Managing OVF applications under SLA constraints on Contrail Virtual Execution Platform

Yvon Jegou, Piyush Harsh, Roberto G. Cascella, Florian Dudouet and Christine Morin
Inria Rennes - Bretagne Atlantique
France
Email: {yvon.jegou, piyush.harsh, roberto.cascella, florian.dudouet, christine.morin}@inria.fr

***Abstract*—The move of users and organizations to Cloud computing will become possible when they will be able to exploit their own applications, applications and services provided by cloud providers as well as applications from third party providers in a trustful way on different cloud infrastructures. To reach this goal, standard application formats must be enabled on the cloud to avoid vendor lock-in, and guarantees concerning protection, performance and security must be supported. This article describes the Contrail VEP component developed by the Contrail project. The VEP component is in charge of managing the whole life cycle of OVF distributed applications under Service Level Agreement rules on different infrastructure providers.**

I. Introduction

The advent of Cloud computing has changed the way users can define new services on top of this lease infrastructure. This new trend opens many possibilities for offering commercial solutions or simply for deploying applications to respond to a current need of the users. Indeed, the complexity of managing the infrastructure is entrusted to cloud providers, freeing the users who can focus on the application itself. Not all future cloud users will have the skills to develop their own virtual machine based applications and some of them will retrieve applications corresponding to their needs from repositories potentially managed by third-party providers.

These application market share repositories are interesting for the uptake of Cloud computing but they can exist only if applications and their description are not only tied to specific software or hardware of a single cloud provider. As such avoiding a potential vendor lock-in situation is critical and can be solved with the use of open standards for application description to facilitate the move to Cloud computing [1]. Although large organizations can afford to develop their own applications, their need for standards in applications formats is also present as it allows these organization to select the best cloud offerings. On the one hand standards and interoperability issues are one important side of the market expansion of Cloud computing; on the other hand providing specific guarantees under a Service Level Agreement (SLA) is what is needed to make users trust the Cloud for their application.

The *pay-per-use* model of Cloud computing is acceptable if users can have minimal guarantees concerning cloud availability, dependability, security, etc. Such requirements can be expressed in Service Level Agreements representing some form of contract between users, organizations and providers. These SLAs must be automatically enforced, the application events and resource usage monitored, the SLA violation detected and possible counter-measures applied. Moreover, to diagnose violations or application misbehavior, auditing of all details of complex applications must be possible.

These issues are some of challenges that the Contrail European project [2] aims to solve and a major objective is to develop a framework for the management of distributed applications on federated cloud providers under the control of Service Level Agreements. Contrail applications are standard OVF applications and are deployed with strict guarantees specified in SLAs documents. SLAs can be negotiated for a specific application, they can be associated to multiple applications, or only to some OVF terms defining a class. The specification of the Contrail components is in progress and a first release of the main component has been made public in May 2012 [3].

This paper describes the Contrail VEP component in charge of the deployment and management of OVF applications under the constraints of SLAs at the provider level. While interoperability and avoiding vendor lock-in the Cloud are important, this paper strictly considers the issue of providing strict guarantees to users, and leave these other important aspects for discussion in another paper [1]. From the analysis of the application requirements and the needs of Cloud computing users, the paper delves into the issues of making Cloud provider trustworthy by supporting SLA constraints while deploying applications. Then, it identifies the required steps and present an implemented solution to manage distributed application, i.e., the Contrail VEP which is the main contribution of this paper. The remainder of this paper is organized as follows. Section II motivates the need for a new solution to manage cloud providers and discusses the objectives and requirements in the context of the Contrail project. Section III presents the Contrail VEP component in charge of application management at provider level. Section IV discusses related propositions and section V concludes this paper.

