Enhancing Grid Infrastructures with
Virtualization and Cloud Technologies

Second Year Cloud-like Management of Grid Sites Research Report

Deliverable D6.6 (V1.0)
4 June 2012

Abstract

This report presents the results of the research and technological development activities undertaken during the second phase of the project by the three tasks in which WP6 is divided. Mainly, this work has been focused on management of complex multi-tier applications, scaling, monitoring and balancing them to face peaks in demand. In addition, advanced networking (network isolation and firewalling) and storage capabilities (datastore abstraction and new transfer drivers) have been developed. TCloud as monitoring API and OCCI and Deltacloud as Virtual Machine Manager APIs have been implemented. Finally, an Inter-Cloud connector component has been introduced in StratusLab providing federation and brokering among sites.

StratusLab is co-funded by the European Community's Seventh Framework Programme (Capacities) Grant Agreement INFSO-RI-261552.

The information contained in this document represents the views of the copyright holders as of the date such views are published.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE MEMBERS OF THE STRATUSLAB COLLABORATION, INCLUDING THE COPYRIGHT HOLDERS, OR THE EUROPEAN COMMISSION BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE INFORMATION CONTAINED IN THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contributors

Name	Partner	Sections
Muñoz Frutos, Henar	TID	1, 2, 3, 4, 5, 7
Ignacio Blasco Lopez	TID	3, 5, 7
Juan Carlos Cuesta Cuesta	TID	3, 5
Huedo, Eduardo	UCM	1, 2, 4, 5, 7
Montero, Rubén S.	UCM	4, 5
Moreno, Rafael	UCM	4, 5
Llorente, Ignacio M.	UCM	4, 5

Document History

Version	Date	Comment
0.1	02 May 2012	Skeleton with initial information.
0.2	20 May 2012	Main chapters contribution.
0.3	30 May 2012	Executive summary, introduction and conclusions.
0.4	3 June 2012	Peer review of the document.
1.0	4 June 2012	Final edition.

Contents

List of Figures

1 Executive Summary

Grid infrastructures are typically static, with limited flexibility for changing application parameters: OS, middleware and resources in general. By introducing Cloud management capabilities, grids can become dynamic. Adding standard tools, such as virtual machines (VMs), resources can be repurposed on demand to meet the requirements of high priority applications. The Cloud platform controls which application images should be running and when. This means dynamic application stacks on top of the available infrastructure, such that any physical resource can be quickly repurposed on demand for additional capacity [6]. This document presents the work done on Cloud-like Management of Grid Sites (WP6) during the second year of the StratusLab project in the three main tasks.

- During the second year in the Dynamic Provisioning of Grid Service task, balancing mechanisms have been implemented in the Service Manager to configure the load balancers, new scalability policies have been included, and an advanced monitoring framework has been introduced for monitoring service KPIs.
- In the Scalable and Elastic Management of Grid Site Infrastructure task, StratusLab has improved the storage capabilities of OpenNebula with the datastore abstraction and new transfer drivers, its network management with network isolation and firewalling, and its technology for Cloud bursting.
- In the task for Cloud-like Interfaces Specific for the Scientific Community, TCloud for Claudia and monitoring, and OCCI and Deltacloud APIs for OpenNebula have been updated, authentication and authorization in OpenNebula have been improved, as well as the technology for Cloud federation and brokering.

Finally, most of the work done in WP6 has been tested by the advanced management and scalability scenario defined and implemented in WP2. This scenario involves the management of a business e-tier application and requires advanced Cloud functionality provided by StratusLab: multi-tier service management, KPI-driven scalability, load balancing, monitoring, secure networking and multi-Cloud deployments.

2 Introduction

The Joint Research Activity (JRA), carried out in WP6, develops advanced technology and features for application deployment on existing Cloud infrastructures through automatic deployment and dynamic provision of grid services as well as scalable Cloud-like management of grid site resources. In the second year, this objective has expanded to include requirements coming from commercial services, not just grid services. Along these lines, WP6 has provided the deployment and management of both grid services and commercial ones.

The present document reports about the work done in WP6 in the second year of the project. The developed software was already documented in D6.5 *Cloud-like Management of Grid Sites 2.0 Software* [11], whose design was defined in D6.4 *Cloud-like Management of Grid Sites 2.0 Design Report* [9].

In general, the work done in this work package for year two has involved:

- The inclusion of Cloud-like interfaces and a language based on standards (such as TCloud, OCCI, OVF or DeltaCloud) to increase the interoperability in Cloud service management.

- The introduction of a Service Manager, on top of the current Virtual Machine Manager, which is able to control and configure grid services as a whole providing scalability mechanism at the service-level (as part of the Service Manager) to scale up and down grid service components according to some Key Performance Indicators (KPIs) and hardware usage.

- Advanced networking capabilities that provide added features such that users can more dynamically create and configure deployment-specific virtual networks, in order to provide finer control and isolation of their system deployed in the cloud.

- Storage capabilities such as the OpenNebula datastores. There are other components related to storage capabilities in the StratusLab architecture, like the Marketplace and the Persistent Disk Service, but they have not been implemented as part of WP6.

