Integrated Management of IaaS Resources

Fernando Meireles[1,2] and Benedita Malheiro[1,2]

[1] School of Engineering, Polytechnic Institute of Porto, Porto, Portugal
[2] INESC TEC, Porto, Portugal
{fmdms,mbm}@isep.ipp.pt

Abstract. This paper proposes and reports the development of an open source solution for the integrated management of Infrastructure as a Service (IaaS) cloud computing resources, through the use of a common API taxonomy, to incorporate open source and proprietary platforms. This research included two surveys on open source IaaS platforms (OpenNebula, OpenStack and CloudStack) and a proprietary platform (Parallels Automation for Cloud Infrastructure - PACI) as well as on IaaS abstraction solutions (jClouds, Libcloud and Deltacloud), followed by a thorough comparison to determine the best approach. The adopted implementation reuses the Apache Deltacloud open source abstraction framework, which relies on the development of software driver modules to interface with different IaaS platforms, and involved the development of a new Deltacloud driver for PACI. The resulting interoperable solution successfully incorporates OpenNebula, OpenStack (reuses pre-existing drivers) and PACI (includes the developed Deltacloud PACI driver) nodes and provides a Web dashboard and a Representational State Transfer (REST) interface library. The results of the exchanged data payload and time response tests performed are presented and discussed. The conclusions show that open source abstraction tools like Deltacloud allow the modular and integrated management of IaaS platforms (open source and proprietary), introduce relevant time and negligible data overheads and, as a result, can be adopted by Small and Medium-sized Enterprise (SME) cloud providers to circumvent the vendor lock-in problem whenever service response time is not critical.

Keywords: IaaS, Deltacloud PACI Driver, Multiple IaaS Interoperable Management.

1 Introduction

The provisioning of the Infrastructure as a Service (IaaS) concept, initiated by the Elastic Compute Cloud (EC2) [1] as part of Amazon Web Services (AWS) [2], was rapidly adopted by other well-known technology enterprises with large computing resources, that launched their own IaaS platforms. As a result, the Research & Development (R&D) community as well as the involved enterprises concentrated efforts on the development of new IaaS platforms. However, since Cloud Computing was a recent concept, lacking pre-defined standards and a consensual definition, the resulting platforms were highly heterogeneous in terms of

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part II, LNCS 8806, pp. 73–84, 2014.

functionalities, architecture and interface libraries. This diversity hinders the selection of an IaaS platform and, above all, constitutes an obstacle to the interoperability among cloud service providers.

To overcome this problem, this paper proposes and presents an open source solution that promotes the interoperability and standardization between heterogeneous IaaS platforms. This work involved the research, proposal and development of an interoperable open source solution with standard interfaces (both Web and application programming interfaces) for the integrated management of IaaS cloud computing resources based on new as well as existing abstraction libraries or frameworks. The research consisted of two surveys covering existing open source and a proprietary IaaS platforms as well as open source IaaS abstraction solutions.

The approach proposed and adopted, which was supported on the conclusions of the carried surveys, reuses an existing open source abstraction solution - the Apache Deltacloud framework [3]. Deltacloud relies on the development of software driver modules to interface with different IaaS platforms, officially provides and supports drivers to sixteen IaaS platform, including OpenNebula and OpenStack, and allows the development of new provider drivers. The latter functionality was used to develop a new Deltacloud driver for PACI. Furthermore, Deltacloud provides a Web dashboard and REpresentational State Transfer (REST) API interfaces. To evaluate the adopted solution, a test bed integrating OpenNebula, OpenStack and PACI nodes was assembled, deployed and the time response and data payload via the Deltacloud framework and via direct IaaS platform API calls was measured. The Deltacloud framework behaved as expected, *i.e.*, introduced additional delays, but no substantial overheads. The Web and the REST interfaces produced identical results.

The developed interoperable solution for the seamless integration and provision of IaaS resources from PACI, OpenNebula and OpenStack IaaS platforms fulfils the specified requirements, *i.e.*, enables IaaS cloud providers to expand the range of adopted IaaS platforms and offers a Web dashboard and REST API for integrated management. The contributions of this work include the surveys and comparisons made, the selection of the abstraction framework and, last, but not the least, the PACI driver developed.