II. Objectives

Providing trust is the major challenge in Cloud computing because companies and users should be able to rely on the leased infrastructure for deploying their applications. With the term *trust* herein we mean the reliability of a cloud provider for what concerns performance guarantees in terms of Quality of Service (QoS) and security guarantees to run their applications

in a safe environment; the former are expressed in terms of Quality of Protection (QoP). For instance, the biggest asset for big enterprises is the intellectual property and the knowledge they have in the data they own. They are rightly reluctant in putting and hosting such data in an environment where they do not maintain absolute control or they do not have strict guarantees that the data will be protected from any unauthorized usage. Another constraint comes from the data protection and privacy laws enforced by different governments. Such laws call for strict geo-location restriction of data hosting and movement.

Service Level Agreements (SLAs) between the end users and the providers is a solution proposed in the literature to negotiate QoS and QoP requirements. However concerns remain for enforcing these constraints while deploying an application at the provider level and making the application SLA-aware. Moreover, how does one verify that the agreed SLAs were honored in the first place, and if violated how can such violations be proved for possible claims and settlements?

Another concern is the disparity in cloud APIs provided by different vendors to the end users. Such disparity results in vendor lock-in situations where a user is unable to migrate her cloud deployment over to another cloud provider because of interface incompatibilities between the two.

These are some of the motivations that call for a more standardized way to describe the applications and design a new component for managing different Cloud providers. The goal of this paper is to delve into the issues for deploying and managing distributed applications spanned over heterogeneous providers. We focus on the integration and support of SLA requirements for selecting the best cloud provider and the solutions we put in place within the framework of the Contrail project to handle the deployment of applications in such a constrained execution environment. Details about interoperability issues and solutions to avoid vendor lock-in are discussed in another paper [1].

A. Essential elements of a distributed cloud application

It makes sense to look at Infrastructure-as-a-Service (IaaS) services to get the true picture as other forms of cloud services such as Platform-as-a-Service (PaaS) and Storage-as-a-Service (SaaS) services are value addition on top of IaaS services.

A cloud application to be hosted over IaaS clouds consists of a set of VMs possibly linked with each other in a private LAN with access to/from external Internet through a gateway or a proxy. Therefore the critical elements of an IaaS cloud application are:

- Virtual Machines description;
- Virtual Network elements linking VMs;
- OS image files to run inside the VMs;
- Data Stores to be attached to the VMs.

Apart from the bare-minimum requirements that has been listed above, the user would also require some formalism in the agreement between herself and the provider. These would include such elements as:

- Service Level Agreements (SLAs);
- Placement restrictions and data protection agreements (QoP);
- Performance requirements such as quality of inter-VM communication;
- Dependability criteria such as data center failure;
- Billing and auditing provisions for compliance tests and verification;
- Monitoring and auditing mechanisms for the user to infer the health of the deployed application.

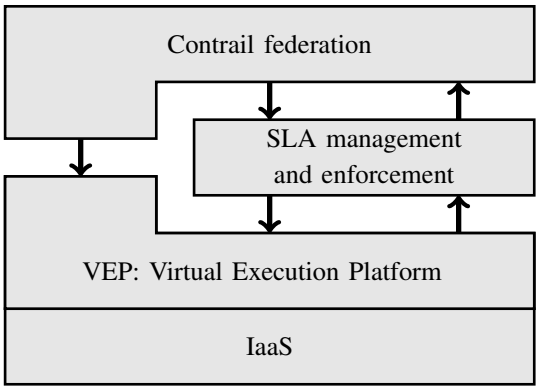

Fig. 1. Overall Contrail architecture

Finally, to fully take advantage of the cloud business, the distributed application must be described in a format accepted by as many providers as possible. To reach this goal, the use of open standards is a key for interoperability. More details about open standards to help building bridges towards the goal of achieving user applications and cloud providers interoperability can be found in [1].

Portable applications also facilitate the exploitation of different forms of cloud federations (cloud brokering, community cloud, cloud bursting) which can take care of negotiating application deployment with multiple providers on behalf of the user. A significant progress has been made for pivotal elements such as storage, infrastructure management, and application description formats, but there still remains much work to be done to reach the final destination.