- Monitoring systems at the physical, virtual hardware and grid service layers for regular administration tasks, accounting purposes and for triggering scalability mechanisms.

- The capability of interfacing a given Cloud site with another through the Inter-Cloud Connector, providing Cloud bursting, federation and brokering.

These objectives have been implemented in the three defined tasks of WP6.

Task T6.1 *Dynamic Provision of Grid Services* has been working towards the provisioning of services in a Cloud environment, where advanced management capabilities have been included. Multi-tier applications, which are defined in the OVF format, are deployed and configured automatically by using StratusLab contextualization. In addition, advanced management of these services is carried out to meet the dynamic needs of an application. Balancing and scalability capabilities are included to face peaks in demand. Finally this dynamism requires advanced monitoring techniques that deliver the required data to an intelligent element making the appropriate (re-)provisioning decisions. This work is explained in Chapter 3.

Task T6.2 *Scalable and Elastic Management of Grid Site Infrastructure* has adapted an open-source Virtual Machine Manager (OpenNebula) to the typical operations of a grid site, in particular virtual resource placement heuristics have been added to optimize different infrastructure metrics (*e.g.* utilization or energy consumption), Cloud-aware techniques for image and network management have been developed, and components for Cloud bursting have been improved. Chapter 4 describes the work performed in this task.

Task T6.3 *Cloud-like Interfaces Specific for the Scientific Community* has defined and developed the Cloud interfaces for the system for accessing to the different StratusLab components: Service Manager, Virtual Machine Manager, monitoring systems and the Inter-Cloud Connector. This work is described in Chapter 5.

Most of this work has been tested by the advanced management and scalability scenario defined and implemented in WP2, which is described in Chapter 6. This scenario is a multi-tier e-business application used as a proof of concept for demonstrating advanced capabilities that a Cloud platform can provide: multi-tier service management, KPI-driven scalability, load balancing, monitoring, secure networking and multi-Cloud deployments.

3 Dynamic Provision of Grid Services

Task 6.1 *Dynamic Provision of Grid Services* has been working towards the provision of services in a cloud environment (including grid and commercial services). Multi-tier services are deployed and configured automatically using descriptions in the OVF format [5] and by using the StratusLab contextualization (see D6.3[8]). In addition, advanced management of these services is carried out in order to face the dynamism of the application. Balancing and scalability capabilities are included to face peaks of demand and distribute the load for the right usage of the service. Finally this dynamism requires advanced monitoring techniques that deliver the required data to an intelligent element making the appropriate (re-)provisioning decisions.

3.1 Service Scalability

Services deployed over cloud technologies should benefit from scalability at service level, which conceals low level details from the user. StratusLab can provide scalability in each layer of the multi-tier application.

Claudia, the Service Manager, controls the service monitoring events and scalability rules. In addition, it is responsible for dynamically requesting virtualized resources from a Virtual Machine Manager like OpenNebula, trying to avoid over or under-provisioning and over-costs based on SLAs and business rules protection techniques.

As described in D6.4 [9] and D6.3 [8] , Claudia provides a means for users to specify their application's behavior in terms of adding or removing hardware resources [2] by means of *elasticity rules* [3]. The elasticity rules follow the Event-Condition-Action approach, where automated actions to scale up or down virtual machines can be executed. In relation to it, in StratusLab scalability policies have been defined to specify the number of nodes to be scaled, the chosen node to be removed and so on.

Taking into account the number of nodes to be scaled, we can have three different policies:

- 1-scale: This is the policy by default. It involves scaling up or down a node.
- n-scale: Scaling a number *n* of nodes. This is specified by the properties SCALE_UP_NUM_NODES and

SCALE_DOWN_NUM_NODES (in the ProductSection of the OVF) for scaling up and down.

- %-scale: Scaling a percentage of the currently deployed nodes. This is specified by the properties in the ProductSection SCALE_UP_PERCENTAGE and SCALE_DOWN_PERCENTAGE for scaling up and down.

In case a VM has to be undeployed, it is possible to configure the service for choosing the policy by the property SCALE_DOWN_POLICY in the ProductSection:

- random: A VM randomly chosen to be undeployed.
- lastly: The last VM deployed is undeployed.
- balancer: The load balancer decides which VM is to be undeployed, taking into account the node's activity.

3.2 Service Balancing

Service scalability requires the existence of a load balancer situated in front of the node to be scaled. This balancer is going to manage different virtual machine replicas of each type (for example different instances of Tomcat, JBoss, etc.) and balance the load among these replicas. In the commercial use case, which we are working on in WP2, the balancer is using a round robin algorithm, but any other algorithms can be used.

In addition, the balancer requires to be reconfigured when new replicas are deployed or undeployed. Claudia is in charge of this reconfiguration through the information collected in the OVF description. The following OVF extract illustrates how to configure the balancer in the virtual system specifying that it is a balancer and the management port.

```
<VirtualSystem ovf:id="balancer"
               rvr:balancer="true"
               rvr:lbport="8088">
```

The following extract shows how it is possible to relate a virtual machine to its load balancer.

```
<VirtualSystem ovf:id="apache"
               rvr:balanced="balancer">
```

3.3 Advanced Monitoring Techniques

In order to evaluate the VM execution status, the monitoring mechanisms constantly check the performance of the system. Monitoring systems evaluate hosts, virtual machines and services with respect to hardware, software and Key Performance Indicators (KPI) metrics.

