Enhancing Grid Infrastructures with
Virtualization and Cloud Technologies

Cloud-like Management of Grid Sites 2.0 Design Report

Deliverable D6.4 (V2.0)
15 December 2011

Abstract

This document reports about the design of the new developments for the components involved in WP6. It involves getting interoperability by the usage of standards APIs like TCloud, OCCI or CDMI and the definition of services and virtual machines by OVF format. In addition, new components have been incorporated to WP6 work: the network manager provides networking configuration, isolation by VLANs and firewall management, the storage manager allows for setting up images and the inter-cloud connector works towards the instantiation of VMs on different clouds, both private and public ones. Finally, some scalability policies have been added in the service manager and the monitoring and accounting systems have been updated due to new users' requirements.

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	2, 3, 4, 7, 8, 9
Caceres Expósito, Juan Antonio	TID	3
Lopez Lopez, Jose Manuel	TID	1, 2, 4, 7
Huedo, Eduardo	UCM	2, 3, 5, 6, 7, 8
Montero, Rubén S.	UCM	2, 3, 5, 6, 7, 8
Floros, Vangelis	GRNET	7
Konstantinou, Ioannis	GRNET	7

Document History

Version	Date	Comment
0.1	03 Oct. 2011	Document structure
0.2	10 Oct. 2011	Update Cloud APIs
0.3	11 Oct. 2011	Monitoring chapter and update in scalability
0.4	14 Oct. 2011	Networking, Storage, Federation
0.5	31 Oct. 2011	Introduction, Executive Summary and Conclusions.
0.6	21 Nov. 2011	Revision
0.7	24 Nov. 2011	New version to satisfy revision comments
0.8	15 Dec. 2011	Final English and grammar corrections
2.0	15 Dec. 2011	Final Edition

Contents

List of Figures

List of Tables

1 Executive Summary

The Joint Research Activity (JRA), carried out in WP6, develops advanced features for the innovative automatic deployment and dynamic provision of grid services as well as scalable cloud-like management of grid site resources. The work on this work package involves the implementation of innovative functionalities in StratusLab.

D4.4 *Reference Architecture for StratusLab Toolkit 2.0* described the StratusLab architecture for Y2, as an updated version containing new components coming from new users' requirements and new topics. Thus, the present document, D6.4, includes components or functionalities, identified in D4.4, which need developments and updates: i) interoperability, ii) service manager, iii) networking manager, iv) storage manager, v) monitoring and accounting and vi) inter-cloud connector.

Interoperability is an important key in the adoption of Cloud services. On one hand, Cloud APIs are required for the distribution of IaaS Cloud and its access outside the infrastructure. Due to it, StratusLab has been centered on standard APIs like TCloud, OCCI, CDMI for the service manager, virtual machine manager, storage and monitoring interfaces. On the other hand, the Open Virtualization Format (OVF) is adopted in StratusLab for defining the service and virtual machine details.

The service manager, Claudia, is the framework that manages a service (virtual machine, network and storage as a whole) and optimizes its resource utilization by dynamically scaling up/down services through elasticity rules. In this version of the document, new scalability policies are included–the percentage of nodes to be scaled and lazy scaling down.

The networking functionality that is going to be added in year 2 involves network configuration, allowing a easier coexistence with DHCP servers; network isolation, by means of VLANs; and firewall management, allowing filtering of virtual machine traffic based on simple rules, e.g. TCP ports.

Regarding storage, WP6 is working on an Image Manager, which allows OpenNebula administrators and users to set up images to be used in VMs easily, and describes its integration with the StratusLab Marketplace (for system images) and with the Persistent Disk Service (for data).

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 metrics. In addition the accounting systems exist for keeping track of the amount of computing resources consumed per user over a period of time.

Finally in year 2, StratusLab is starting to consider inter-cloud scenarios. In particular, Stratuslab is considering three scenarios: cloud bursting (to a public Cloud), federation (among two equal partners clouds, or StratusLab sites) and brokering (by the existence of a broker). New developments are required for each scenario in OpenNebula drivers or in Claudia modules.

2 Introduction

The Joint Research Activity (JRA) activity, carried out in WP6, develops advanced features on innovative automatic deployment and dynamic provision of grid services as well as scalable cloud-like management of grid site resources. More specifically, the objectives to be accomplished can be expressed as [19]: i) the extension of currently available service-level open-source elasticity frameworks on top of cloud infrastructures, ii) the invention of new techniques for the efficient management of virtualized resources of grid services and iii) the inclusion of novel resource provisioning models based on cloud-like interfaces.

The present document is a continuation to the work done in D6.1, *Cloud-like Management of Grid Sites 1.0 Design Report* [20], where an updated StratusLab reference architecture was defined for year 1.

2.1 StratusLab Architecture

The document D4.4 *Reference Architecture for StratusLab Toolkit 2.0* [21] documented the new updated version of the architecture for StratusLab. Figure 2.1 represents the this new version. It identifies two main blocks: "Image Management" and "IaaS Cloud" and the main interactions users have with the system. The IaaS Cloud contains the virtual machine manager, the service manager (see Chapter 4) and the monitoring and accounting manager (see Chapter 7), components that were already explained in D6.1[20]. New for v2.0 are the storage manager (see Chapter 6), network manager (see Chapter 5) and a control dashboard. The dashboard is an upgrade of the web monitor service, with the important addition of accounting/-billing functionality. The network manager provides 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. The storage manager includes the persistent disk manager and the OpenNebula image manager. Another important new component is the "Inter-Cloud Connector" (see Chapter 8), which is responsible for interfacing a given cloud site with another. Due to this, new topics are established in D6.4 with respect to D6.1: storage, networking and inter-cloud.

