouds through the consideration of service expansion across a diversity of Cloud offerings. So far, the majority of multi-cloud approaches have explored IaaS layer, neglecting the rest of the Cloud stack. This limits enterprises to enjoy the benefits of an agile, automated and adaptable infrastructure though flexibility provided by PaaS multi-cloud. This paper presents and compares results of two research projects addressing PaaS multi-cloud architectures: ASCETiC and SeaClouds.

Peer-review under responsibility of organizing committee of the international conference on cloud forward: From Distributed to Complete Computing

Keywords: PaaS; Multi-Cloud computing; Platform-as-a-Service; Multi-Cloud PaaS; ASCETiC Project; SeaClouds Project

1. Introduction

Cloud computing first emerged as Infrastructure-as-a-Service (IaaS) boosted by the significance acquired by Amazon Web Services[1]. In parallel, Salesforce.com was offering Software-as-a-Service (SaaS) - based on the concepts of application service provision (ASP), Grid and Utility computing dating back to the 70's - an offering

* Corresponding author.
E-mail address: ana.juanf@atos.net

Peer-review under responsibility of organizing committee of the international conference on cloud forward: From Distributed to Complete Computing
doi:10.1016/j.procs.2016.08.281

that also included a customization layer, force.com. Soon, driven by the existence of force.com and the eruption of this market with the entrance of Google's App Engine, it became clear the existence of a middleware layer in between IaaS and SaaS: Platform-as-a-Service (PaaS). PaaS enables simplified consumption of Cloud infrastructure and sustains Cloud applications. PaaS is nowadays the Cloud area showing the most impressive growth, according to figures recently published by Gartner[2].

At the same time, overall Cloud market is transitioning from experimental per use-case computing alternative to a general and conventional IT strategy adopted both by commercial enterprises and public bodies. While private Cloud is now in the path to become mainstream, market is now evolving to shift its focus on the next step - hybrid multi-cloud computing models - searching for the right balance among functionality, flexibility and investment protection. A Multi hybrid cloud model allows organizations to provide, use, and manage IT resources across their private cloud set-ups and any compatible public cloud.

In order for these hybrid multi-cloud approaches to develop further and cope with expectations, the cloud has to become an element of a multi-faced approach to service delivery within an enterprise's broader digital infrastructure, heading towards a truly hybrid strategy and tackling all Cloud layers. The digital infrastructure of the future has to provide with a variety of service delivery venues where users will be able to schedule and automate the delivery of application to the most suitable clouds (private/public) depending on application specific characteristics, SLAs and policies. In this context, so far the majority of multi-cloud approaches have explored IaaS layer, neglecting the rest of the Cloud stack, and not considering automation and orchestration brought by PaaS. This limits enterprises to transform their IT and enjoy the benefits of an agile, automated and adaptable IT infrastructure though flexibility provided by PaaS Multi-Cloud.

In this paper we present the results of two research projects addressing from different approaches PaaS Multi-Cloud architectures: ASCETiC[21] and SeaClouds[25]. We elaborate on the building blocks and desired capabilities for a multi-cloud PaaS and compare both projects' architectures with identified required features.

2. Platform-as-a-Service Concept and Status

NIST[3] defines Platform-as-as-Service as "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment". Consumers of PaaS services are two main user groups: Enterprises with their own internal software development activities; and ISVs interested in selling SaaS services on top of a hosted PaaS.

Since its initial emergence, PaaS layer has undertaken a major shift, from mainly accounting of only two large players as Microsoft (Azure) and Google (App Engine), to the wide variety of capabilities and programming languages found in PaaS offerings today. These new entrants' products range from specialized niche segments to solutions for multiple programming languages. Often they do not even offer their own execution environment, but they rely on IaaS providers for this. So they become a proxy for the consumption of IaaS services.

