Enhancing Grid Infrastructures with
Virtualization and Cloud Technologies

Cloud-like Management of Grid Sites 1.0 Design Report

Deliverable D6.1 (V1.0)
16 November 2010

Abstract

This document presents the StratusLab architecture regarding automatic deployment and dynamic provision of grid services, as well as scalable cloud-like management of grid site resources. Some of the components included in the architecture provide solutions for the gaps identified in the deliverable D4.1 "Reference Architecture for StratusLab Toolkit 1.0.". Different alternatives were taken into consideration regarding cloud-like APIs, service definition language and contextualization, scalable cloud frameworks and monitoring and accounting solutions. Some chosen solutions include TCloud and OCCI API, OVF, OpenNebula contextualization and the Claudia framework

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

Document History

Version	Date	Comment
0.1	06 Oct. 2010	Initial version for comment.
0.2	15 Oct. 2010	TCloud, OVF, service scalability.
0.3	17 Oct. 2010	OCCI and contextualization in OpenNebula.
0.4	18 Oct. 2010	Monitoring and Accounting.
0.5	18 Oct. 2010	Introduction, Executive Summary and Conclusions.
0.6	22 Oct. 2010	Chapter revision.
0.7	22 Oct. 2010	Typographical, English, and formatting corrections.
0.8	5 Nov. 2010	Internal revision of the content.
0.9	11 Nov. 2010	External revision.
1.0	12 Nov. 2010	Final edition.

Contents

List of Figures

List of Tables

1 Executive Summary

The StratusLab project aims to provide system administrators and resource providers with the functionality that will enable the efficient exploitation of computing resources for the provision of grid services. StratusLab will capitalise on the inherent attributes of cloud computing and virtualization in order to offer grid administrators the ability to setup and manage flexible, scalable and fault tolerant grid sites. This way we will enable the optimal utilization of the resource provider's physical infrastructure and facilitate the day-to-day tasks of the grid site administrator.

The Joint Research Activity (JRA), carried out in WP6, conducts research 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: 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 for grid services and iii) the inclusion of novel resource provisioning models based on cloud-like interfaces.

Several research lines were opened to solve the gaps identified in the deriverable D4.1 "Reference Architecture for StratusLab Toolkit 1.0.". Different alternatives were taken into consideration regarding cloud-like APIs, service definition language and contextualization, scalable cloud frameworks and monitoring and accounting solutions. Some chosen solutions include TCloud and OCCI API, OVF, OpenNebula contextualization and the Claudia framework.

Claudia 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 (not just virtual machines), controlling the configuration of multiple Virtual Machine (VM) components, virtual networks and storage support by optimizing the use of them and by dynamically scaling up/down services applying elasticity rules, Service Level Agreements (SLAs) and business rules.

TCloud released by Telefónica and submitted to the DMTF and OCCI from the Open Grid Forum (OGF), are two main initiatives towards cloud computing management, which can be used in StratusLab as interfaces for the StratusLab components in the architecture.

As for the service definition language, Open Virtualization Format (OVF) was chosen, since it provides a standard way for describing service and virtual ma-

chines. a DTMF standard. Once VMs are installed, they need to be contextualized, i.e. to be passed configuration data at boot time. Currently, there are no standard ways to perform this step so far, but we will introduce some mechanism from OpenNebula and some OVF recommendations (for instance ISO images).

In 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) and dual-boot, 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 timely repurposed on demand for additional capacity.

2 Introduction

The Joint Research Activity (JRA) activity, carried out in WP6, conducts research 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 [21]: 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 D4.1, *Reference Architecture for StratusLab Toolkit 1.0* [23], where a StratusLab reference architecture was defined from a set of users requirements collected in D2.1 *Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures* [22]. D4.1 document identified a set of existing gaps in the current infrastructure, which are taken as a starting point for the work to be done in the JRA activity.

Thus, the main activity and change of this document with respect to the work done in D4.1 [23] is the inclusion of the service manager, which allows service providers (i.e. grid service providers) to deploy their service in a simple and efficient way, to dynamically control the service provisioning and to specify scalability constrains in the IaaS cloud. Concretely, it involves the provision of all those components needed to:

1. solve the technological gaps identified in WP4 [23].

2. provide technology to meet the requirements analyzed in D2.1 "Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures" [22].

2.1 Solution of Gaps Identified by WP4

Table 2.1 presents the gaps identified in D4.1 [23] together with a set of solutions which will be discussed and analyzed in the following chapters.

As commented in D4.1 [23], there is still a gap in order to deploy a single IaaS cloud distribution, accessible from outside the infrastructure, over a well-defined service API. Because of this, the research activity works towards the use of standardized APIs for resource, VM and service management. These APIs will allow

Table 2.1: *Identified Gaps and Solutions*

Gaps	Description	Solution
APIs	Cloud like API	TCloud, OCCI
	Integration of cloud and grid layers	e.g. grid certificates
VM Creation and Contextualization	Contextualization standards Interoperability between sites	OVF and OpenNebula contextualization
Scalability and reconfiguration	Scaling actions SLA-powered services	Claudia (Service Manager)
Monitoring	Grid monitoring services	Ganglia
Accounting	Grid accounting services	

the service manager and the virtual machine manager to be exposed as services and to be remotely accessible. Chapter 3 will discuss the use of standardized APIs inside StratusLab architecture, primarily TCloud and OCCI.