Compared to the work from the project's first year, the monitoring systems have been extended to include any kind of probes (not just hardware information, but also service KPI), any kind of monitoring software (not just ganglia, for instance also collectd) and its framework has been designed to contain all the elements for monitoring (probes, collector, aggregator, database, API).

When a VM is deployed, the monitoring software and probes are introduced, installed, and configured in the virtual machine via the standard contextualization process. Once configured, the probes in each virtual machine report KPI information to the collector, which stores it in the database. Finally, it is possible to access to the information via the TCloud API (see 5.1.3).

3.3.1 Monitoring Installation by StratusLab contextualization

The contextualization mechanisms used for monitoring are the same ones utilized by Claudia (see [8]). This is specified in the ProductSection in the OVF file by using the Property section. Concretely, by using the the SCRIPT_LIST name of the property, it is possible to specify a set of scripts to be executed by StratusLab contextualization, as shown in the following extract (whitespace added for clarity):

```
<Property ovf:key="SCRIPT\_LIST"
          ovf:value="monitoring_rubisdb.sh/
                     collectd_ubuntu.tgz/
                     PluginTailBE.conf"/>
```

Claudia translates the SCRIPT_LIST property into the SCRIPT_EXEC in the CONTEXT of the OpenNebula template. With the SCRIPT_EXEC property, it is possible to execute scripts passed by contextualization. In addition, all the files and scripts are passed by contextualization to the VM.

```
SCRIPT\_EXEC="monitoring\_rubisdb.sh;"
```

Basic monitoring scripts are provided by StratusLab that install the monitoring systems software (collectd), configure them with the collector information, and ensure the right probes are executed.

3.3.2 KPI types

In addition to the metrics provided by the collectd software, it is possible to include any kind of plug-in (Mysql, Apache and so on) and also any *ad hoc* probe. Some metrics used in the use cases of StratusLab have involved:

Hardware information Information about hardware resources such as CPU, RAM, disk or network. DisksUsed, memoryUsed, cpuIdleSeconds are examples of this kind of monitoring information.

Apache, JBoss Some metrics for Apache or JBoss have been added by specific probes. It includes metrics like Apache CPU, heap or permGen.

Response time metrics Metrics for measuring the response time of the requests to Apache, Tomcat or JBoss. Metrics measured are requestDelay (for current response time) and requestDelayTotal (total response time).

Number of connections The number of established connections, like for example, HAEstablishedCons in the HAProxy load balancer.

4 Scalable and Elastic Management of Grid Site Infrastructure

Task T6.2 *Scalable and Elastic Management of Grid Site Infrastructure* has adapted an open-source VIM (Virtual Infrastructure Manager), OpenNebula [7], to the typical operations of a grid site. In particular, virtual resource placement heuristics have been added to optimize different infrastructure metrics (e.g. utilization or energy consumption), cloud-aware image and network management techniques have been developed, and the technology for Cloud bursting has been improved.

4.1 Virtual Resource Placement Heuristics

4.1.1 Placement Policy Definition

The placement of the VMs can now be defined on a VM-basis (restricted to `oneadmin` user) or globally for the datacenter. This allows admins to set a global optimization policy to meet specific goals. Four predefined policies have been included:

- Packing: Minimize the number of hosts in use by packing the VMs in the hosts to reduce VM fragmentation.
- Striping: Maximize resources available for the VMs by spreading the VMs in the hosts.
- Load-aware: Maximize resources available for the VMs by using those nodes with less load.
- Custom: Use a custom `RANK` expression.

4.2 Cloud-Aware Image Management Techniques

4.2.1 Support for User Data Injection in VMs

A common requirement for VM contextualization is the ability to push user specific data into the VM, most notably access keys. This requires first the ability to store arbitrary data associated with each user, and then a flexible method to select and include user data in the context device. Both requirements are now met by OpenNebula.

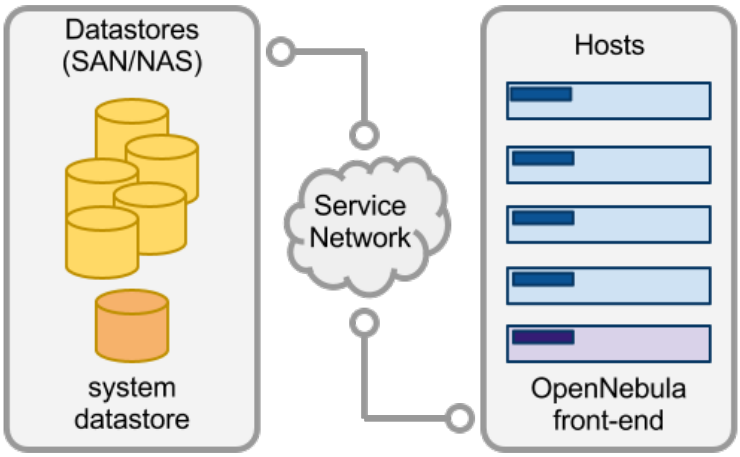

Figure 4.1: *OpenNebula datastores.*

4.2.2 Improved Storage Management

The storage capabilities of OpenNebula have been improved in StratusLab with the addition of the Datastore abstraction. A datastore is an abstraction of any storage medium for VM disk images (see Figure 4.1). Datastores are distributed to the hosts with specific transfer drivers. This allows a single host to include multiple datastores of different types.