Now, in PaaS there is the curious situation of IaaS vendors pushing up the stack to offer an added value by enabling programming frameworks on top of their infrastructure in order to overcome the risk of becoming a commodity. Meanwhile, SaaS vendors offer platform tools to tailor their on-demand portfolio with the intention of creating customer loyalty and establishing a wider market place around their offering. All together providing a truly diverse array of capabilities tagged as PaaS offerings. A PaaS cloud provides a container platform where users deploy and run their components. Diversities are present in supported programming tools (languages, frameworks, runtime environments and databases), in the various types of underlying infrastructure, and even on capabilities available for each PaaS.

However, most known platforms are Azure[4] and Google App Engine[5]. These two platforms add to the execution environment a set of tools in order to facilitate development on top of the platforms. Google App Engine enables developers to build applications locally (on developer machines) and deploy these to the cloud, on the same environments that powers Google applications. It offers fast development and deployment and simple administration. The App Engine platform provides an execution environment where applications run on a virtualized

technology foundation that scales automatically on demand. Google App Engine is often criticized for not providing transparency to the user to control infrastructure and how this infrastructure is used. Developers do not have direct control over resource allocation, because the underlying system and hardware resources are masked by the App Engine layer. Azure Platform is a PaaS for applications with specific focus on the .NET framework- It is a part of Microsoft's Cloud computing strategy, along with their software as a service offering. The platform consists of various on-demand services hosted in Microsoft data centres and commoditized through three product brands. These are: Windows Azure, an operating system providing scalable compute and storage facilities; SQL Azure, a Cloud-based, scale-out version of SQL Server, and Windows Azure AppFabric, a collection of services supporting applications both in the Cloud and on premise. Additionally, existing PaaS platforms, such as Cloud Foundry automatize the application deployment to a set of template VMs, with complete and isolated platform stack both considering private Cloud set-ups and also public Cloud offerings based in the platform such as Pivotal and Atos Canopy's Cloud.
In addition it can also be noted that PaaS offerings today fit at least into one of the following classifications [14,15] :

Table 1. PaaS Classification.

Classification	Description	Examples
SaaS with extensions	Customize and extend the capabilities of a SaaS application	*Salesforce's Force.com*
Purpose-built PaaS	A framework that simplifies the development of a specific class of applications	*Azure*
PaaS tied to a application paradigm	Provides general capabilities, but supports only one programming model or development/deployment environment	*CloudBees, OpenShift, Google AppEngine*
PaaS tied to a IaaS Cloud	May provide general capabilities, but can be used only in the context of a determined IaaS Clouds being a single public Cloud or a single type of private Cloud infrastructure	*CloudFoundry, AWS Elastic Beanstalk*

3. Multi-Cloud Concept and Status

Multi-Cloud is characterised by "serial or simultaneous use of services from diverse providers to execute an application" [6]. Other authors use the term Inter-Cloud for describing different modes of inter-connection across clouds, however following the same approach [7]. A number of scenarios which demonstrate simultaneous usage of diverse Cloud that constitute hybrid heterogeneous private and public clouds have been defined in [8].

- Cloud bursting is most common and simpler hybrid/multi-cloud model. In this model an application that is executing in a private cloud "bursts" into a public cloud when the demand for computing capacity spikes.
- The federated cloud scenario is characterised by a cloud provider at sub-contracts capacity from other providers as well as offer spare capacity to a federation of cloud providers.
- In a multi-provider scenario, the user, or a broker acting on behalf of the user, manages the multi-cloud provisioning of the services. Access to this functionality can be provided either directly or thought by a cloud marketplace in order to hide management complexity.

In these scenarios, so far the majority of the work has concentrated in the IaaS layer. While significant examples are found today in contexts like eScience or Future Internet PPP about existing cloud federations, such as EGI[9] and FI-Lab[10]. These existing examples are characterised by avoiding the majority of associated interoperability issues by using homogeneous technology environments.
PaaS interoperability was the focus for Cloud4SOA[11] solution, a scalable approach to semantically interconnect heterogeneous PaaS offerings across different Cloud providers sharing the same technology. More recently, soCloud[12] has defined a service-oriented component-based PaaS enabling to manage portability, provisioning, elasticity and high availability across diverse clouds. [13] presents a federated multicloud PaaS infrastructure that considers the specific needs of migrating services across clouds, as well as, to manage distributed applications that span multiple multi-cloud PaaS. Its development considers infrastructure services for managing both our multi-cloud PaaS and the SaaS applications.