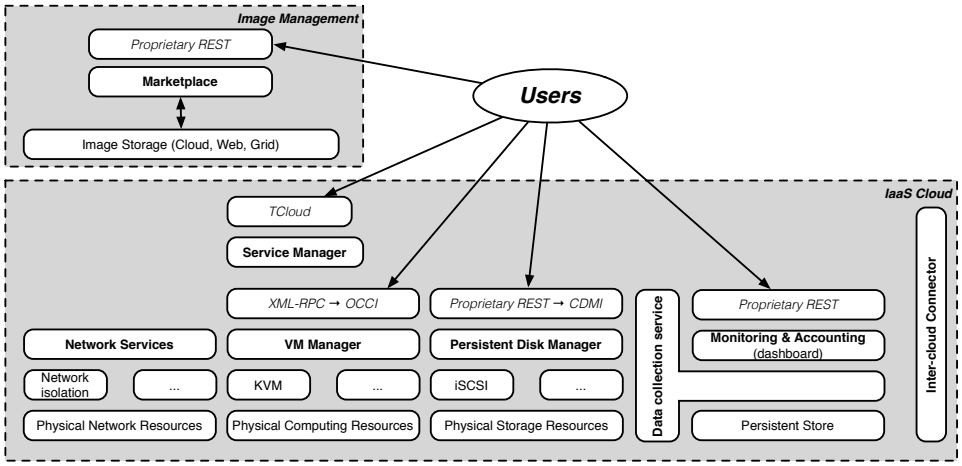

Figure 2.1: *StratusLab Architecture*

2.2 Technical work in WP6

D4.4 explained the StratusLab architecture updated for Y2 including information about the different components. This document, D6.4, provides information about these new topics in StratusLab, mainly materialized as components. Table 2.1 presents the topics for WP6, which includes gaps identified in WP4, requirements from customers and other topics included in StratusLab Y2 work plan.

Table 2.1: *Topics and Components*

Topic	Description	Artefact	Work in StratusLab
Interoperability	Cloud-like APIs	TCloud	extended
		OCCI	used
		jclouds	developed
		CDMI	possibly taken from VENUS-C
		(Network API)	possibly taken from Mantychore
		(Accounting API)	possibly taken from VENUS-C
	Language definition	OVF	extended
Scalability	Scaling actions SLA-powered services	Claudia scaling policies	extended
Networking	Network configuration	OpenNebula Contextualization	extended
	Network isolation	OpenNebula Network Manager drivers	developed
	Firewall management	OpenNebula Network Manager drivers	developed
Storage	Persistent disk	Persistent Disk Service	developed in WP5
	Image and volume management	OpenNebula Image Manager drivers	developed
Monitoring & Accounting	Cloud monitoring	Solution based on Ganglia or collectd	extended
	Cloud accounting	OpenNebula Accounting Module	developed
Federation	Cloud Bursting	OpenNebula Cloud drivers / OpenNebula Image Manager driver / OpenVPN	used / developed / used
	Cloud Federation	OpenNebula Cloud drivers / OpenNebula Image Manager driver / OpenVPN	used / developed / used
	Cloud Brokering	Claudia Placement Module	developed

3 Interoperability

Interoperability is an important topic for StratusLab, allowing users to access different sites uniformly and to reuse running virtual images. StratusLab is focused on the use of standards for its software implementations so that they can be interoperable with other providers, avoiding undesirable vendor lock-in. StratusLab works on the introduction of standard APIs for the Cloud services and the identification of a language for defining the service and virtual machines.

3.1 Cloud-like Application Programming Interfaces

StratusLab tries to provide grid users with a homogeneous computing environment that simplifies application management and hides the infrastructure's complexity. The integration of grid and cloud technologies will bring grid application developers a more dynamic and flexible computing environment. In this framework, StratusLab has to complement existing grid services by exposing cloud-like APIs to users of the grid infrastructure. This will allow existing users to experiment with these new APIs and to develop new ways to use grid resources.

In this direction, StratusLab works towards the use of cloud-like Application Programming Interfaces (APIs) for managing cloud computing capabilities including computation, networking, storage and so on. That is, service providers (e.g. grid administrators) can use programmatic APIs to access to the shared resources in order to manage them. Thus, the service providers request resources from the infrastructure providers or IT vendors to deploy the services and virtual machines.

Concretely, StratusLab is working on the APIs shown in Table 3.1. The following sections will analyze these APIs.

As can be seen in the table, StratusLab is evaluating solutions coming from other collaborating projects. For networking, the Mantychore project [13] is developing a system to provide network infrastructure resources as a service. For accounting, the VENUS-C project [26] will provide a solution based on OGF's Usage Records [25]. Also, VENUS-C is developing a storage service supporting CDMI.