B. Contrail: Striving Towards Interoperability

The European project Contrail [2] [3] is developing a complete cloud platform which integrates a feature-rich PaaS offering on top of a federated IaaS cloud providers. The Contrail software stack will have an extensive SLA support along with required monitoring mechanisms to enforce and manage the negotiated SLAs between the customers and the providers. Contrail incorporates an extensive set of dedicated security suites to manage the authentication, authorization and other security needs across a cloud federation.

The overall architecture of a Contrail system is described in figure 1.

The Contrail stack is made of three major components: the federation component, the SLA management and enforcement component and the Virtual Execution Platform. The federation

component communicates with multiple providers; to each provider corresponds a SLA management component and a VEP component.

A SLA management and enforcement component is associated to each provider. It is in charge of SLA negotiation with the provider layer and of the enforcement of negotiated SLAs. Negotiated SLAs are communicated to the VEP component in order to be applied. The VEP is in charge of provisioning resources from the IaaS layer in conformance with negotiated SLAs, of monitoring the application during the whole life cycle and of providing monitoring data to the SLA enforcement component. In case a SLA violation is detected from the monitoring data, the SLA enforcement service can react through requests to the VEP layer or can inform the federation about the violation.

The Contrail framework authentication and authorization modules incorporate several widely used protocols such as OAuth and Shibboleth [4] and the attributes repository is designed to be easily extensible in order to provide support for easy incorporation of third party attribute repositories in order to enable easy migration of user accounts and attributes into Contrail.

Each VEP service receives application management requests (deployment, snapshot, elasticity) from the federation service and translates these requests into requests to the provider IaaS service. The VEP provides a uniform interface for distributed application management to the federation layer. It can be integrated to the IaaS platform, or can exploit the IaaS component through its public interface. Next section details the Contrail Virtual Execution Platform functionalities.

III. Contrail Virtual Execution Platform

The VEP service is in charge of managing the whole life cycle of a distributed application on a provider IaaS infrastructure. The VEP receives requests from a Contrail federation manager, but can also be configured to support applications directly supplied by users.

VEP provides a RESTful API to the federation and to users. All operations on VEP RESTful *resources* are subject to authorization from the Contrail authentication and authorization modules (not described in detail in this paper).

The major features of the Contrail VEP are:

- support for SLAs through Constrained Execution Environments;
- support for the DMTF's OVF standard for application description (without any extension);
- support for partial application deployment to allow multi-provider application deployment from the federation layer;
- support for elasticity management of applications;
- support for advance reservation;
- support for application snapshots.

A. Contrail VEP namespaces

Contrail VEP provides means to manage applications as a whole: a single request to VEP can result in the deployment of many resources. Items deployed during the same request may reference each other: it is not possible to reference some resource using its unique ID generated at deployment time as long as this resource is not deployed.

To provide means to identify items before they are deployed, Contrail VEP allows to associate a `name` attribute to any component and to reference this component using this name. References to RESTful resources are defined in the VEP API using the attribute `href`, the default reference being the unique ID allocated by the system at resource creation-time. In Contrail VEP, the unique ID can be replaced by `#resource-name` (`resource-name` being the attribute name of the resource) as the `href` value when the resource is located in the right name-space. To each type of VEP component is associated a name-space: OVF virtual systems are located from the `OVFs` collection of the application, application virtual machines are located from the application `VMs` collection, CEE virtual machine handlers from the `VMs` collection of the CEE containing the application, etc. The VEP specification (not yet available) lists, for each resource type, the collection where the component names are searched.

To avoid dependencies on the order in which VEP components are listed in the documents submitted to VEP, logical names are resolved in two steps: the first step creates all new resources and registers their names; the second step resolves all logical names.

B. Contrail Constrained Execution Environment

Each application submitted to VEP is deployed and run in a Constrained Execution Environment (CEE). A CEE is a virtual infrastructure in charge of executing Contrail applications. For instance, it is possible to define an elastic HPC-cluster CEE with medium sized virtual machines, a shared NFS server and all resources connected through a 10 Gigabits network. The virtual machines are added to the CEE on demand. Multiple applications deployed in the same CEE can share virtual networks and so cooperate. A CEE contains resource allocation handlers and constraints.