2 IaaS Platforms

The IaaS platforms surveyed include the OpenNebula [4], OpenStack [5] and CloudStack [6] open source frameworks and the PACI [7] proprietary solution.

OpenNebula is the only European IaaS platform studied. This platform manages virtual resources from public and hybrid clouds. It presents a layered architecture, which enables the centralised management of data-centres, and provides a detailed level of customization. At the top of the stack, it exposes multiple API to communicate with AWS EC2 [1] and the OpenGrid Forum (OGF) Open Cloud Computing Interface (OCCI) solutions [8][9].

OpenStack is a highly dynamic platform, presenting several new functionalities with each software release. However, it is fragmented into multiple software modules (OpenStack projects) with dedicated interface libraries [10]. This fragmentation hardens the installation process, the management of the platform and increases the complexity of the system. On the other hand, it interacts with several third-party applications, uses RESTful interfaces and offers OCCI [11], AWS EC2 [1] and S3 [12] interface libraries.

Apache CloudStack uses a modular architecture for the automation and centralised management of data-centres, which is organized in zones, pods and clusters. It uses a Query API as well as an API translator so that applications written for CloudStack can also run in AWS EC2 [1]. Although the studied version of CloudStack (4.2.1) does not provide official OCCI support, it is available via a third-party contribution [13].

PACI includes various proprietary products to enable the creation, management, monitoring and billing of public or hybrid (if the PACI platform is used) IaaS platforms. It exposes an open interface (RESTful API) to enable the development of third-party applications for the interaction with the system. However, PACI is a platform without software modules to support directly the interaction with other IaaS platforms. This behaviour is common among proprietary solutions in order to generate the user lock-in phenomenon.

Table 1 compares the authentication, hypervisors, management, interfaces, network, storage and governance features of the studied IaaS platforms. The main differences among the open source IaaS platforms are related to the architecture, interface libraries and governance models. This diversity is caused by

Table 1. IaaS platforms comparison

Features	*OpenNebula*	*OpenStack*	*CloudStack*	*PACI*
Author./Authen.	Password, SSH RSA keypair, X509, LDAP	In-memory Key-Value Store, PAM, LDAP, X509	Password, LDAP, SSH RSA keypair	Password, LDAP
Hypervisors	XEN, KVM, VMware vSphere	KVM, LXC, UML, VMWare vSphere, Xen, PowerVM, Hyper-V	VMware vSphere, KVM, Citrix Xen	Parallels hypervisor, KVM
Management	Centralized	Scattered	Centralized	Centralized
Interfaces	XML-RPC API; AWS EC2, OCCI, OCA	RESTful API; AWS EC2, S3, EBS and OCCI	Query API; AWS EC2, OCCI, Plug-in API	RESTful API
Network	Virtual router, Contextualization	Nova-network, Newtron	Virtual router	POA
Storage	Volume Storage	Volume and Object storage (Glance, Swift, Cinder)	Volume Storage	System DB
Governance Model	Benevolent Dictator	Foundation	Technical Meritocracy	Proprietary

the absence of well defined architectural standards for the commoditization of IaaS systems. Every IaaS platform tends to provide distinct functionalities and be compatible with specific third-party services in order to monopolize the market and impose its technologies as standards. OpenStack is a good example of an IaaS platform that tries to monopolize the market. On the other hand, the proprietary IaaS platform PACI has a limited set of features and no interoperable mechanisms to interact with other platforms, which may purposely lead to a vendor lock-in problem. There are also significant differences regarding the type and number of interfaces, the level of customization, the organization of the groups of operations as well as the structure of the request and response messages provided by the four IaaS platform interface libraries. OpenStack and PACI rely on RESTful interfaces, while OpenNebula and CloudStack use natively XML-RPC and Query (RESTlike) interfaces, respectively.

3 Abstraction Solutions

Interface abstraction libraries provide a collection of implementations for the development of middleware systems that abstract the peculiarities of the underlying IaaS platform and offer a standard and unique API for the management of multiple IaaS clouds. Deltacloud [3], jClouds [14] and Libcloud [15] are examples of existing cloud abstraction solutions.