3.1.1 TCloud

TCloud [24] is a RESTful API, submitted by Telefónica to the DMTF for consideration, which constitutes an extension of some of the main standardization ini-

Table 3.1: *Cloud-like APIs in StratusLab*

	TCloud	OCCI	CDMI	jclouds	Others
Service Manager	X				
VM Manager		X		X	
Network Manager		X			Mantychore
Storage Manager		X	X	X	
Monitoring & Accounting	X				VENUS-C

tiatives in Cloud management, such as the Open Virtualization Format (OVF) and vCloud [27]. This API provides operations for services and VM self-provisioning operations, resource self-management and monitoring extensions.

The TCloud API includes an extension for monitoring which can be used together with other TCloud capabilities. It incorporates the operations admitted by cloud resources for monitoring actions. Each operation is described using the same notation as TCloud API [24] core operations. URIs are abbreviated using the <item-uri> form, where item may be an organization, a Virtual Data Center, a service, a service instance, and anything which can be measure.

The implementation of TCloud used in StratusLab is already provided by Claudia, the StratusLab Service Manager, to be the access point for service providers.

3.1.2 Open Cloud Computing Interface (OCCI)

OCCI [15] is the OGF standard for infrastructure management interfaces in the context of cloud computing. StratusLab will support this standard, which is already supported in OpenNebula, and will ensure its interoperability with other implementations. Therefore, it will be possible to manage StratusLab compute, storage and network resources with the OCCI standard interface provided by OpenNebula using a RESTful service.

3.1.3 Cloud Data Management Interface (CDMI)

CDMI [18], released by SNIA (Storage Networking Industry Association), defines the functional interface that applications will use to create, retrieve, update and delete data elements from the Cloud. Therefore, this could be another interface for accessing the Persistent Disk Service [21]. In fact, there is a ongoing collaboration with VENUS-C [26] to incorporate its CDMI implementation inside StratusLab.

3.1.4 jclouds

jclouds [12] is an open source library to access cloud providers and cloud software stacks using Java and Clojure. With the API it is possible to use portable abstractions or cloud-specific features.

For example, the jclouds Compute API is a portable means of managing nodes in clouds. It can manage nodes as a set and address resources in any cloud without needing separate connections. It also has a Template feature which allows users to search for configurations that match parameters such as CPU count or operating system. Finally, it contains utilities to execute scripts as part of the bootstrap process of the nodes. A binding of the jclouds Compute API will be developed around OpenNebula, using the Java OCA (OpenNebula Cloud API).

In addition, the jclouds BlobStore API is a portable means of managing key-value storage providers such as Amazon S3. It offers both asynchronous and synchronous APIs, as well as Map-based access to data. A binding of this API could be developed around the Persistent Disk Service.

3.2 Virtual Appliance (Service) Language Definition

As StratusLab works with different sites with a variety of services, it is required that virtual machines are portable between sites. Thus, StratusLab needs to guarantee interoperability in service and virtual machine definitions between sites running the StratusLab distribution initially, and eventually to develop solutions for letting users and sites utilize other cloud services beyond the StratusLab frontier, including public and commercial clouds.

In order to get services and VM portability among sites, StratusLab works on the definition of the virtual appliance in a standard way. As a result of D6.1 [20], the Open Virtualization Format (OVF) was selected.

StratusLab defines virtual appliances, or services, by using OVF. This OVF format is going to be the data model used by the TCloud API for the access to the Service Manager. In addition, the use of OVF also for the definition of the VMs for the OCCI API, to access the VM Manager, in the OCCI requests' payload, is also being analyzed.

3.2.1 Open Virtualization Format

The Open Virtualization Format (OVF) objective [7] is to specify a portable packaging mechanism to foster the adoption of Virtual Appliances (VApp) (i.e. preconfigured software stacks comprising one or more virtual machines to provide self-contained services) as a new software release and management model (e.g. through the development of virtual appliance lifecycle management tools) in a vendor and platform neutral way (i.e., not oriented to any virtual machine technology in particular). OVF is optimized for distribution and automation, enabling streamlined installations of VApps.

3.2.2 OVF extensions required for the grid site specification

OVF many of the StratusLab requirements, but still there are some pending issues which are not covered, as shown in D6.3 [22]. OVF can be extended easily when needed; our extensions include:

Custom automatic elasticity Service providers specify rules and actions to au-

tomatically govern the scaling up and down of the service. The scalability rules are part of the OVF file and specify when the service can scale up and down and under which conditions.

Performance monitoring (KPI) Service providers define key performance indicators that are monitored by the cloud infrastructure, e.g. to trigger the elasticity actions. It involves including information about the KPI which drives the scalability (e.g. a service KPI or hardware KPI) in the OVF description.

Number of replicas to be deployed The maximum and minimum number of replicas to be scaled up and down.

Self-configuration Virtual machines composing the service are dynamically configured, e.g. IP addresses. It allows specifying information about software configuration in the OVF that will be used later for contextualization.

4 Service Scalability

Scalability can be defined as "the ability of a particular system to fit a problem as the scope of that problem increases (number of elements or objects, growing volumes of work and/or being susceptible to enlargement)" [3]. The framework used in StratusLab that provides scalability mechanism is Claudia. It provides a wider range of scalability mechanisms and a broader set of actions that can be undertaken (addition, removal, reconfig, federation...) on top of several cloud infrastructure providers. It also brings flexibility allowing combinations of several metrics.