Resource handlers define resource configurations for the allocation phase, the volume of physical memory dedicated to a virtual machine on a server or the minimal throughput of a network for instance, and monitoring rules for the exploitation phase. Resource handlers also maintain the list of resources allocated to the CEE and their status (in use or free). Resource handlers can also provide information about the cost associated to resource exploitation. Typical resource handlers of Contrail VEP are:

- virtual machine handlers,
- network handlers,
- shared volume handlers.

Constraints are mainly related to placement restrictions, performance guarantees and security requirements. When a constraint is linked to a resource handler, all virtual resources allocated from this handler must enforce the associated constraint. Some typical placement constraints supported by Contrail:

- `sameHost`: all resource items linked to this constraint must be placed on the same host;
- `sameCluster`: all resource items linked to this constraint must be placed on the same server cluster;
- `notInCountries`: linked resources must not be allocated on the countries listed in the constraint;
- `differentDataCenters`: linked resources must be allocated on different datacenters.

A user can have multiple constrained execution environments, each environment corresponding to a different virtual platform. A CEE is created through the instantiation of a CEE template. Resource providers can publish predefined CEE templates in a CEE repository. In Contrail, CEE templates and instances can be derived from negotiated Service Level Agreements. Some details of the CEE can be made configurable, for instance the destination of the monitoring data. It is possible to integrate an application to a CEE template. The application is automatically started when the CEE is instantiated.

In order to deploy a new application, a user (or the federation) must first select or instantiate a CEE defining the application execution environment. When an OVF application is to be deployed in a CEE, each OVF virtual resource (virtual system, shared disk or network) is handled by a resource handler of the CEE. The rules for associating virtual resources to handlers can be defined in various ways, explicitly in deployment documents or implicitly using default rules (see section III-E). Each handler defines the physical resources to be allocated for each virtual resource mapped on the handler. For instance, a virtual machine handler defines the number of processor cores, the physical memory size, the disk space (and its location) to allocate for each virtual machine. These values can be defined using ranges, in which case, the final value is derived from other means, the virtual hardware specification of an OVF virtual system for instance.

C. Contrail application

Once it is deployed, a Contrail application is made of one or more OVF documents and a list of virtual machines, networks and shared volumes instantiated from the OVF resource items (virtual system, network, shared disk). The OVF structure is maintained during the whole application lifecycle as OVF details can be exploited when extra resources are added to provide elasticity. This OVF structure is also used to export application snapshots in OVF format. An OVF virtual system can be instantiated multiple times. The instantiation process of virtual machines considers an OVF virtual system, a CEE VM handler and configuration parameters. The CEE handler to be used for an OVF virtual system can be specified in the instantiation request, specified by default mapping rules defined for the application or derived from default CEE mapping rules. The details of the physical resources to be allocated are derived from the selected CEE handler and from the OVF virtual hardware specifications. Contradictory requirements can result in deployment errors. A similar process is applied for network and shared volume instantiation.

Contrail also provides support for application templates which can be deployed multiple times and in different CEEs. The structure of an application template is similar to the structure of an instance. The major difference is that the references to CEE handlers are virtual (logical names remain unresolved). These references are replaced by references to effective CEE handlers when the application is instantiated in a CEE.

Contrail supports two major forms of application deployment methods: the direct method and the incremental method. The simplest form of application deployment in VEP is the submission of an OVF file to a CEE interface (direct method). To support this method, the CEE must integrate default mapping rules for OVF virtual resources. Using this method, VEP automatically creates a new application in the CEE and applies all the deployment steps defined in section III-E until the application reaches the `running` state.

The incremental application deployment method involves the following steps:

1) Create an empty application in a CEE; multiple applications can be hosted by the same CEE and share network resources.
2) Submit the OVF file for deployment to the application (`POST` request on the `OVFs` collection of the application using the Contrail RESTful API); multiple OVFs can be composed in a single application.
3) Submit one (or more) deployment documents to the application.
4) Trigger the `start` action of the application.