Deltacloud is an open source framework from the Apache Software Foundation [16] that aims to abstract differences between IaaS cloud platform interface libraries. It is written in Ruby and contains a Web dashboard, a group of IaaS provider drivers [17] (including OpenNebula and OpenStack) and multiple API – the Deltacloud RESTful API, the Distributed Management Task Force (DMTF) open standard Cloud Infrastructure Management Interface (CIMI) REST API [18] and the AWS (EC2 [1] and S3 [12]) API. Each driver exposes the list of implemented Ruby collections. These collections describe the abstractions offered by the Deltacloud API [19] and each collection represents an entity in the back-end provider node.

Apache jClouds and *Libcloud* are open source libraries, developed by Apache Software Foundation [16] in Java and Python, that abstract the differences among multiple cloud provider interface libraries. jClouds offers both portable abstractions and cloud-specific features, which enable the management of buckets (BlobStore) and compute operations (ComputeService), and has a list of compatible cloud providers and IaaS platforms, including OpenStack and CloudStack [20]. The Libcloud library supports an extensive group of IaaS platforms [21], including OpenNebula, CloudStack and OpenStack and allows users to manage compute, storage and network cloud resources.

Deltacloud, jClouds and Libcloud are among the most representative cloud IaaS abstraction solutions and are used in several R&D cloud interoperability related projects, *e.g.*, Aeolus and mOSAIC [22][23]. Deltacloud, which provides by default three different service API (native RESTful Deltacloud, CIMI and AWS EC2 API), is a framework that includes a Ruby client, a Web dashboard

and a driver development environment to support the integration of further IaaS platforms. jClouds and Libcloud are standard programming libraries and, unlike Deltacloud, do not integrate additional development tools. In terms of IaaS platform support, Libcloud provides official integration with the studied open source IaaS platforms (OpenNebula, OpenStack and CloudStack), jClouds supports CloudStack and OpenStack while Deltacloud supports OpenNebula and OpenStack. None of these abstraction solutions provides support for PACI. Table 2 presents the comparison between these open source abstraction solutions.

Table 2. Open-source abstraction solutions comparison

Features	*Deltacloud*	*jClouds*	*Libcloud*
Type	Framework	Library	Lybrary
Programming language	Ruby	Java	Python
Supported providers	17 cloud providers	30 cloud providers	38 cloud providers
Supported operations	Compute, Storage, Network	Compute, Storage	Compute, Storage, Network
Platform integration	Drivers	Maven dependencies	Drivers
API	REST, CIMI, AWS		
Other interfaces	Web dashboard, Ruby client		

Although Libcloud provides official support for the analysed open source IaaS platforms, there are also third-party drivers that integrate CloudStack with Deltacloud [24]. Thus, the Deltacloud abstraction framework was adopted because it provides additional development tools and Web services (*e.g.*, the Ruby Command Line Interface and Web Dashboard), exposes broadly used interface libraries (CIMI and AWS EC2) and provides documentation for the development of Deltacloud drivers to integrate new IaaS platforms that can be used for the development of the PACI driver.

4 Interoperable Service Proposal and Development

The Interoperable Service uses the Deltacloud abstraction framework as a middleware between cloud users and IaaS platforms, permitting the management of multiple IaaS platforms via a single service. The architecture of this Interoperable Service is composed by the back-end driver modules (OpenNebula, OpenStack, CloudStack and PACI driver), the software daemon `deltacloudd` and the Graphical User Interface (GUI) and API services. Figure 1 illustrates the architecture of the Interoperable Service.

The back-end driver modules, composed of the OpenNebula, OpenStack, CloudStack and PACI drivers, are integrated and developed to enable the abstraction and interaction with the respective back-end IaaS platforms. These drivers define, through method instantiation and implementation, the Deltacloud operations that the IaaS platform provides. The software daemon `deltacloudd` is included in the deltacloud-core component and is responsible for the start-up and deployment of the front-end interface services (the GUI and API services). The GUI