The Claudia platform [23] is an advanced service management toolkit that allows service providers to dynamically control the service provisioning and scalability in an IaaS Cloud. Claudia manages services as a whole, controlling the configuration of multiple VM components, virtual networks and storage support by optimizing the use of them and by dynamically scaling up/down services applying elasticity rules, SLAs and business rules.

Claudia is responsible for the instantiation of service applications (controlling the service lifecycle) and dynamically asking for virtualized resources to a virtual machine manager like OpenNebula, trying to avoid over/under provisioning and over-costs based on SLAs and scalability rules.

4.1 Scalability policies

Claudia provides a means for users to specify their application behavior in terms of adding or removing more of the same software or hardware resources [3] by means of *elasticity rules* [4]. The elasticity rules follow the Event-Condition-Action approach, where automated actions to resize a specific service component (e.g. increase/decrease allocated memory) or the deployment/undeployment of specific service instances are triggered when certain conditions relating to these monitoring events (KPIs) hold.

By using it, it is possible to establish some scalability rules indicating when Claudia should scale up or scale down. More information and examples about scalability rules for grid services were defined in D6.3 [22]. 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 dif-

ferent 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 property in the ProductSection SCALE_UP_NUM_NODES and SCALE_DOWN_NUM_NODES for scaling up and down.
- %-scale: Scaling a percentage of the currently deployed nodes. This is specified by the property 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. It can be "random", "lastly" or "balancer".

- 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. In the gLite service deployment and scalability, the Compute Element provides a service with a set of operations including the request for the node to be undeployed.

Finally, it is possible to create a "Lazy scale down", indicating that Claudia should wait a concrete time for starting with scaling down. This is specified by the property in the ProductSection SCALE_DOWN_LAZY_TIME.

5 Network Manager

StratusLab is planning to collaborate with the Mantychore project [13] to provide network services. However, this document is focused on the network management capabilities of OpenNebula, as the StratusLab Virtual Machine Manager.

5.1 Network Configuration

Network interfaces are configured using the contextualization capabilities provided by OpenNebula. However, DHCP is used on most data centres and, despite OpenNebula trying to replace all DHCP functionality, system administrators want a single point of administration, mainly to avoid inconsistencies. Currently, in StratusLab sites using DHCP, the default DHCP server is configured to assign statically IP addresses corresponding to predictable MAC addresses, and OpenNebula is configured to assign IP and MAC addresses matching the DHCP configuration. This procedure should be revised to avoid inconsistencies between OpenNebula and the DHPC server, and to reduce the configuration effort.

5.2 Network Isolation

When a new VM is launched, OpenNebula will connect its network interfaces (defined in the NIC section of the VM template) to the bridge specified in the Virtual Network definition. This will allow the VM to have access to different public or private networks. 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 will be provided through the OpenNebula Network Manager and its drivers.

For example, OpenNebula will provide support for host-managed VLANs to restrict network access through VLAN tagging. This mechanism is compliant with the IEEE 802.1Q standard [11], but it requires support from the hardware switches. Also, OpenNebula will allow administrators to restrict network access with Open vSwitch [16], a production quality, multilayer virtual switch.

5.3 Firewall Management

OpenNebula will provide support for enabling simple firewalling rules to allow a regular user to filter TCP, UDP or ICMP traffic. Also, it will be possible to restrict

network access through ebtables [8] rules. The ebtables program is a filtering tool for a Linux-based bridging firewall. It enables transparent filtering of network traffic passing through a Linux bridge. No special hardware configuration is required.

6 Storage Manager

There are several components to manage storage in StratusLab. The StratusLab Marketplace has been developed in WP2 to store metadata about system images for StratusLab sites. On the other hand, the StratusLab Persistent Disk Service has been developed in WP5 to persistently store user data and system images. This document focuses on the OpenNebula Image Manager, which allows OpenNebula administrators and users to set up images to be used in VMs easily, and describes its integration with the StratusLab Marketplace and with the Persistent Disk Service.

6.1 Storage Management in OpenNebula

OpenNebula manages three different types of images:

- **System images** contain a working operating system.
- **CD-ROM images** are read-only data.
- **Datablock images** are storage for data, which can be accessed and modified from different VM.

These images can be created from an existing file, but for datablock images it is possible to specify a size and file system type and let OpenNebula create an empty image. An system image has to be created from a contextualized VM by extracting its disk. Once a VM is deployed, it is possible save the changes made to any disk as a new image.

The Image Manager is in charge of storing and managing registered images. For example, when a new image is registered, its file is copied or downloaded to the front-end. The way the Image Manager accomplishes these tasks is through a driver that knows how to perform the following actions:

- `cp`: copy a image into the repository.
- `mkfs`: creates a new empty image initialized with a file system.
- `mv`: put a new version of an image into the repository.
- `rm`: deletes an image from the repository.

All these actions have an associated script that knows how to perform the action. By default, OpenNebula comes with a set of scripts that know how to store the images in the file system. However, this pluggable architecture allows the integration of any storage backend. Therefore, the Image Manager will be further integrated with the StratusLab Marketplace and the Persistent Disk Service as well as with external image catalogs, as will be shown in Section 8.2.1.

Another important characteristic of the OpenNebula Image Manager is its interaction with disks of running VMs. This requires both the extension of some functionality of the VM manager to generate snapshots and to hot-plug volumes to a running VM, and the integration of these features with the Image Manager. In this way, a VM disk could be stored back to the image repository once the VM is shutdown, or a snapshot of the disk can be made without stopping the VM, storing it to be used in other VMs or to back up the current one.