OpenNebula 3.4 includes the following datastore types:

- System: to hold images for running VMs. Depending on the storage technology used, these temporal images can be complete copies of the original image, qcow deltas or simple filesystem links.
- `fs`: to store disk images in a file form. The files are stored in a directory mounted from a SAN/NAS server.
- `iscsi`: to store disk images in a block device form. Images are presented to the hosts as iSCSI targets.
- `vmware`: a datastore specialized for the VMware hypervisor that handles the vmdk format.

The system has been architected to be highly modular, so these base types can be easily adapted to any specific deployment.

Hosts are not tied to a single transfer mechanism (transfer driver) and now can access images from different datastores in different ways. A single VM can include disks from different datastores. Also the transfers associated with persistent or `save_as` images have been simplified. There are also new drivers to use in combination with the datastores: `qcow2`, `iscsi` and an improved version of

vmware that uses the vmdkfs tools, which add to the shared and ssh drivers in OpenNebula 3.4.

Nevertheless, in StratusLab 2.0, which includes OpenNebula 3.2, the Marketplace and the Persistent Disk services are still integrated using transfer drivers.

4.3 Cloud-Aware Network Management Techniques

4.3.1 Improved Network Management

When a new Virtual Machine is launched, OpenNebula will connect its network interfaces to the bridge or physical device specified in the Virtual Network definition. This will allow the VM to have access to different networks–public or private.

The OpenNebula administrator must take into account that although this is a powerful setup, it should be complemented with mechanisms to restrict network access only to the expected VMs, to avoid situations in which an OpenNebula user interacts with another user's VM. This functionality is now provided through Virtual Network Manager drivers. The OpenNebula administrator may associate one of the following drivers to each host:

- dummy: Default driver that doesn't perform any network operation. Firewalling rules are also ignored.
- fw: Firewall rules are applied, but networking isolation is ignored.
- 802.1Q: restrict network access through VLAN tagging, which also requires support from the hardware switches.
- ebtables: restrict network access through Ebtables rules. No special hardware configuration required.
- ovswitch: restrict network access with Open vSwitch Virtual Switch.
- vmware: uses the VMware networking infrastructure to provide an isolated and 802.1Q compatible network for VMs launched with the VMware hypervisor.

Therefore, network operations are now coupled with the VM lifecycle. This simplifies the management of networking (no hooks are needed) and solves previous issues with VLANs when migrating and restoring VMs. The network drivers define three actions (pre, post and clean) that can be easily customized if needed.

On the other hand, networks can be now defined with an arbitrary range including an starting and ending IP address, network and network mask, or CIDR notation. It is possible also to define a network and a starting IP address to lease addresses.

Finally, network leases can now be put on hold to reserve them. This comes in handy when there are some IP addresses within the VLAN already assigned (e.g.

.1 to the gateway). When a lease is put on hold, OpenNebula will not use it for a VM, until it is released.

4.3.2 Network Isolation and Firewall Management

As mentioned before, StratusLab has included support for host-managed VLANs to restrict network access through VLAN tagging. This mechanism is compliant with the IEEE 802.1Q standard, but it requires support from the hardware switches. Additionally, StratusLab has developed network access restriction through VLAN tagging with Open vSwitch, a production quality, multilayer virtual switch.

Alternatively to the use of VLAN tagging, it is possible to restrict network access through ebtables rules. The ebtables program enables transparent filtering of network traffic passing through a Linux bridge. This complements the previous automatic setup of simple firewall rules for TCP/UDP ports and ICMP traffic.

4.4 Cloud Bursting

Cloud bursting consists of combining local resources from a Private Cloud with remote resources from a Public Cloud, thus creating a Hybrid Cloud. The Public Cloud provider is usually a commercial Cloud service, such as Amazon EC2 or ElasticHosts.

OpenNebula supports Hybrid Cloud deployments fully transparent to infrastructure users, being the infrastructure administrator who takes decisions about the scale out of the infrastructure according to infrastructure or business policies. Therefore, there is no modification in the operation of OpenNebula to integrate Cloud services. A Cloud service is managed as any other host, but it may provide "infinite" capacity for the execution of VMs.

OpenNebula currently provides support for building Hybrid Clouds with Amazon EC2, ElasticHosts and RedHat Deltacloud. StratusLab has improved the driver to create hybrid clouds with Amazon EC2, to support most of the EC2 features like tags, security groups or VPC (Virtual Private Cluster).

However, each provider uses different formats to store images. Fortunately, as described in Section 4.2.2, OpenNebula shows a pluggable architecture based on the datastore abstraction that allows the integration of any storage backend. In particular, Amazon S3 could be integrated by means of a special datastore, so it would be possible to download, contextualize and integrate S3 images in a local image repository. Due to time restrictions, this kind of datastores for hybrid storage has not been developed in StratusLab.