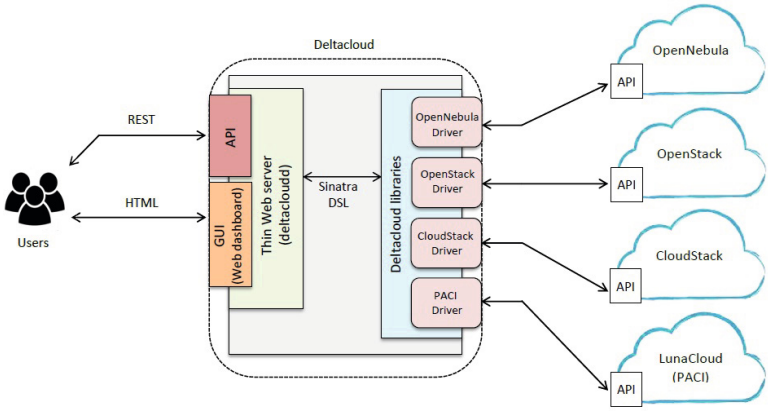

Fig. 1. Interoperable Service architecture

service presents a simple Web dashboard containing the driver implemented collections and operations. The API service has a RESTful implementation that uses the Deltacloud defined collections and operations to expose the cloud resources of the IaaS platforms [19].

Depending on the configuration of the Deltacloud daemon, two different deployments modes can be adopted: (*i*) the single tenant configuration where a single Deltacloud daemon loads a pre-defined YAML file containing the credentials and the cloud provider URL endpoint for each driver module; and (*ii*) the multiple tenant configuration where multiple server instances, containing each the GUI and API services, are defined by individual back-end driver modules, *i.e.*, each Deltacloud server instance contains a specific driver, port and cloud provider endpoint URL to access the respective back-end IaaS platform.

5 Tests and Results

In order to test the developed Interoperable Service, a test bed containing OpenNebula, OpenStack, CloudStack nodes and Internet access to a PACI cloud provider was assembled – Figure 2. This test bed is not intended to test the individual properties and capabilities of each IaaS system.

The OpenNebula, OpenStack, CloudStack and PACI driver modules were tested and evaluated in terms of functionality and interoperability performance using this test bed. The experiments compared the Deltacloud API calls with the direct IaaS platform API calls in terms of response time per operation (*i.e.*, the total amount of time required to perform a HTTP request and obtain the response) as well as the HTTP request packet length and HTTP response content length. The execution of the API operations (via the Deltacloud API and via the IaaS platform API) and the measurement of the corresponding response time was performed with the cURL command line tool [25]. The HTTP request packet

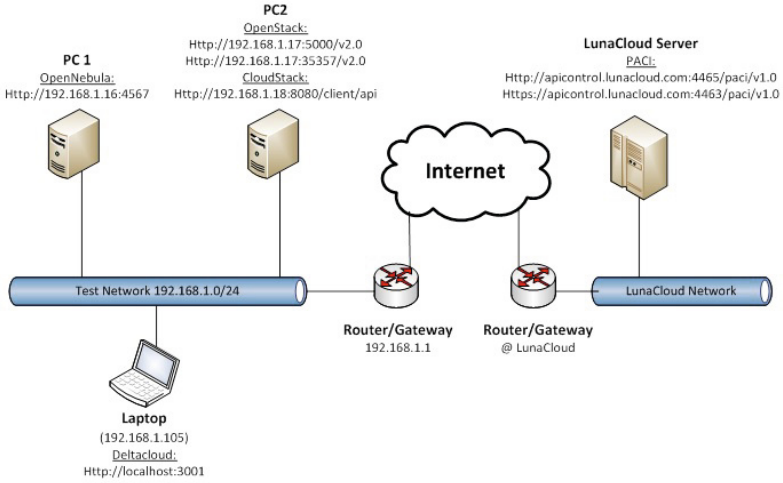

Fig. 2. Test bed platform

length and response payload were measured using the Wireshark software [26]. For the sake of these tests, the HTTP Secure Sockets Layer (SSL) encryption security procedure was purposely discarded.

Problems were detected with the OpenNebula, OpenStack and CloudStack drivers. The OpenNebula driver supplied with the Deltacloud framework had two minor bugs related with an id argument mismatch in the `destroy_image` method (included in the `opennebula_driver.rb` file) and the instantiation of an unused `xmlfile` argument in the delete method of the `occi_client.rb` file. Both problems were corrected and reported. The OpenStack driver, although fully functional, lacked the start and stop VM operations in the OpenStack rubygem. Moreover, the `delete_instance` method was defined as an alias of the `stop_instance` method, causing the destruction of the VM whenever the Stop Instance operation is invoked. The third party CloudStack driver, added to Deltacloud in order to integrate the CloudStack IaaS platform, did not work. From the analysis of the driver implementation, it was possible to conclude that the driver is incomplete and, thus, non functional.