7 Monitoring and Accounting

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 metrics. In addition the accounting systems exist for keeping track of the amount of computing resources consumed per user over a period of time.

Monitoring and accounting are components that interact with the rest of components in StratusLab. It is possible to obtain metrics from physical hosts, storage and networking, virtual hardware information from virtual machines and metrics for services like KPIs. This means that any component in StratusLab can report monitoring information, by the usage of probes, to the collector service. This monitoring information can be taken from the customer to other components to check about the resource status. For instance, Claudia can use monitoring information for the service optimization, so that, it can scale up or down VMs according to users' policies. The accounting component can take this information for the service usage. This information also can be used for resource scheduling and os on.

On the other hand, accounting information is going to be consumed by different components for different objectives: to check the user's resource consumption for billing, to check if the user can deploy more services, and so on.

7.1 Monitoring

Every distributed system needs to incorporate monitoring mechanisms in order be able to constantly check the performance of the system. This is especially true in service clouds that are governed by Service Level Agreements (SLAs) and so the system needs to be able to constantly check that the performance adheres to the agreed terms. Monitoring data can also be used in a variety of ways, concretely for service optimization to scale up or down VMs according to users' policies, accounting for the service usage, resource scheduling and os on.

Different types of monitoring information are required:

Physical Infrastructure At this level, we must be able to monitor the usage of the physical bare-metal infrastructure, that is, the machines that are the VMs containers.

Virtual Hardware monitoring data CPU, memory, disk or network input/output

from the different deployed Virtual Machines should be collected.

Key Performance Indicator Performance estimation of the deployed service and verification of the service's status. Typical service KPIs like response time and availability should be measured, as well as, concrete service KPIs which depend on the application.

Software metrics Characterize the installed software.

7.1.1 Monitoring probes

There are several monitoring probes frameworks in the market that provide the monitoring functionalities.

Nagios [2] is an open source network monitoring system used to monitor services and hosts. With Nagios, the user is able to define monitoring parameters together with a set of thresholds. Nagios is designed more to check the service features and to compare with some established thresholds for providing alarms; it is less focused on collecting monitoring data.

A package, which is focused more on collecting monitoring data, can be Ganglia. Ganglia [14] is a distributed monitoring system for high-performance computing systems such as clusters and Grids. In Ganglia each node monitors itself and propagates the information to either all other nodes or to a subset of the nodes, if a hierarchical design is used. The system uses common technologies such as XML for data representation, XDR for data transport and the Round Robin Database tool (RRDtool) [17] for data storage. This information could be used to feed OpenNebula with monitoring information about physical and virtual machines, instead of the current SSH probes. Ganglia meets the requirements for physical infrastructure monitoring for the purposes of StratusLab. Ganglia is currently in use on over 500 clusters around the world and can scale to handle clusters with more than 2000 nodes.

Another monitoring probe framework is collectd. It is simple solution, since it is only collects data and does not work with thresholds and alarms. It can be combined with Nagios to provide the alarms functionality. It seems to be a good candidate for measuring KPIs or platform metrics. In fact, it is flexible, since it collects data by using a set of plugins and its results can be provided to RRD or other output formats. In order to measure application servers and java applications, we can look at the Java Management eXtension (JMX). It is used to monitor applications that have MBeans (Managed Bean) interfaces. The MBeans constitute the probe and are executed in the virtual machine where the application is hosted. The information provided by MBeans can be integrated with collectd via a plugin or as part of jcollectd. There are MBeans for Tomcat, JBoss and Glassfier among others.

7.1.2 Architecture

The monitoring system needs to be as flexible as possible, due to the large variety of metrics to measure, for example, infrastructure, product metrics, applications

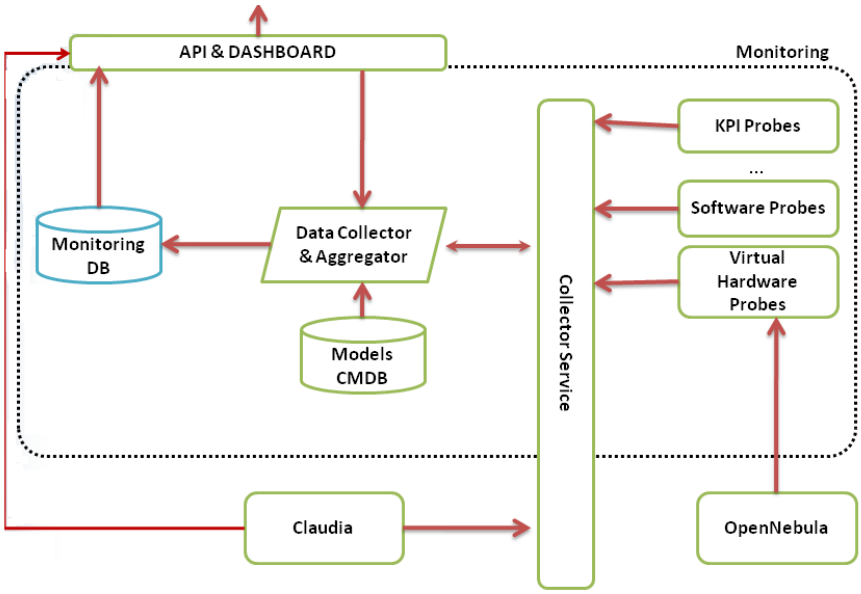

Figure 7.1: *Monitoring Systems*

metrics, and KPIs. The idea is not to extend a current monitoring framework or use a concrete probe system, but to design a new framework which can use all different monitoring systems required (e.g. collectd, mBeanCmd, etc.).