Since VMs deployed on different clouds have to communicate through the Internet, a suitable communication channel (usually a VPN) has to be established between them. In the case of virtualizing a computing cluster, the virtual cluster front-end should be deployed in the Private Cloud with Internet connectivity to be able to communicate with those worker nodes deployed in the Public Cloud. The worker nodes could communicate with the front-end through a private local area network. Local worker nodes should be connected to this vLAN through a

virtual bridge configured in every physical host. External worker nodes should be connected to the vLAN with an OpenVPN tunnel, which has to be established between each remote node (OpenVPN clients) and the cluster front-end (OpenVPN server). With this configuration, every worker node (either local or remote) could communicate with the front-end and could use the common network services transparently.

5 Cloud like-Interfaces Specific for the Scientific Community

Task T6.3 *Cloud-like Interfaces Specific for the Scientific Community* defined the Cloud interfaces for the system to provide a Grid as a Service. During this year this task has involved the improvement of the Cloud IaaS API: TCloud for Service Manager and OCCI and Deltacloud for Virtual Machine Manager. In addition, TCloud has been chosen and implemented for monitoring. Also, OpenNebula's security has been improved, with cloud partitioning, improved logging and a new CloudAuth driver; a VM template repository has been developed, offering a predefined set of VMs (instance types) that users may instantiate; and the technology for Cloud federation and brokering has been created or improved.

5.1 Cloud IaaS API

StratusLab has to complement existing grid services by exposing Cloud-like APIs to users of the grid infrastructure. StratusLab works towards the use of Cloud-like Application Programming Interfaces (APIs) for managing Cloud Computing capabilities including resource sharing.

5.1.1 OCCI and Deltacloud as the OpenNebula APIs

The OpenNebula OCCI API is a RESTful service to create, control and monitor Cloud resources based on the OGF OCCI 0.8 API specification. The OpenNebula OCCI service, as shown in Figure 5.1, is implemented upon the new OpenNebula Cloud API (OCA) layer that exposes the full capabilities of an OpenNebula private cloud; and Sinatra[1], a widely used light web framework.

OpenNebula's OCCI API has been extended to include VM types, that can now be defined in the server configuration file and tagged with arbitrary information, like size, QoS parameters or price. These VM types can be programatically queried through the API. Also, OCCI provides user/group information in resources and extended information of resources. Finally, in order to support the new VLAN features in OpenNebula, the OCCI networks can now be defined through a template, as for VMs.

Apache Deltacloud is a REST-based API that abstracts the differences between

[1] www.sinatrarb.com

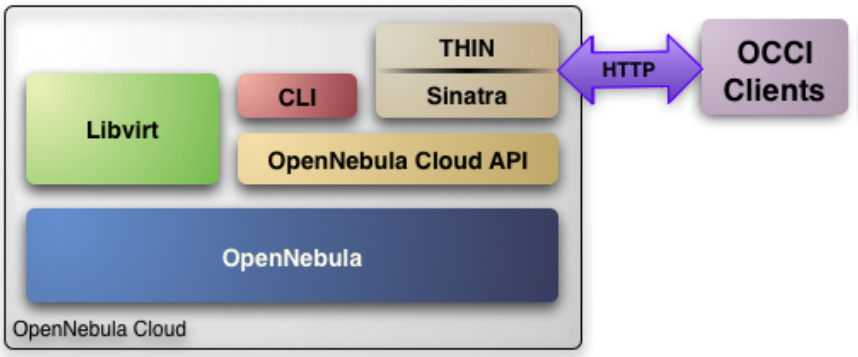

Figure 5.1: *OCCI implementation in OpenNebula.*

clouds, enabling the management of resources in different IaaS clouds using a single API. A series of back-end drivers translate the request to each Cloud provider's native API. Currently, all major Cloud service providers are supported. The OpenNebula back-end driver in Deltacloud has been updated to interact with OpenNebula 3.x clouds. This way, StratusLab sites could be accessed through the Deltacloud API, using different tools (see Figure 5.2).

5.1.2 TCloud as the Claudia API

The TCloud API [12] is a RESTful, resource-oriented API accessed via HTTP which uses XML-based representations for information interchange. It constitutes an extension of some of the main standardization initiatives in Cloud management, such as the Open Virtualization Format (OVF), defined by the DMTF, and the vCloud specification [5], published by VMware and submitted to the DMTF for consideration. TCloud API defines a set of operations to perform actions over: i) Virtual Appliances (VApp), which are Virtual Machines running on top of a hypervisor, ii) Hardware, the virtual hardware resources that the VApp contains, iii) Network, both public and private networks, and iv) Virtual Data Center (VDC), as a set of virtual resources (e.g. networks, computing capacities) which incarnate VApps. The TCloud API defines operations to perform actions over the listed resources categorized as follows: Self-Provisioning operations to instantiating VApps and VDC resources and Self-Management to manage the instantiated VApps (power on a VApp). In addition, it provides extensions for monitoring, storage, and so on.

StratusLab incorporates an implementation of the TCloud specification, the tcloudserver as the Claudia API.