4. Multi-Cloud PaaS Desired Capabilities and Building Blocks

Going beyond PaaS offerings today, the aim in this paper is to describe a Multi-Cloud PaaS, understood as a open, flexible and interoperable solution that simplifies the process of developing, deploying, integrating, and overall, managing applications executing across in public and private Clouds. In order to achieve this, next sections elaborate on Multi-Cloud PaaS envisaged model and innovative capabilities.

4.1. Multi-Cloud PaaS envisaged model

In order to describe envisaged characteristics of multi-cloud for PaaS, the model proposed by [20] will be used. This model defines and puts together the four core elements of Cloud from a provider's perspective:

- Multi-tenancy is a key factor to make the most of the PaaS Cloud environment: by being multi-tenant a platform achieves economies of scale, reduces cost per user and enables to share continuous improvements along with a wider community. However, it also adds additional requirements for isolation into its architectural design where the execution of an application should not have any impact on others with regards to security, performance, availability and administration levels.
- Cloud Reach represents the elasticity factor, commonly the capability to scale. Not only in the traditional sense of more infrastructure capacity, but also: The capability to support multiple platforms and; The possibility to stretch across Clouds and connect environments
- Service delivery includes service management, monitoring and provisioning, pay-as-you-go pricing and billing capabilities.
- Functional scope represents the traditional software platform capabilities present in any development platform (not Cloud specific).

4.2. Multi-Cloud PaaS Desired Capabilities

For each one of the core elements of the Multi-Cloud PaaS, features and desired capabilities are detailed. It has to be noted that although some of the existing PaaS in the market today offer partial coverage of the capabilities described, none of them offers the overall desired functionality.

4.2.1. Multi-tenancy

Execution isolation: In the context of Cloud computing, multi-tenancy is a Cloud's design for sharing the computing resources that are in use among the different concurrent users. Isolation is the capability of perceiving one shared environment as dedicated and safe. Complete isolation among applications executed in PaaS environments can be achieved using multiple strategies[19]. Among them the following approaches are identified: Virtual Multi-tenancy: This approach simply relies on the isolation provided by resource virtualization (VMs) and hypervisors in the infrastructure management layer. Recently these approaches have evolved to use of Container technologies, although not yet widespread; Organic Multi-tenancy: This approach is based on isolation achieved at different PaaS component's levels, such as application servers, DBMS…

Security at multiple levels: While Cloud computing offers a paradigm shifting technological solution for computational resources and software, the concerns about privacy and confidentiality of data still are a major concern for adoption. It is required capabilities for underlying (data) security and resilience of resources delivered in the PaaS and IaaS multiple enable users to uptake of the Cloud-based delivery model.

Compliance: Public and hybrid Cloud scenarios are characterized by a constant flow of data which cannot be allocated to a particular place. This brings uncertainty regarding the various data protection legislations, which transcend national borders and therefore complicate the compliance with the Data Protection legislations worldwide. Enterprises or individuals using the PaaS to develop applications that handle confidential and private data need to safeguard its privacy. Therefore, from a legal point of view, providing mechanisms to enable data protection and privacy in Cloud environments should be a basic functionality.

4.2.2. Cloud Reach

Self-service: This capability requires of the complete automation of provisioning, configuration and administration of resources in order to bridge the gap between development of an application and its operation. First, it has to include capabilities to deploy and promote application source code, software and artifacts along all the application life-cycle (considering different environments i.e. development, testing and production). After this, it has to offer capabilities for automated deploying and configuration of applications binaries, as well as, application infrastructure components, such as OS, Application containers, DBMS, Load Balancers etc. Finally virtual service provision, based on hypervisor specific VM instantiation from templates done by technology agnostic means has to be provided. In any case, for multi-cloud deployments (considering multiple IaaS providers), due to the current lack of interoperability among VM formats, there is the need of virtual image transformation processes.