The results obtained for OpenNebula are presented in Table 3. The interaction via the Deltacloud API, which relies on the OpenNebula driver module, increases the operation response time, particularly in the listing operations, *e.g.*, the List Instances, List Images and List Hardware Profiles operations. It is also possible to observe that the Delete Image and Create Instance operations have almost the same average response time.

The HTTP request packet length and returned payload per operation reinforce the interpretation of the response time results from Table 3. The HTTP request packets length of the OpenNebula OCCI API operations are slightly bigger than the ones of the Deltacloud API operations. This can be observed mainly in the

Table 3. OpenNebula results

	Time Response (s)		*Data* (B)			
	OCCI API	*Deltacloud API*	*OCCI API*		*Deltacloud API*	
			Request	*Response*	*Request*	*Response*
List Collections	0.010	0.030	174	348	172	654
List Instances	0.130	0.582	194	6585	181	7376
Show Instance Information	0.089	0.150	183	1219	183	729
Create Instance	0.462	0.482	515	591	298	592
Stop Instance	0.295	0.426	349	1218	189	907
Start Instance	0.203	0.396	348	1219	190	639
Reboot Instance	0.194	0.359	348	1218	191	907
Delete Instance	0.323	0.361	187	0	187	0
List Images	0.105	0.529	194	2828	178	10 966
Show Image Information	0.071	0.133	183	287	180	1101
Delete Image	0.220	0.234	187	0	184	0
List Hardware Profiles	0.012	0.067	200	720	189	1232

Create Instance, Stop Instance, Start Instance and Reboot Instance operations. On the other hand, the length of the HTTP response payload varies and is bigger for the responses of Deltacloud API List Collections, List Instances, List Images, Show Image Information and List Hardware Profiles operations (being the List Instances and List Images the responses containing the larger values), identical in the Create Instance operation and larger for the responses of the OpenNebula OCCI API Show Instance, Stop Instance, Start Instance and Reboot Instance operations. In the case of the Delete Instance and Delete Image operations, the returned payload length is nil since they are silent.

The OpenStack response time as well as the HTTP request packet length and returned payload (using the Deltacloud API and the OpenStack services API) are presented in the Table 4. Since the authentication request is performed in each Deltacloud API operation when using the OpenStack driver, the average HTTP authentication request response time was added to the average of the OpenStack services (Nova and Glance) API operations time response. As expected, the response time of the Deltacloud API operations is significantly higher than the response time of the OpenStack services API (Keystone, Nova

Table 4. OpenStack results

	Time Response (s)		*Data* (B)			
	OpenStack API	*Deltacloud API*	*OpenStack API*		*Deltacloud API*	
			Request	*Response*	*Request*	*Response*
List Instances	0.131	1.196	2605	16 522	157	13 608
Show Instance Information	0.064	0.387	2635	2371	194	1346
Create Instance	0.380	0.848	2748	665	290	931
Reboot Instance	0.198	1.025	2721	0	202	1346
Delete Instance	0.244	0.947	2670	0	200	1346
List Images	0.095	0.508	2604	7050	154	16 547
Show Image Information	0.070	0.445	2634	967	191	1378
Delete Image	0.194	0.412	2669	0	194	0
List Hardware Profiles	0.024	0.216	2598	1356	165	2177

and Glance) operations. This occurs for all listed operations except for the Delete Image and List Hardware Profiles operations. In fact, the average response time of operations like the List Instances, Create Instance and Reboot Instance Deltacloud API operations reached values higher than the operations performed via the Deltacloud API using the OpenNebula driver.

The authentication procedure (authentication token) used by OpenStack is reflected in the values of the HTTP request packet length of the OpenStack API operations, which is substantially bigger than the values of the corresponding Deltacloud API operations. On the other hand, the HTTP response payload varies. The OpenStack API List Instance and Show Instance Information operations return bigger payloads than the corresponding Deltacloud API operations, the Delete Image operation returns the same payload in both cases (a void HTTP response body) and the remaining operations return a smaller payload than the Deltacloud API counterparts. The Reboot Instance and Delete Instance operations are silent. Usually, the HTTP response of the Delete Instance operation defined by the Deltacloud API is also silent. However, since the OpenStack driver defined the `stop_instance` method as an alias of the `destroy_instance` method, the pause of an OpenStack instance with the Deltacloud API deletes the instance. In fact, it sends the Delete Instance operation, but returns the Stop Instance operation result.