5.1.3 TCloud as Monitoring API

The TCloud API also contains a set of extensions, like monitoring. This includes capabilities for obtaining information about the virtual resources used by VApps,

Figure 5.2: *Accessing OpenNebula through Deltacloud.*

VDCs or Organizations. Some examples include obtaining the amount of memory used by a Virtual Machine or obtaining the consumed bandwidth for a VDC.

It involves the operations admitted by Cloud resources for monitoring actions. Each operation is described using the same notation as TCloud API core operations. URIs are abbreviated using the `<item-uri>` form, where item may be an organization, a VDC, a service, a BB instance, or anything that can be measured.

5.2 Authentication and Authorization

One of the main characteristics of an IaaS cloud is its multi-tenancy nature. In order to efficiently implement a multi-tenant system, a flexible user system is needed, which allows the definition of user groups and access control lists to define specific access rights to each virtual resource. Therefore, the user system of OpenNebula has been extended to support groups. A user now is part of a user group. By default users in the same group can list and share (if labeled as public) any resource type (network, VMs, disk images etc.). The access control to each resource has been also improved with the addition of ACLs. An ACL express the user (or set of users) that may perform a given operation (e.g. create, delete or deploy) on a given virtual resource or set of them (e.g. VMs, networks, hosts or images).

Also, Cloud requests can be routed to an specific cluster with its own storage and network resources to better isolate public Cloud users. Usually medium to large Cloud sites structure their resources on multiple clusters, each with its own storage and networking systems. Cloud users are assigned to an specific cluster to prevent image thrashing across large datacenters. This feature extends the previous cluster concept available in OpenNebula 3.0.

Some attributes in VM Templates (`DISK/SOURCE`, `CONTEXT/FILES`, `NIC/MAC`

and NIC/VLAN_ID) have been restricted, because they can be easily used to gain oneadmin access or to compromise VMs of another user. There are new authentication drivers for LDAP (with base and group filtering), for Cloud API and OpenNebula front-end servers (server-based drivers). Also, as some of the drivers may take some time to authenticate a request (e.g. LDAP), session tokens can now be cached by OpenNebula. Finally, a new permission set has been included to manage access control to virtual resources. The new permissions overcome the limitations of the previous PUBLIC attribute and allow users to share resources in multiple ways. Combined with the ACL system (also simplified to match the new permissions), this allows the implementation of multiple roles.

The OpenNebula authentication and authorization system has been extended in three areas. First, it now avoids some potential security holes when the end-user may choose the driver to authenticate with OpenNebula, specially when using X.509 certificates. Second, the security of public Cloud API has been strengthened by including special server accounts to run the services, these server processes can use symmetric cryptographic ciphers or X.509 to authenticate with OpenNebula. And third, there have been improvements in the X.509 and SSH authentication methods like native support for proxies or better support for DN strings.

Also, a new framework has been included to add logging information to the servers. In particular, the logging facilities of the OpenNebula's Cloud API have been extended to ease the maintenance and deployment of several services like the Sunstone graphical interface or the OCCI API server (also used by the Deltacloud API drivers).

Finally, there is a new CloudAuth driver that delegates the authentication to the OpenNebula core. Therefore any OpenNebula auth driver can be used to authenticate Cloud users or the Sunstone web UI.

5.3 VM Template Repository

Usually IaaS clouds offer a predefined set of VMs (instance types) that users may instantiate. This leads to a simplification of the provisioning interface for final users that are only allowed to instantiate a predefined error-free set of VMs. The predefined instances may include different OS types, packed with multiple software stacks (i.e. virtual appliances). This mechanism is even more robust in combination with the new group and user access control lists. In this way, access to a given instance type can be granted only to a specific set of users. Therefore, OpenNebula has been extended to include a new Template Repository that allows OpenNebula administrators and users to register VM definitions in the system, to be instantiated later. These templates can be instantiated several times, and also shared with other users.

5.4 Cloud Federation

Cloud federation is very similar to Cloud bursting, described in Section 4.4, but in this case the remote Cloud provider is a partner infrastructure, such as another

StratusLab site running a different OpenNebula instance. Therefore, the exchange of resources can be made in both ways.

The technology to perform Cloud federation is very similar to the Cloud bursting case, but using the public Cloud interface of StratusLab sites. For computing, it is possible to access other OpenNebula instances using the Deltacloud adaptor, leveraging the Deltacloud API for OpenNebula that has been updated in StratusLab (see Section 5.1.1), or the ONE2ONE adapter, which has been entirely developed in StratusLab. For storage, in this case there is no need to import or export images, but just to use the same Markeplace service for them. For networking, the solution based on VPNs is still suitable.

5.5 Cloud Brokering

Cloud service brokering is a form of Cloud service intermediation, in which a company or other entity adds value to one or more Cloud services on behalf of one or more consumers of that service [4]. The broker role does many of the same things that a traditional IT services provider does in a service aggregator role, but also addresses additional complexities, particularly relevant to Cloud Computing and to achieving specific IT outcomes. The brokering service can provide a set of functionalities:

- Integration brokerage – integration of different Cloud providers
- Governance – policy compliance of Cloud service consumption
- Community management – manage the provisioning of services and consumers among the different Cloud providers

In Stratuslab, Claudia assumes the role of this brokering service. Claudia is a Service Manager and can work on top of several Cloud providers. By an aggregated API (TCloud), Claudia can invoke each provider in the same way. Then, the aggregated API is in charge of translating from the TCloud API and model to the Cloud provider API.