Transparency (Independence) and Full control on the underlying infrastructure: With regards to management of the underlying infrastructure, ideally PaaS offerings will require transparency in order to enable the user to have full control over underlying infrastructure. This will be achieved though compete automation and self-service managing the complexity of the management of the underlying infrastructure, while allowing visibility and control over applications execution, for advanced users.

Application Elasticity: Elasticity is an essential core feature of PaaS' application execution. It defines the ability of the underlying infrastructure to adapt to application's demand, potentially considering both functional and non-functional requirements (i.e number of concurrent users, application response time, number of open DB connections, etc). Elasticity is the ability to increase and decrease resources, while scalability considers resource addition, although commonly they are used as synonyms.

Two types of scalability are often considered: horizontal and vertical scalability: Horizontal scalability refers to the amount of instances to satisfy e.g. changing amount of requests, and; Vertical scalability refers to the size of the instances themselves and thus implicit to the amount of resources required maintaining the size. Cloud elasticity required capability has to involve both vertical and horizontal elasticity and rapid up- and down-scaling.

Interoperability and Portability: One of the main barriers to Cloud Computing adoption is interoperability and portability across providers and products. In the scope of PaaS two approaches to interoperability are taken:

- Interoperability and portability of applications among PaaS environments: This is the ability to migrate applications among PaaS offerings.
- Interoperability and portability of applications among PaaS execution environments (IaaS): A PaaS deployment can interact using the same API and the same application with several IaaS Clouds.

Placement Optimization: Best venue execution selection: This capability provides important benefits, since the user is not tighten to a single provider, but it can decide the best choice to deploy each service among the different Cloud offers, based on the type of service to deploy and different application execution requirements. Examples of these can be: level of trust in the provider, based on previous application execution experiences, eco-efficiency, risk, cost, security levels offered, legal constraints or quality of service parameters (e.g. availability, performance, operation, etc.).

4.2.3. Service Delivery

Service Level Agreement (SLA) Management: SLA Management is a key aspect of providing a commercially viable PaaS offering. Importantly, there are two sides to the SLA management in this context. The first side is SLA among the PaaS provider and the PaaS user. The second side is established among the PaaS provider and different providers it may use for application execution. Functionalities that support users to manage both SLAs are required for Multi-Cloud PaaS approaches.

Application execution monitoring and auditing: It refers to the capacity of PaaS users to assess and validate Service Level Agreements and actual Quality of Service being offered by providers during application execution, as well as, to get insights about PaaS and IaaS providers' internal procedures. Aspects such as regulatory and security controls (incident inventory, handling and corrective actions, disaster recovery plans and business continuity, etc.) and data location traceability, are crucial for users to overcome the loss of control inherent to use a Cloud platform.

Consideration of multiple pricing models: Pricing is a critical factor for Cloud providers, as it affects customer behavior, loyalty to a provider, and can determine organization success organization's success. Typical pricing model in Cloud computing is based in pay-per-use. However several refinements can be considered into this schema: per VM, per hour, per hour of CPU time. These that refer to IaaS characteristics are often applied also to PaaS offerings. In addition other pricing models, as subscription ones are starting to emerge considering usage per month. However, more evolved pricing models considering the diversity of SLA Cloud terms have yet to reach the market, and would be necessary for realization of Multi-Cloud PaaS.

4.2.4. Functional scope

Programming models and application architectures: It is always stressed that essential characteristics of Cloud infrastructures are inherent support for elastic, scalable, and resource-friendly application execution. However, these characteristics are not acquired independently of application architecture and platform; Multi-Cloud PaaS has to support full exploitation of Cloud benefits and characteristics.

Integration: It refers to the capability to integrate with legacy software and on-premises assets (i.e. licensed software), as well as, with third-party Cloud services (Cloud Orchestration). A part of considering new services development, Multi-Cloud PaaS has to enable the adaptation and combination of legacy- and licensed-software.

Application Data management capabilities: Most PaaS today offer data management services use with only limited scalability MySQL, PostgresSQL, CouchDB or other open source data management solutions that are easy to integrate with the platform's backend, but that provide very limited scalability and functionality for users. Other approaches rely on integrating their PaaS offering with Data-as-a-Service offerings such as AWS S3, SimpleDB or Microsoft Azure. These have to be extended in Multi-Cloud PaaS approaches.