The application keeps track of deployed OVF items. When a virtual resource terminates, the physical resources are freed but the resource descriptor are retained until explicitly deleted.

Figure 2 shows a CEE defining two virtual machine handlers (`smallVM` and `largeVM`), one shared disk handler (`largeVolume`) and one network handler. An application made of five virtual machines, a shared disk and a network is deployed. All VMs are handled by the `smallVM` handler. Three of these VMs are instantiated from the same OVF virtual system.

D. VEP deployment document

A deployment document contains a list of items to be added to the application: virtual machine, network or shared volume. For each item, this document defines an optional logical name, the reference to an OVF item to be deployed, the reference to a CEE handler to use (optional) and configuration data (a list of key-value pairs). If no CEE handler is specified, default rules defined in the application descriptor or in the CEE are applied to locate a handler. OVF networks and shared disks can be explicitly deployed only once in an application. OVF virtual systems can be deployed multiple times. The configuration data are exploited to associate values to OVF properties.

E. Resource allocation

Contrail VEP gathers all resource requests during the instantiation phase before initiating the allocation phase in order to

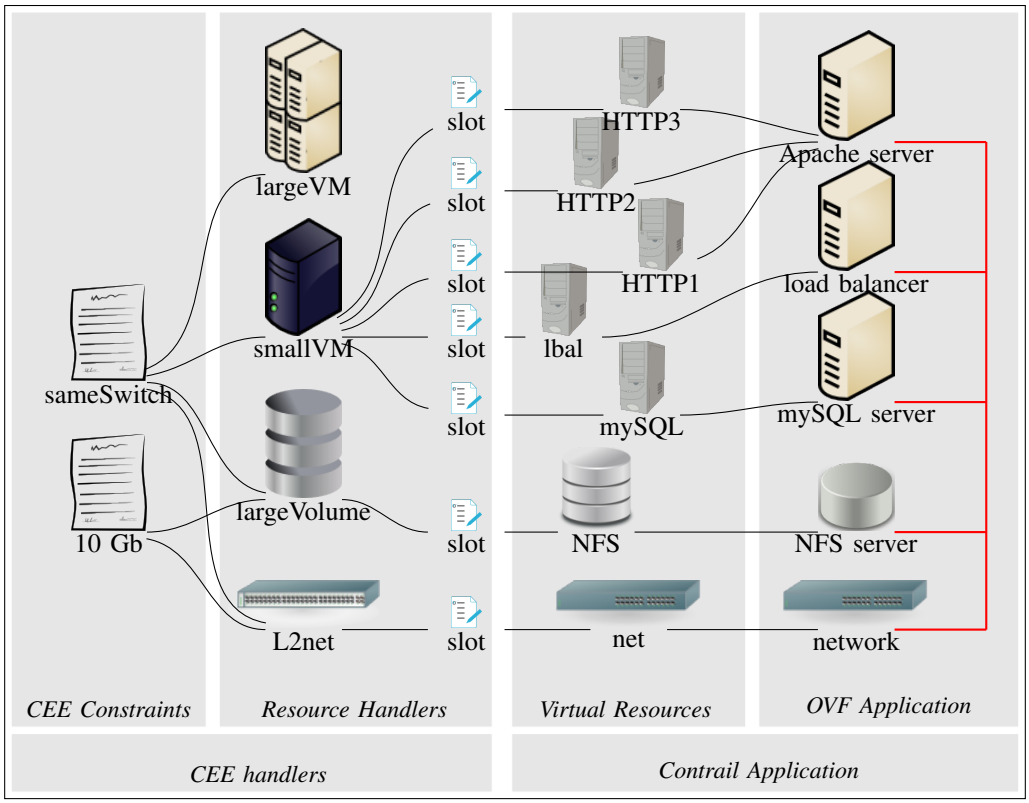

Fig. 2. Application deployed in a VEP Constrained Execution Environment: the OVF defines three virtual systems (Apache server, load balancer, mySQL server), a shared disk, all connected through a single network. Four virtual machines are deployed and handled by the smallVM handler: three instances of Apache virtual system and one instance of each other virtual system. The constraints require that all virtual resources are deployed on hardware connected through the same network switch and the shared disk be connected to the network through 10 Gb link.