Probes provided by software providers can be included to monitor their applications. The metrics collected can be collected at the monitoring system layer, that is, collected at collectd or ganglia layer, but they are going to be aggregated at the virtual appliance layer.

A service must take into consideration the set of virtual machines which constitute it, the networks and the storage support. This aggregated information should consider all this parameters and the aggregation rules will depend on the service features.

Scalability is another major issue. The monitoring framework needs to be able to scale to large numbers of monitored nodes. This implies that a centralized collectd system should be avoided.

Finally, all historic monitoring data should be stored in a database accessible by an API or dashboard. From the previous analysis, we can draft a version of the monitoring framework; the design is in Figure 7.1.

Probes The software, installed in the virtual or physical machine, that is able to capture metrics. Metrics coming from probes installed in the virtual machine include infrastructure metrics as well as service metrics. These probes can be software like collectd, mBeanCmd, Ganglia, etc. In StratusLab, new probes or ones from third parties like collectd or Ganglia are going to be extended to send information to the collector service.

Collector Service Responsible for collecting all the metrics obtained from probes. Besides monitoring systems, another StratusLab services (e.g. accounting) can obtain the metrics' values from this collector service. The collector service is a new service in StratusLab to be developed in year 2.

Data Collector and Aggregator Responsible for obtaining the monitored values from the Collector Service, filtering them according to some policies, aggregating the information following the data model and storing it in the monitoring database. An initial prototype will be adapted to gather the monitoring information from the collector service.

Monitoring Database Stores all the metrics provided by the Data Collector and Aggregator. These metrics are organized according to the data model in the CMDB; that is, the Monitoring Database will include also aggregated information at service layer. The monitoring database can be based on MySQL or PostgreSQL.

API/Dashboard Allows any user or component access to monitoring information. The monitoring API is TCloud (see Section 3.1), and its implementation will be based on the tcloud-server with an *ad hoc* driver for accessing the monitoring systems.

7.2 Accounting

Accounting is a fundamental aspect of the provisioning of computing resources, regardless of the type of infrastructure (cloud, grid, etc.). In a nutshell, an accounting system is responsible to keeping track of the amount of computing resources consumed per user over a period of time. This information can be used for various purposes including resource billing and the enforcement of fair share policies from the resource provider.

7.2.1 Grid infrastructure accounting

Grid sites already implement their own services for keeping various accounting information. Currently APEL [1] and DGAS [6] are the two accounting solutions supported by the EMI software platform and can be enabled in an EGI grid site. These accounting tools collect information related to resource consumption from grid users. The focus of accounting record-keeping is solely placed on information regarding grid jobs. Currently there is no support for grid data management resource accounting.

The accounting information in grids are currently used primarily for enforcing Operational Level Agreements (a variation of SLAs) to grid sites that in turn ensure a certain quality of service offered by a grid infrastructure. Other than that, they help create a more complete picture about resource requirements per user or VO and are used for future planning and negotiations with resource centers. From

the technical point of view accounting data can also be important for building self-configuration capabilities in the grid middleware. Activities like the Grid Observatory [10] are working on the systematic collection of grid usage data with the purpose of building ontology for grid domain knowledge.

7.2.2 Cloud infrastructure accounting with OpenNebula

StratusLab relies on OpenNebula for the provision of core cloud services. Thus OpenNebula's accounting facilities play a fundamental role in the accounting services of the StratusLab cloud architecture. OpenNebula will provide an accounting module to track the information of resource utilization that will be stored in predefined intervals of time.

In addition, OpenNebula will provide historical monitoring information through a statistics module that will keep a predefined number of samples containing statistics for Hosts and Virtual Machines. These samples are built from the information that is retrieved from each resource by the OpenNebula Information Manager.

7.2.3 Accounting stakeholders

We can identify four different entities:

Grid user The end user of the grid infrastructure. A grid user submits jobs to a Workload Management System comprised of a central resource scheduler (WMS) and a set of distributed Computing Elements that act as local schedulers for dispatching job requests to a set of Worker Node machines. A grid user also exploits the storage management capabilities of the grid materialized primarily by a set of Storage Elements and an LFC service that keeps track of files and their replicas in the SEs. A grid user belongs to one or more Virtual Organizations.

VO manager The representative of a community of grid users (the Virtual Organization) that collaborate in the context of a certain domain to solve similar problems. A VO requires a number of computing resources to achieve this goal. The VO manager typical will negotiated with a number of resource centers in order to support the VO by allocating a percentage of their resources and make them available for the VO grid users.

Grid site manager Manages a grid site by providing the necessary technical support but also being the main contact point for interaction with the central grid authorities (e.g. EGI.eu) and the VO managers. A grid site may choose to support a number of different VOs based on their locality, their scientific domain and the overall utilization of the site's resources.

Cloud provider Manages a set of physical resources and provides IaaS capabilities to a set of users. These users may reside within the same administrative domain (private clouds) or can be third parties having remote access to the virtualized resources (public clouds).