5. Multi-Cloud at PaaS level

In this section we present both ASCETiC and SeaClouds architectures and approaches to multi-cloud, concluding with a comparative analysis of both projects with regards to core elements and innovative features identified in previous section.

5.1. Multi-Cloud at PaaS level: the ASCETiC application manager approach

The ASCETiC project focused on providing novel methods and tools to support software developers aiming to optimize energy efficiency and minimize the carbon footprint resulting from designing, developing, deploying, and running software in Clouds. It tries to cover the three typical Clouds stacks, from SaaS to IaaS passing by PaaS layer. In this section we are going to focus on explaining the multi-cloud approach at ASCETiC PaaS level.

Application description

The PaaS layer is employing Open Virtualization Format (OVF) specification[22] to define a complete set of virtual machines (VMs) and its relation to be deployed at an IaaS provider. Employing the extensibility available in the OVF standard, new fields were added to reflect several necessities:

- SLA negotiation terms such as: power expend by a VM, maximum cost or vertical elasticity limits.
- Specification in self-adaptation rules to be applied when a SLA term is about to violated.

Although, OVF was selected because it is quite standard and supported by several virtualization providers, the project has also started to study the usage of TOSCA [24]. No implementation has been yet released supporting this format in ASCETiC. In the Figure 1 we can observe the ASCETiC PaaS architecture block diagram. This diagram will be used to explain in detail in the following paragraphs the workflow of ASCETiC at PaaS level.

The user of ASCETiC will submit their application, composed of images plus the OVF description using a REST interface exposed by the Application Manager (AM). Once this OVF and images are received, the AM checks their validity and starts preparing a series of SLA templates.

For each provider stored in the Provider Registry (PR), the AM will create a template for the SLA Manager (SLAM) to negotiate with each IaaS provider to which the ASCETiC PaaS is connected. The templates together with the OVF document are them submit via the AM to the SLAM.

The SLAM will contact them to its equivalent counterpart in the IaaS provider. It will get an offer for this specific application deployment per IaaS and it will rank back the offers and return them to the AM. At the same time, the price offered back by the IaaS provider is updated with a new price by the PaaS Price Modeller, before presenting those negotiation results to the user.

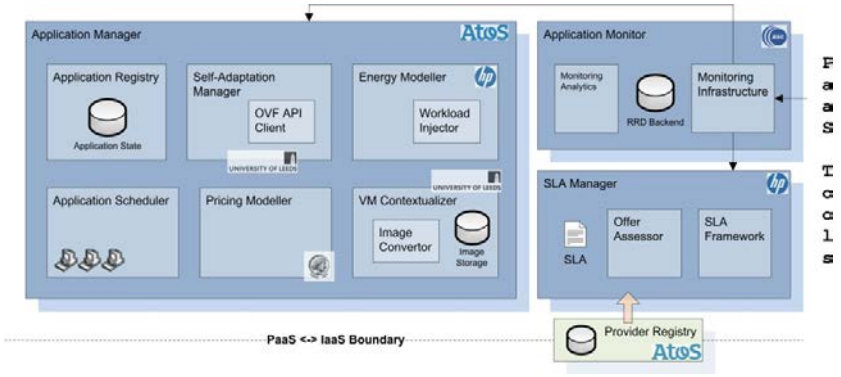

Figure 1 ASCETiC PaaS Architecture

The AM can perform two actions at this stage. If the user specified that s/he wanted to perform manual negotiation, then the AM will show all the offers for her/him to select one. If the automatic flag was enabled when submitting the application, the AM will select the best offer by the ranking selected by the SLAM (based on cost).

The AM will them move the application to the contextualization phase. The VM Contextualizer module will prepare the images to be deployed in the selected IaaS provider. It is expected that different IaaS provider have different contextualization procedures.

The application is now to be deployed by the AM. The Application Scheduler will contact the IaaS provider and start creating the different VM instances as required by the initial deployment plan specified in the OVF document. The application deployment details for each of the phases are exposed to the user by a REST interface that the user can pool to know exactly the status its deployment is. Also, an AMPQ interface for the user to subscribe and get notifications with the same information is available.