avoid partial allocation failure. Once the deployment process is triggered, the following steps take place:

1) Networks: check that the OVF networks have been instantiated. If not, generate network items mapped on a basic level 2 ethernet switch handler of the CEE.
2) Resource requirements: for each deployed item, check that a free allocation slot is present on the handler. If not, create a new allocation slot and link it to the handler and to the OVF virtual resource item. This new slot has no associated resource.
3) Reservation: consider the list of allocation slots with no associated resource: reserve the missing resources. The resource allocator must consider all constraints associated to the resource handlers during the allocation process. If all requested resources cannot be allocated, the whole reservation step fails.
4) Networks: build (or update) the network structure. Consider the OVF network IP settings of the OVF virtual systems. If allocation is `AutomaticAllocation`, get an IP address from the address pool provided by the CEE network handler (must be compatible). If allocation is `manual`, check that the IP is accepted by the handler. Assign statically allocated IPs to machine interfaces.
5) Contextualization: generate the environment document for each virtual machine. VEP provides means to reference values defined during the deployment process through macros and to assign these values to OVF properties using the deployment document contextualization items. The contextualization item of a virtual machine can, for example, reference the static IP address assigned to another virtual machine.

After this last step, all data necessary for the deployment of the application on the provider are ready. The effective deployment is handled by plugins adapted to the provider IaaS.

F. Application elasticity management

To provide elasticity, the Contrail VEP supports dynamic addition of new virtual machines to an existing application through the incremental submission of new deployment documents. Elasticity requests can be sent by the user, the Contrail federation layer, or triggered by the SLA enforcement service following rules defined in the SLAs.

G. SLA enforcement and auditing

The CEE resource handlers define the information and the events which must be monitored during the life cycle of the environment. Contrail provides means to configure these handlers in order to record the monitoring data at specific locations and/or to publish these data on some media. Basic

support for data publication on a publish-subscribe system is currently provided. Monitoring data can also be stored and provided to users on request.

To allow full auditing capabilities of Contrail applications, VEP maintains the relationships between all internal objects created to manage the applications life cycle:

- which virtual machines have been generated from some OVF virtual system;
- which virtual machines are handled by a CEE handler;
- which virtual machines are running on a server (could belong to different CEEs or users);
- ...

VEP virtually allows to browse all objects deployed on a cloud provider. Browsing objects is, of course, subject to authorization, and only objects "accessible" to a user are shown to the user. The system also allows for *third party* auditing.

H. Application snapshots

To avoid introducing a specific format for exporting application snapshots, a Contrail application snapshot is represented as a standard OVF application. The snapshot of a running application is represented by an application template. OVF snapshot export is possible only for stopped application snapshots as all application state must be stored on disk images. A multi-provider federation application snapshot is obtained through the merge of all provider snapshots.

A Contrail application snapshot request can specify which deployed virtual machine or shared disk must belong to the snapshot. The OVF snapshot is generated from the original OVFs composing the application. Each snapshotted virtual machine is added to this OVF as an extra OVF virtual system, in the same environment (OVF collection) as the original virtual system. An OVF product section is associated to each generated virtual system containing the properties defined at deployment time. References to OVF shared disks images are replaced by references to the image snapshots. Networks are retained in the OVF descriptor. The original virtual system descriptors are maintained in the final OVF envelope: these undeployed virtual systems need to be present in order to maintain the elasticity capabilities of the snapshot. However, to avoid automatic deployment of these virtual systems, an OVF-2 `scaleout` section with a default value 0 is associated to each original virtual system. To keep track of specific features of Contrail across snapshots, for instance elasticity or the state of a network CEE handler (IP address pool), a Contrail extension section can be added to the final OVF document, for instance in a Contrail product section.