From the point of view of the cloud, the grid site manager is end user of the cloud resources and the entity for which accounting information need to be kept. Cloud usage quotas and resource usage restrictions are enforced on the grid site level. Therefore these restrictions are transparent to grid users. For what concerns VO requirements for cloud resources, this can be negotiated between the cloud provider and the VO manager but the final allocation should be done through a site manager. Accounting information will be available per VO but usage quota will not be applied on a VO level.

7.2.4 Integration of Cloud and Grid accounting information

Clouds and grids reside in two distinctive layers of the StratusLab architecture. In principle the hosting of grid services is independent of the underlying infrastructure regardless if this is a set of physical machines or virtualized resources within a public or private cloud environment. Thus a reasonable approach is to consider grid services as a platform sitting on top of an IaaS cloud, providing specialized abstractions of computing resources and respective access protocols. For example the computing unit of a grid service is a job that encapsulates requirements and restrictions alongside with an application and its data dependencies.

In essence the only difference when running grid sites on clouds is that the latter enforce a certain business model. This business model has to be made visible on the grid service layer, or generally speaking on the platform service layer. On the other hand grids rely on a resource delegation mechanism; the grid site manager allocates a subset of resources from a physical cluster which can be used by any members of a given VO. For the grid resource provider to be able to cope with the cloud business model, the provider should be able to correlate grid resource usage data with cloud resource usage data. This is in essence the requirement that accounting integration between the two layers has to satisfy.

During the first year of the project, we extensively experimented with the deployment and provision of grid services running on top of our IaaS cloud distribution. One of the goals has been to identify the level of integration required in order to facilitate resource accounting between the two layers. Our conclusion is that for the time being we should aim for a modest level of integration based on the support of common information exchange standards.

Based on the interaction we had with other DCI projects and in particular during the EGI Cloud Computing Workshop [9], most opinions currently converge that the accounting problem is still open and that a good candidate standard in order to pursue interoperability is OGF's Usage Record (UR) [25]. OGF UR defines a common file interchange format for the collection, recording and transfer of accounting information among grid sites. The standard is targeted for infrastructures that offer job processing capabilities thus the job is the core entity for which information are kept. If we consider that the lifecycle of a VM resembles that of a job execution it is possible to reuse the same standard for keeping and exchanging cloud computing accounting records. This approach appears to be the best choice at the moment and will enable cloud resource providers to exchange information

data with large scale grid infrastructures like EGI. Still this is not an optimal solution and we expect that either the standard will involve in the coming months in order to incorporate explicitly cloud accounting data or to stimulate the formation of a new standard for such purpose.

8 Inter-Cloud Connector

In a Multi-Cloud scenario, multiple external and internal cloud computing services are deployed and managed to match business needs. In StratusLab this is implemented by the Inter-cloud Connector. The Inter-cloud Connector allows instantiation of VMs on public clouds (like Amazon EC2 or Flexiscale) as well as partner clouds (like other StratusLab sites) [21]. The component will abstract the external clouds' APIs by providing a common interface to them that can be plugged into the VM Manager component allowing a hybrid cloud approach (where the VM manager decides to outsource computation to another cloud) or the Service Manager component with a cloud brokering approach (when the VM is deployed in different cloud providers according to users' policies).

8.1 Multi-Cloud Scenarios

Stratuslab is considering three scenarios with multiple clouds: cloud bursting to a public Cloud, federation among two equal partners clouds, such as different StratusLab sites, and brokering.

8.1.1 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.

8.1.2 Cloud Federation

Cloud federation is very similar to Cloud bursting, 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.

8.1.3 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 [5]. 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

This scenario can be implemented by the Inter Cloud Broker provided by Claudia.

8.2 Technology

8.2.1 Cloud Bursting with OpenNebula

OpenNebula currently provides support for building Hybrid Clouds with Amazon EC2, ElasticHosts and RedHat Deltacloud. OpenNebula leverages its Deltacloud adaptor to access any major Public Cloud, such as GoGrid, Rackspace, Terremark or RimuHosting, and Private Clouds running OpenNebula or RHEV-M.

However, each provider uses different formats to store images. Fortunately, as described in Section 6, the OpenNebula Image Manager shows a pluggable architecture that allows the integration of any storage backend. In particular, it can be integrated with external image catalogs, like Amazon S3, so it would be possible to download, contextualize and integrate S3 images in a local image repository.

Finally, 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.

8.2.2 Cloud Federation with OpenNebula

The technology to perform Cloud Federation is very similar to the previous case. For computing, it is possible to access other OpenNebula instances using the Deltacloud adaptor. For storage, there is no need to import or export images, but just to share the same Markeplace service. For networking, the solution based on VPNs is still suitable.

8.2.3 Inter Cloud Broker with Claudia

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 to be added to Claudia. This is placement decision module, which drives the placement. This module is based on a set of business rules which defines the user's policies. Depending of these rules, the services and virtual machines will be deployed in a particular cloud infrastructure.

To sum up, a new model for Claudia called Placement is being developed inside StratusLab in order to decide in which providers deployed the service. On the order hand, more drivers to access to different Cloud providers are being developed, such as the Flexiscale one.