Contrary to the open source IaaS platforms (OpenNebula, OpenStack and CloudStack), which were in the same test network as the laptop used to perform the tests, the PACI IaaS platform was in an external network. This way, the latency of the network was taken in consideration in the results presented in Table 5. The analysis of the results shows that, despite the registered latency, the time response values of the PACI API operations are lower than the values registered for the OpenNebula OCCI API and OpenStack API operations, with the exception of the List Images operation. Although, the List Image operation lists 103 Images in comparison with the 10 images that were listed by the same

Table 5. PACI results

	Time Response (s)		*Data* (B)			
	PACI API	*Deltacloud API*	*PACI API*		*Deltacloud API*	
			Request	*Response*	*Request*	*Response*
List Instances	0.032	1.385	180	726	177	10 276
Show Instance Information	0.033	0.124	188	938	185	1128
Create Instance	0.374	0.647	675	165	343	932
Stop Instance	0.078	0.278	193	17	191	939
Start Instance	0.064	0.280	194	18	192	1126
Delete Instance	0.078	0.186	191	19	188	0
List Images	0.655	3.235	186	46 029	174	153 680
Show Image Information	0.042	0.094	206	449	194	1508
List Load Balancers	0.035	2.483	191	818	182	9592
Show Load Balancer Information	0.033	1.717	195	1063	186	947
Create Load Balancer	0.393	0.520	203	167	355	610
Delete Load Balancer	0.075	0.081	198	21	189	0
Associate Instance with LB	0.076	0.463	204	131	284	947
Dissociate Instance from LB	0.081	0.286	206	28	286	610

operations of the OpenNebula OCCI API and OpenStack API. In comparison, the results of the interaction with the Deltacloud API, using the PACI driver, show a significantly response time increase, mainly with the List Instances, List Images, List Load Balancers and Show Load Balancers Information operations. The List Images operation presents the highest time response value, since the driver has to process the information of 103 returned images. On the other hand, the response time results for the remaining operations is justified by the need to perform additional calls to the back-end PACI API and to process the returned information. The refinement of this methodology may improve the measured response time. Other Deltacloud API operations, *e.g.*, Show Instance Information, Stop Instance, Start Instance, Delete Instance and Show Image Information present lower response time than the corresponding operations via the OpenNebula and OpenStack drivers.

The length of the HTTP request packets is larger for the PACI API operations with the exception of the Create Load Balancer, Associate Load Balancer and Dissociate Load Balancer operations. These Deltacloud API operations require more parameters than the corresponding PACI API operations. The Deltacloud API operations return a larger payload than the direct API calls with the exception of the Show Load Balancer Information, Delete Instance and Delete Load Balancer operations. In the case of the last two operations, the Deltacloud API does not send a HTTP response body.

6 Conclusions

In order to propose and develop an interoperable service for the integrated management of cloud resources provisioned by different IaaS platforms, a survey was conducted to compare the features of the most popular open source IaaS platforms - OpenNebula, OpenStack and CloudStack - and of a proprietary IaaS platform - PACI. This survey concluded that, although the open-source IaaS platforms expose similar functionalities, the architecture, interface library operations and governance models are significantly different. The proprietary solution does not support directly the interaction with other IaaS platforms and originates, on purpose, the vendor lock-in problem to monetize new products and paid support services. Additionally, the proprietary IaaS platform has a smaller group of functionalities in comparison with the open source IaaS platforms studied. Regarding the interface libraries, the OpenNebula, OpenStack, CloudStack and PACI client API showed significant differences in terms of type and number of interfaces, level of customization, organization of the groups of operations and structure of the request/response messages.

A second survey on existing IaaS abstraction solutions compared the Deltacloud framework and the jClouds and Libcloud libraries. The result was the selection of the Deltacloud framework since it provides many of the desired functionalities (Web dashboard, multiple API, a Ruby client application), includes several IaaS platform driver modules and integrates new IaaS platforms through the development of new dedicated driver modules.