IV. Related work

Very few propositions to manage whole applications on the Cloud currently exist.

The Cloud Infrastructure Management Interface (CIMI, [5]) Model of the DMTF Cloud Management Working Group defines a framework for application life cycle management on a Cloud provider infrastructure. In the Contrail VEP, CIMI applications (called systems in CIMI) can be generated (imported) from an OVF document. CIMI also provides means to export a deployed CIMI system in the OVF format. The major difference concerning OVF is that Contrail VEP keeps track of all OVF details during the whole application life cycle while CIMI generates a system template. In the Contrail solution, the internal structure of OVF documents is exploited during the application lifetime for managing elasticity and for auditing. Another major difference between both propositions is that the Contrail VEP integrates support for elasticity and for federations through deployment documents, and for basic SLAs through the Constrained Execution Environments. Contrail also maintains the relationship between the allocated resources and the original OVF descriptors to facilitate auditing and SLA verification. The first VEP API is evolving and next release API might be derived from the CIMI API to favor interoperability.

The CompatibleOne [6] project develops a Cloud Management Software for managing applications across multiple clouds. CompatibleOne supports cloud brokering and cloud federations. The framework is implemented using the OCCI [7]–[9] protocol from OGF. The major difference with Contrail is that CompatibleOne currently uses its own format for application description and does not provide any support for OVF nor SLAs.

Wu et al [10] proposes resource allocation algorithms for SaaS providers who want to minimize infrastructure cost and SLA violations. The scope of our proposition is limited to the management of OVF distributed applications at IaaS level under SLA control. The SLA terms considered here are limited to resource provisioning. The proposed infrastructure allows to gather all constraints to be considered for resource provisioning and to provide these constraints to the resource allocators. The allocation algorithms themselves are not considered.

V. Conclusion

A major objective of the Contrail project is to develop a framework for the management of distributed applications on federated cloud providers under the control of Service Level Agreements. Heterogeneous providers offer various support for networking, virtual machines management, monitoring, resource allocation etc. Only few typical SLA terms are currently handled by some providers, mainly about placement or localization as in OpenStack. The Contrail VEP component described in this document allows to manage the whole life cycle of elastic applications described in OVF, an open standard, on heterogeneous IaaS providers. VEP brings support for SLAs at the provider level, although the effectiveness of this support depends on the IaaS capabilities concerning resource allocation and monitoring.

Acknowledgment

Contrail is a European Commission funded project under FP7 program and is funded by grant 257438. The authors would like to acknowledge the work done by all Contrail Consortium members and teams.

REFERENCES

[1] P. Harsh, F. Dudouet, R. G. Cascella, Y. Jegou, and C. Morin, "Using open standards for interoperability: Issues, solutions, and challenges facing cloud computing," in *Accepted to 6th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud*, October 2012.

[2] "Contrail: Open Computing Infrastructures for Elastic Services," http://contrail-project.eu/, last accessed August 2012.

[3] "Contrail Wiki," http://contrail.projects.ow2.org/xwiki/bin/view/Main/WebHome, last accessed August 2012.

[4] "Shibboleth," http://shibboleth.net/index.html, last accessed August 2012.

[5] D. Davis and G. Pilz, "Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP," vol. DSP-0263, May 2012, http://dmtf.org/standards/cloud.

[6] "The Open Source Cloud Broker," http://www.compatibleone.org/, last accessed August 2012.

[7] T. Metsch, A. Edmonds, R. Nyren, and A. Papaspyrou, "Open Cloud Computing Interface - Core," vol. GFD.183, June 2011, http://occi-wg.org/.

[8] T. Metsch and A. Edmonds, "Open Cloud Computing Interface - Infrastructure," vol. GFD.184, June 2011, http://occi-wg.org/.

[9] ——, "Open Cloud Computing Interface - RESTful HTTP Rendering," vol. GFD.185, June 2011, http://occi-wg.org/.

[10] L. Wu, S. K. Garg, and R. Buyya, "SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID2011)*, 2001, pp. 195–204.