9 Summary

This document is an update of the document D6.1 including more StratusLab components and topics for year 2. The main contributions have been the inclusion of more cloud-like APIs for networking, storage and monitoring component, some information about networking and storage functionality, more work on monitoring and accounting, and the inclusion of the inter-cloud scenarios.

Concretely, this document has analyzed the inclusion of a service manager (called Claudia) inside the StratusLab architecture (on top of the OpenNebula) in order to manage the overall service instead of isolated virtual machines. In addition to provide a higher abstraction level to the service provider, it allows service providers to define the service's behavior in terms of service scalability. This service scalability is formalized in scalability rules and some scalability policies.

For the definition of this service behavior as well as the service, virtual machines and networks features, the document has analyzed the OVF standard. Furthermore, the usage of standards API has been identified as an important point to be included in StratusLab. Thus, TCloud and OCCI are the main choices for APIs to access to the Service Manager and Virtual Machine Manager. Moreover, jclouds has been included as a client library to access StratusLab due to customers' requirements. Considering storage, we are considering CDMI and OCCI, as well as jclouds. Finally, TCloud is chosen for monitoring information.

Networking and Storage have been analyzed in this document as being important functionalities identified in WP4. The networking functionality allows each user (or group) to create, manage and remove sets of isolated LANs (VLAN). Additionally it allows filtering of virtual machine traffic based on simple rules, e.g. TCP ports. Regarding storage, WP6 is working on an Image Manager system to set up images, which can be operating systems or data, to be used in Virtual Machines easily.

In order to evaluate the VM execution status, the monitoring mechanisms constantly check the performance of the system. In this document we have identified an architecture and some technologies for having metrics from different layers including physical, hardware information, KPI and software metrics. In addition, for keeping track of the amount of computing resources consumed per user over a period of time, OpenNebula will provide its own accounting system.

Finally, StratusLab has started to address inter-cloud scenarios in year 2, in order to make it possible the instantiation of VMs on public clouds (like Amazon

EC2 or Flexiscale) as well as partner clouds (other StratusLab sites) according to some user's policies.

The next steps for this work package will be the implementation of the functionality described in this document. Thus, D6.5 *Cloud-like Management of Grid Sites 2.0* will include information about the software implementation in order to achieve the functionality described in this document. In addition, D6.6 Second Year Cloud-like Management of Grid Sites Research Report will describe all the activities carried out for applying the functionality commented in this document to concrete use cases.

Glossary

Appliance	Virtual machine containing preconfigured software or services
Appliance Repository	Repository of existing appliances
CDDLM	Configuration Description, Deployment, and Lifecycle Management
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
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
SSD	Solution Deployment Descriptor
Virtual Machine / VM	Running and virtualized operating system
VMI	VM Manager Interface
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] APEL. Accounting Processor for Event Logs. Online resource. http://goc.grid.sinica.edu.tw/gocwiki/ApelHome.

[2] W. Barth. *Nagios: System and network monitoring*. No Starch Press San Francisco, CA, USA, 2008.

[3] 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.

[4] 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.

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

[6] DGAS. Distributed Grid Accounting System. Online resource. http://www.to.infn.it/dgas/index.html.

[7] 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.

[8] ebtables: Linux Ethernet Bridge Firewalling. Online resource. http://ebtables.sourceforge.net.

[9] EGI. EGI User Virtualization Workshop. Online resource. http://go.egi.eu/uvw1.

[10] Grid Observatory. Online resource. http://www.grid-observatory.org.

[11] IEEE 802.1: 802.1Q - Virtual LANs. Online resource. http://www.ieee802.org/1/pages/802.1Q.html.

[12] jclouds. Online resource, 2011. http://www.jclouds.org.

[13] Mantychore Project. Online resource, 2011. http://www.mantychore.eu.

[14] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[15] OCCI-WG. Open Cloud Computing Interface - Infrastructure. Technical report, Open Grid Forum, 2011. http://ogf.org/documents/GFD.184.pdf.

[16] Open vSwitch: An Open Virtual Switch. Online resource. http://openvswitch.org.

[17] RRDtool. Online resource. http://www.rrdtool.com/.

[18] SNIA. Cloud Data Management Interface (CDMI). Technical report, SNIA, 2011. http://www.snia.org/cdmi.

[19] Stratuslab Consortium. Stratuslab Description of Work, 2009.

[20] Stratuslab Consortium. Deliverable 6.1 Cloud-like Management of Grid Sites 1.0 Design Report. Online resource, 2010. http://stratuslab.eu/lib/exe/fetch.php/documents:stratuslab-d6.1-v1.0.pdf.

[21] Stratuslab Consortium. Deliverable 4.4 Reference Architecture for StratusLab Toolkit 2.0. Online resource, 2011. http://stratuslab.eu/lib/exe/fetch.php?media=documents:stratuslab-d4.4-v1.0.pdf.

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

[23] Telefónica I+D. Claudia Project. Online resource, 2010. http://claudia.morfeo-project.org/wiki/index.php/Main_Page.

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

[25] UR-WG. Usage Record - Format Recommendation. Technical report, Open Grid Forum, 2006. http://www.ogf.org/documents/GFD.98.pdf.

[26] VENUS-C Project. Online resource, 2011. http://www.venus-c.eu.

[27] VMware. vCloud API Programming Guide, Version 0.8.0. Online resource, 2009. http://communities.vmware.com/static/vcloudapi/vCloud_API_Programming_Guide_v0.8.pdf.