Once the application is running, the user has the same functionality as typical Cloud application managers. It can delete the deployment, add more VMs to it, remove VMs, modify the properties of a VM, and so on. The main differences start with the management of the application with performance, energy or costs terms.

The SLAM module is actively monitoring that the application is running inside the negotiated parameters, both from the side of the IaaS provider, and taking into account application behavior. If there is some violation of those parameters, the SLAM notifies the Self-Adaptation Manager (SAM) about it. The SAM will fire some of the rules previously specified in the OVF document to try to get the application inside the SLA defined boundaries.

- At the same time the user is offered a set of functionalities in terms of energy and application performance monitoring:
- At any time the user can register an event in the application execution. An event is something coherent that happens in the application frequently.
- The user can store any kind of metric in the App Monitor. The App Monitor offers both a REST and AMPQ API for the user to programmatic employ it from its application.

IaaS Requirements: Of course, the ASCETiC AM cannot work without a compatible IaaS. The ASCETiC project is developing at the same time the necessary components such as a compatible SLA Manger at IaaS level to

be installed in the interested IaaS providers to federate with an ASCETiC PaaS. More information about this is available at [21].

Status: All the previous workflows and functionalities are already working in the actual implementation of the AM, which code is openly available at [21]. The ASCETiC project has validated the solution in a triple testbed environment. Each one of those testbeds has a compatible ASCETiC IaaS provider based on OpenStack. Three different types of applications have been used to validate the previous workflows, carefully selected to have quite different requirements and behaviors (from typical elastic Cloud applications to more HPC type ones). The ASCETiC project is continuing to be developed. It is expected that new functionality for this component goes in the direction of IaaS provider migration methods for a running application.

5.2. Multi-Cloud at IaaS and PaaS level: the SeaClouds approach, the PaaS Unified layer

SeaClouds provides a platform that performs a seamless adaptive multi-cloud management of service-based applications, by developing Cloud Service Orchestrators and a set of tools to manage complex applications, thus avoiding the problem of Cloud lock-in. This is achieved by supporting the distribution of modules that compose cloud-based applications over multiple and technologically diverse IaaS or PaaS offerings, by using a unified management API and universal metrics for monitoring and verifying functional and non-functional properties.

SeaClouds relies on Apache Brooklyn[23], a top level Apache project, as its deployment engine. Brooklyn is a framework for modelling, monitoring, and managing applications in IaaS through autonomic blueprints. A Brooklyn blueprint represents an application: the modules' artifacts, the software stack, the configuration of the relationship between modules, autoscaling policies… may be described in the blueprint. Apache Brooklyn is an open source project with an active community and rising popularity. This fact and the features provided by Brooklyn (declarative deployment, broad catalog of supported entities, a complete GUI…) justified the selection of Brooklyn as deployment engine in SeaClouds, in terms of capabilities and maintenance. Currently Brooklyn uses a blueprint that complies with CAMP's syntax and exposes many of the CAMP REST API endpoints. However, a non-official TOSCA extension has been developed, to which SeaClouds collaborated in its implementation.

TOSCA is an OASIS open standard that defines a description of services and applications, including their components, relationships, dependencies, requirements, and capabilities. It can be described as a technology centered on the application, a concept that closely matches the SeaClouds approach. For this reason, SeaClouds has embraced TOSCA, and uses the TOSCA specification to describe the deployment blueprints. The TOSCA blueprint provides the necessary abstraction with regard to the underlying IaaS or PaaS provider. In terms of implementation, the deployment engine uses Apache jclouds[26] in the case of IaaS providers. In the case of PaaS providers, there are a few libraries that accomplish the same tasks as jclouds, but they do not fit the capabilities needed in the project. To solve this issue, SeaClouds built an abstraction layer that facilitates the management of PaaS providers.