In order to work on this mode, a new module has added to the Claudia distribution. This is placement decision module, which drives the placement decisions among different sites. This module is based on a set of business rules which defines the user's policies in the OVF. Depending of these rules, the services and virtual machines are deployed in a particular Cloud infrastructure.

6 Advanced management and scalability use case

Cloud services potentially allow enterprises to outsource their storage, networking and processing needs so that they do not have to invest in the infrastructure, converting capital expenses into operating expenses. By hosting their application in the cloud, instead of on their own installations, they can more easily face peaks in demand, while maintaining high service availability and only paying for the resources actually used.

Modern internet applications are complex software implemented in several layers, differentiating the user front-end, the business logic and the database. These are multi-tier applications, for which any Internet web application serves as an example.

As explained in D2.3 deliverable [10], the advanced management and scalability scenario, renamed from commercial application, is focused on the deployment and management (including scalability) of a multi-tier application. Concretely, the application deployed is the website of a shopping mall. As any SaaS application, it is composed by three tiers:

1. Presentation tier: The presentation layer represents the interface between the user and the rest of the application. In order to guarantee performance during peak demand, this layer will scale. It is implemented by an Apache web server which hosts the webpages of the application.

2. Business Logic tier: It has the business logic of the application. It is implemented by a JBoss and a set of ears. This layer can be scaled if needed, consequently, a load balancer will be included.

3. Back-end tier: All the data in the database or in the storage service belong to this tier. MySQL is the database management system in charge of the data.

6.1 StratusLab Solution

Nowadays, Cloud providers are only focusing on the deployment and management of single VMs, their scalability according to hardware information (CPU, RAM or disk) or deployment on a single datacenter (e.g. Amazon site regions [1]). StratusLab goes beyond the state of the art providing a set of high Cloud services

which can provide advanced functionality such us multi-tier application management, scalability and balancing, advanced security, and multi-Cloud deployment. Concretely, these StratusLab components are:

OpenNebula This is the Virtual Machine Manager in charge of controlling VMs. It is the core of StratusLab and has been extended to provide this advanced functionality for networking and storage.

Claudia This is the Service Manager responsible for controlling complex multi-tier applications. It includes their deployment, scalability based on service KPIs and customers' requests and balancing among different replicas.

Advanced networking capabilities They involve firewalling and virtual local area networks, which isolate the service from others deployed in the Cloud.

Advanced storage capabilities Management of data through services, like using the persistent disk service to store the application database.

Advanced monitoring techniques Scalability requires monitoring information based on application KPIs, not just hardware information.

Marketplace Repository of stock images to store the individual services (Apache, JBoss, MySQL) to be used in the application.

Inter-Cloud Connector This is the component is charge of Cloud federation and brokering, so that, services can be deployed on multiple sites for redundancy or access to more resources.

Standards Standard APIs and format descriptions allow for interoperability. TCloud is used as Service Manager and monitoring API, OCCI as Virtual Machine Manager API, and OVF is utilized as service description language.

It is possible to see that most of StratusLab components are required for providing the functionality that e-business applications are demanding. Figure 6 specified the features used in the use case located them in the StratusLab architecture v2.0.

These high level Cloud services have been tested and validated by this scenario. Some conclusions and lessons learned are explained in Deliverable D2.5 [10].

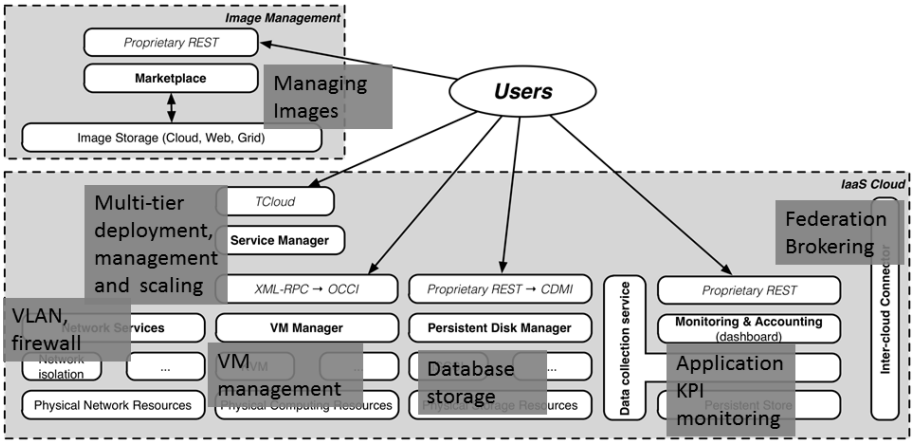

Figure 6.1: *Scenario functionalities and StratusLab architecture v2.0*

7 Conclusions

This document has provided an overview of the work done within the scope of WP6 in the second year of the project, concerning the dynamic provision of grid services, the development of a scalable and elastic management of grid site and Cloud like-interfaces specific for the scientific community.