To evaluate the proposed solution, a test bed was assembled, deployed and used to determine the time response and data payload via the Deltacloud framework and via direct IaaS platform API calls. In terms of driver functionalities, these experiments showed that the new PACI driver was fully functional, the OpenNebula and OpenStack drivers were fully operational after minor corrections and improvements and the CloudStack driver module was incomplete and non functional. In terms of driver performance, the results showed that the use of Deltacloud drivers to access the IaaS platform resources introduces an expected response time delay when compared with the direct platform API calls. In the majority of the operations, the HTTP request packet length of the Deltacloud API was lower and the results of the HTTP response payload were substantially higher in the case of the Deltacloud API listing operations. In general, the Deltacloud abstraction framework reduces the HTTP request and response detail to the essential information. The PACI platform, despite being located at an external network, presented the lowest time response of the tested platforms.

The solution adopted for the integrated management and provision of IaaS resources from PACI, OpenNebula and OpenStack IaaS platforms fulfils the specified requirements, *i.e.*, integrates multiple IaaS platforms and offers a Web dashboard and a REST API for user management. The contributions of this work include the surveys made, the selection of the abstraction framework, the assembled test bed platform and, last, but not the least, the developed PACI driver. Although the PACI driver performed well, it can be refined to enhance the response time of certain operations. Future improvements to the Deltacloud API may also enhance the performance of the included drivers. The PACI driver was shared with the Deltacloud community and the detected OpenNebula and OpenStack driver malfunctions were also reported.

Acknowledgements. This work was partially supported by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP – 01-0124-FEDER-022701».

References

1. Amazon: Amazon Web Services: Amazon EC2 (April 2014), `http://aws.amazon.com/ec2`
2. Amazon: Amazon Web Services (April 2013), `http://aws.amazon.com`
3. Deltacloud: Deltacloud Framework (May 2014), `http://deltacloud.apache.org`
4. C12G Labs: Open–Source Enterprise Cloud Simplified (April 2014), `http://opennebula.org`
5. OpenStack Foundation: Open source software for building private and public clouds (April 2014), `http://www.openstack.org`
6. Apache Software Foundation: Apache CloudStack: Open Source Cloud Computing (April 2014), `http://cloudstack.apache.org`

7. Parallels: Parallels Automation for Cloud Infrastructure (May 2014), http://sp.parallels.com/products/paci
8. Open Grid Forum (OGF): Open Forum - Open Standards (April 2014), https://www.ogf.org/dokuwiki/doku.php
9. OGF: OCCI: About (April 2014), http://occi-wg.org/about/
10. OpenNebula: OpenStack Programs (May 2014), https://wiki.openstack.org/wiki/Programs
11. OpenStack: OCCI (April 2014), https://wiki.openstack.org/wiki/Occi#Summary
12. AWS: Amazon S3 (April 2014), https://aws.amazon.com/s3
13. Isaac Chiang: rOCCI Server – A Ruby OCCI Server (June 2014), https://github.com/isaacchiang/rOCCI-server
14. jClouds: The Java Multi-Cloud Toolkit (May 2014), http://jclouds.apache.org
15. Apache Libcloud: One Interface To Rule Them All (May 2014), http://libcloud.apache.org
16. Apache: The Apache Software Foundation (April 2014), http://www.apache.org/foundation
17. Deltacloud: Deltacloud drivers (May 2014), http://deltacloud.apache.org/drivers.html#drivers
18. DMTF: Cloud Management Initiative (May 2014), http://dmtf.org/standards/cloud
19. Deltacloud: Deltacloud API (May 2014), https://deltacloud.apache.org/rest-api.html#rest
20. jClouds: Providers (May 2014), http://jclouds.apache.org/reference/providers
21. Libcloud: Supported Providers (May 2014), http://libcloud.apache.org/supported_providers.html
22. Aeolus: Manage Your Cloud Deployments with Ease (May 2014), https://github.com/aeolusproject/aeolusproject.github.com/wiki
23. mOSAIC: Open source API and platform for multiple clouds (May 2014), http://www.mosaic-cloud.eu
24. Childers, C.: CloudStack Driver for Deltacloud (May 2014), https://github.com/chipchilders/deltacloud/tree/cloudstack-driver/server/lib/deltacloud/drivers/cloudstack
25. Stenberg, D.: cURL (May 2014), http://curl.haxx.se
26. Wireshark Foundation: Wireshark (May 2014), http://www.wireshark.org