This abstraction layer is the PaaS Unified Library[27]. The library provides basic operations for the management of applications in PaaS providers in a homogenous way. It relies on the official Java clients for each platform. The library can be used to deploy applications programming against it in Java code or using the included REST service, which works on top of the library. The supported operations are: deploy, un-deploy, start, stop, scale and bind service. The current supported providers are CloudFoundry v2 based providers (Pivotal, Bluemix, Canopy Cloud Fabric...), OpenShift v2 based providers (OpenShift Online) and Heroku.

The library decouples the login with the provider and the actual operations to be performed, and therefore, it is structured in two main interfaces: PaaS Client. Its job is to login to the provider in order to get a Session Object; PaaS Session. Acts as a façade to the different basic operations stated previously.

SeaClouds monitors and assess the response time, availability and throughput of applications. According to the desired values for these metrics that have been expressed by the developer in the design of a new application, the corresponding monitoring rules and SLA are generated and included in the TOSCA Deployment Application Model. The flow of actions at design and deployment time is as follows:

1. The user expresses the global QoS: response time, availability and throughput.
2. The Planner generates the monitoring rules (to assess QoS) and the SLA agreement (to keep track of violations).
3. The user agrees with the Agreement and starts the deployment.
4. The Deployer deploys the monitoring rules and starts the agreement enforcement.

Once the application is deployed, it is monitored with the Tower 4Clouds platform, developed in MODAClouds project [28]. The main capabilities of Tower 4Clouds are:

- The user defines the QoS constraints, in the form of monitoring rules,that need to be assessed at runtime. These rules are cloud-provider independent. In SeaClouds, the rules are generated according to the desired response time, availability and throughput of an application.
- The Data Collectors that are deployed together with the application send the monitoring data to a central Data Analyzer, according to the installed monitoring rules, which specify what and how resources should be monitored. No reconfiguration is required after scaling or migration activities. For PaaS applications, an application data collector has been implemented, which is able to collect response times and throughput measurements. The SeaClouds platform is in charge of automatically deploying the data collectors and configuring them once the application is deployed.
- The Data Analyzer processes the data gathered by the data collectors, performing aggregations and/or verifying conditions as specified in the monitoring rules. Some predefined actions can be defined in a monitoring rule and executed when a condition is satisfied.

5.3. Comparative analysis

The following table compares and summarizes ASCETiC and SeaClouds approaches for Multi-Cloud PaaS core elements and the identified desired capabilities.

Table 2. Capabilities and MultiCloud PaaS Approach Comparison.

	ASCETIC	**SeaClouds**
Multi-tenancy		
Execution isolation	Virtual Multi-tenancy	Virtual Multi-tenancy and Organic Multi-tenancy
Security at multiple levels	Not considered	Not considered
Compliance	Not considered	Not considered
Cloud Reach		
Self-service	Standards Based (OVF, TOSCA)	Standards Based (TOSCA)
Transparency	Full (IaaS)	Full (IaaS) and Partial PaaS (depending on PaaS provider capabilities)
Application Elasticity	Rules based	Only considered at IaaS level
Interoperability and Portability	Across IaaS supported by jClouds	Across: IaaS supported by jClouds PaaS (Heroku, OpenShift and Cloud Foundry)
Placement Optimization	Supported (based on self-management on energy and price optimization)	Supported based on prize
Service Delivery		
Service Level Agreement (SLA) Management	Supported (Energy, Price and Performance)	Supported (Performance)
Application execution monitoring and auditing	Application Monitoring	Application Monitoring
Consideration of multiple pricing models	Yes	No
Functional scope		
Programming models and application architectures	Yes (Parallel PM) OVF Virtual appliance	TOSCA Composition
Application Data management capabilities	Not supported	Support DB Schemas

6. Conclusions and Next steps

The comparison performed in this paper in terms of approaches for Multi-tenancy, Cloud Reach, Service delivery and Functional Scope has helped to identify potential collaboration opportunities across the ASCETiC and SeaClouds projects with regards to Multi-Cloud PaaS approach. These collaboration opportunities are in the first term related to the adoption of the TOSCA standard and SLA management. In addition, although known, the analysis also reveals areas of potential improvements for both research projects such as Security and Compliance.