In this second year, commercial services have been deployed in the Cloud. These services have been defined by OVF and deployed by using Claudia. New scaling rules have been added this year for driving the scalability of the service (considering scale down policies and the number of nodes affected by scaling). Claudia has also implemented some functionality to configure the balancers for the virtual machine replicas. This scalability is driven by KPIs which are collected in the monitoring systems.

In addition some work has been done in order to adapt OpenNebula to the typical operations of a grid site. In particular, the development of Cloud-aware image management techniques, with the new datastore abstraction and new transfer drivers; the development of Cloud-aware network management techniques, with network isolation and firewalling; and the improvement of technology for Cloud bursting.

The task related to Cloud-like interfaces uses as standard interfaces TCloud, OCCI and Deltacloud. OpenNebula's security has been improved, with Cloud partitioning, improved logging and a new CloudAuth driver; a VM template repository has been developed, offering a predefined set of virtual machines (instance types) that users may instantiate; and technology for Cloud federation and brokering has been created or improved.

Most of the work implemented in the WP6 is demonstrated by the advanced management and scalability use case in WP2, where a multi-tier service is deployed, managed, scaled, and balanced, even across multiple clouds.

Glossary

ACL	Access Control List
Appliance	Virtual machine containing preconfigured software or services
APEL	Accounting Processor for Event Logs
Appliance Repository	Repository of existing appliances
CDDLM	Configuration Description, Deployment, and Lifecycle Management
CE	Compute Element
DHCP	Dynamic Host Configuration Protocol
DMTF	Distributed Management Task Force
Front-End	OpenNebula server machine, which hosts the VM manager
Hybrid Cloud	Cloud infrastructure that federates resources between organizations
IaaS	Infrastructure as a Service
IP	Infrastructure Provider
Instance	a deployed Virtual Machine
JRA	Joint Research Activity
KPI	Key Performance Indicator
Machine Image	Virtual machine file and metadata providing the source for Virtual Images or Instances
NFS	Network File System
Node	Physical host on which VMs are instantiated
OASIS	Organization for the Advancement of Structured Information Standards
OCCI	Open Cloud Computing Initiative
OGF	Open Grid Forum
OVF	Open Virtualization Format
Public Cloud	Cloud infrastructure accessible to people outside of the provider's organization
Private Cloud	Cloud infrastructure accessible only to the provider's users
Regression	Features previously working which breaks in a new release of the software containing this feature
Service Manager/SM	A toolkit to provides Service Providers to dynamically control the Service provisioning and scalability
Service Provider/SP	The provider who offers the application to be deploy in the Cloud
SMI	Service Manager Interface

SLA	Service Level Agreement
SE	Storage Element
SSD	Solution Deployment Descriptor
Virtual Machine / VM	Running and virtualized operating system
TCloud	It is a RESTful API use for Cloud service management
VMI	Virtual Manager Interface
VIM	Virtual Infrastructure Manager
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
Web Monitor	Web application providing basic monitoring of a single StratusLab installation
Worker Node	Grid node on which jobs are executed

References

[1] Amazon. Amazon elastic compute cloud (amazon ec2). Online resource., 2012. http://aws.amazon.com/ec2/.

[2] J. Cáceres, L. M. Vaquero, L. Rodero-Merino, A. Polo, and J. J. Hierro. Service Scalability over the Cloud. In B. Furht and A. Escalante, editors, *Handbook of Cloud Computing*, pages 357–377. Springer US, 2010. 10.1007/978-1-4419-6524-0_15.

[3] C. Chapman, W. Emmerich, F. G. Márquez, S. Clayman, and A. Galis. Software architecture definition for on-demand cloud provisioning. In *HPDC '10: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 61–72, New York, NY, USA, 2010. ACM.

[4] D. M. Smith (Gartner Inc.). Hype Cycle for Cloud Computing, 2011.

[5] DMTF. Open virtualization format specification. Specification DSP0243 v1.0.0d. Technical report, Distributed Management Task Force, Sep 2008. https://www.coin-or.org/OS/publications/optimizationServicesFramework2008.pdf.

[6] E. Huedo, R. Moreno-Vozmediano, R. Montero, and I. Llorente. Architectures for Enhancing Grid Infrastructures with Cloud Computing. In M. Cafaro and G. Aloisio, editors, *Grids, Clouds and Virtualization*, Computer Communications and Networks, chapter 3, pages 55–69. Springer, 2011.

[7] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 13(5):14–22, 2009.

[8] Stratuslab Consortium. Deliverable 6.3 Cloud-like Management of Grid Sites 1.0 Research Report. Online resource., 2011. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d6.3-v1.0.pdf.

[9] Stratuslab Consortium. Deliverable 6.4 Cloud-like Management of Grid Sites 2.0 Design Report. Online resource., 2011. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d6.4-v2.0.pdf.

[10] Stratuslab Consortium. Deliverable 2.5 Report on Evaluation of StratusLab Products. Online resource., 2012. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d2.5-v1.0.pdf.

[11] Stratuslab Consortium. Deliverable 6.5 Cloud-like Management of Grid Sites 2.0 Software. Online resource., 2012. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d6.5-v1.0.pdf.

[12] Telefónica. TCloud API Specification, Version 0.9.0. Online resource., 2010. http://www.tid.es/files/doc/apis/TCloud_API_Spec_v0.9.pdf.