References

1. Amazon Web Services, https://aws.amazon.com/
2. Gartner, “Gartner Says Market Leaders Failed to Capitalize on PaaS Growth in 2015”, http://www.gartner.com/newsroom/id/3283217
3. SP 800-145, “A NIST definition of Cloud Computing”, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2011.
4. Microsoft Azure, https://azure.microsoft.com
5. Google App Engine, https://appengine.google.com
6. Petcu, D. (2013). Multi-Cloud: Expectations and Current Approaches. In Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds - MultiCloud ’13 (p. 1). New York, New York, USA: ACM Press. doi:10.1145/2462326.2462328
7. Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected Cloud Computing Environments. ACM Computing Surveys, 47(1), 1–47. doi:10.1145/2593512
8. Ferrer, A. J., Hernández, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., … Sheridan, C. (2012). OPTIMIS: A Holistic Approach to Cloud Service Provisioning. Future Gener. Comput. Syst., 28(1), 66–77. doi:10.1016/j.future.2011.05.022
9. EGI site, https://www.egi.eu/
10. FI-Lab, https://www.fiware.org/lab/
11. Eleni Kamateri et al. 2013. Cloud4SOA: A semantic-interoperability paas solution for multi-cloud platform management and portability. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 8135 LNCS (2013), 64–78. DOI:http://dx.doi.org/10.1007/978-3-642-40651-5_6
12. Fawaz Paraiso, Philippe Merle, and Lionel Seinturier. 2014. soCloud: a service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. Computing 98, 5 (August 2014), 539–565. DOI:http://dx.doi.org/10.1007/s00607-014-0421-x
13. Fawaz Paraiso, Nicolas Haderer, Philippe Merle, Romain Rouvoy, and Lionel Seinturier. 2012. A Federated Multi-cloud PaaS Infrastructure. In 2012 IEEE Fifth International Conference on Cloud Computing. IEEE, 392–399. DOI:http://dx.doi.org/10.1109/CLOUD.2012.79
14. Cloud4SOA project, D1.1 Requirements Analysis, http://www.cloud4soa.eu/sites/default/files/Cloud4SOA%20D1.1%20Requirements%20Analysis.pdf
15. Forrester Blog, Platform-As-A-Service, Chapter 2, http://blogs.forrester.com/john_r_rymer/10-05-11-platform_as_a_service_chapter_2
16. Younggyun Koh; Knauerhase, R.; Brett, P.; Bowman, M.; Zhihua Wen; Pu, C.; , "An Analysis of Performance Interference Effects in Virtual Environments," Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on , vol., no., pp.200-209, 25-27 April 2007
17. Xing Pu; Ling Liu; Yiduo Mei; Sivathanu, S.; Younggyun Koh; Pu, C.; , "Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , vol., no., pp.51-58, 5-10 July 2010
18. Ibrahim, S.; Bingsheng He; Hai Jin; , "Towards Pay-As-You-Consume Cloud Computing," Services Computing (SCC), 2011 IEEE International Conference on , vol., no., pp.370-377, 4-9 July 2011
19. Steve Bobrowski. 2011. Optimal Multitenant Designs for Cloud Apps. In Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD '11). IEEE Computer Society, Washington, DC, USA, 654-659. DOI=10.1109/CLOUD.2011.98 http://dx.doi.org/10.1109/CLOUD.2011.98
20. Redefining Software Platforms - How PaaS changes the game for ISVs, http://ippblog.intuit.com/blog/2009/10/redefining-software-platforms---how-paas-changes-the-game-for-isvs.html, http://developer.intuit.com/uploadedFiles/Developer/MyIDN/Technical_Resources/QBPD/Redefining%20software%20platforms%20Sep09.pdf
21. ASCETiC project, http://www.ascetic-project.eu/
22. OVF, https://www.dmtf.org/standards/ovf
23. Apache Brooklyn, https://brooklyn.apache.org/
24. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA), https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
25. Seaclouds, seaclouds-project.eu
26. Jclouds, https://jclouds.apache.org/
27. PaaS Unified Library, https://github.com/SeaCloudsEU/unified-paas
28. ModaClouds project, http://www.modaclouds.eu/