It is important to guarantee the interoperability among sites in order to reuse running virtual images. Thus, a standard definition of services and virtual machines as well as a standard VM contextualization mechanism are a must. Chapters 4 and 5 work towards the construction of standards in the contextualization and the definition of virtual machines and services. On the one side, Chapter 4 will introduce the Open Virtualization Format (OVF) for the definition of services and virtual machines. On the other side, StratusLab has defined a contextualization mechanism that is implemented with functionality provided in OpenNebula (Chapter 5), and that will guarantee virtual machine interoperability between sites.

Another key goal of StratusLab is the extension of currently available service-level open-source elasticity frameworks on top of cloud infrastructures to provide the required expressivity of current scientific applications including service elasticity rules for scaling up and down to meet SLAs. Thus, Chapter 6 will describe an existing open source service manager called Claudia and how it is used for service scaling.

The project will also evaluate the applicability of existing grid monitoring and accounting tools in the context of grid-over-cloud infrastructures and will identify their shortcomings (e.g. not being able to monitor the physical resources, since running on the cloud layer). The results are shown on Chapter 7.

2.2 Requirements from Grid Service Providers

For the design of the architecture, it is also important to obtain requirements from the final users. Some of these requirements were analyzed in D2.1 [22] and can be summed up as:

- *The StratusLab distribution must allow both full-virtualization and para-virtualization to be used.* It was already explained in D4.1 [23] and this document continues with the same idea, the usage of hypervisors, running on top of the physical resources (network, storage and compute) to provide virtualization. The most popular hypervisors are KVM, Xen and VMWare.

- *The StratusLab distribution must be simple enough for users themselves to configure their own resources as a cloud.* This private cloud appeals to members of research communities. In this matter, StratusLab works towards simplifying the deployment of grid service applications into the cloud, or a grid site, etc, with the inclusion of the Service Manager.

- *The cloud service must have a command line interface and a programmable API.* StratusLab provides a set of command line tools [23] and an API to access cloud services. Surveys show that a command line interface is preferred by both users and administrators.

- *The cloud distribution must allow a broad range of grid and standard services to be run.* Ranging from standard sites to grid services and storage services. The latter should be fast enough in order to be useful, so some kind of performance evaluation must be carried out.

- *The cloud distribution must allow dynamic service provision.* Users demand services, not just VMs, therefore service-level management is a must. Apart from that, service requirements vary and the cloud framework must fit the needs at each particular moment. This can only be accomplished via a wide range of scalability mechanisms in connection with a flexible monitoring system.

2.3 Organization of Following Chapters

The document is organized as follows: Chapter 2 provided a high level view of the tasks to be done in the research activities (WP6) as well as a set of research objectives. The remainder of the document is organized according to those objectives. The first one involves the use of standardized APIs to access cloud services (Chapter 3). The use of such APIs improves the interoperability as well as the usage of standards to formalize service definition and virtual machine contextualization as Chapters 4 and 5 identify. In addition, Chapter 6 presents the inclusion of a service manager (called Claudia) inside the StratusLab architecture in order to manage the overall service instead of isolated virtual machines and provide service scalability and Chapter 7 introduces the usage of monitoring and accounting in StratusLab. Finally, Chapter 8 presents the StratusLab architecture and Chapter 9 some conclusions obtained in the document.

3 Cloud-like Application Programming Interfaces

StratusLab tries to provide grid users with a homogeneous computing environment that simplifies applications management and hides the infrastructure 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 line, StratusLab works towards the use of cloud-like Application Programming Interfaces (APIs) for managing cloud computing capabilities including resource sharing. That is, service providers (e.g. grid users) can use programmatic APIs to access to the shared resources in order to manage them. Thus, the service providers (the cloud/grid client entity) requests resources from the infrastructure providers or IT vendors to deploy the services and virtual machines.

Considering the abstraction layers which are included in StratusLab, two kinds of APIs should be considered:

- **The Service Manager Interface** (SMI) is the API for the service manager and the access point for service providers.
- **Virtual Manager Interface** (VMI) is the API for accessing to the virtual machine manager, (OpenNebula for StratusLab). It hides the inherent underlying heterogeneity existing in cloud infrastructure providers from the service manager.

StratusLab is focused on the use of standards in the APIs definition, so that, the software implementation can be interoperable with other providers, avoiding undesirable vendor lock-in. Next sections are going to analyze some APIs alternatives that can fit with StratusLab requirements at least in a initial state. This analysis activity is done inside the two main initiatives for the cloud API standardization: i) Distributed Management Task Force (DMTF) [26] as a key player in the business arena and ii) Open Grid Forum (OGF) [19] as one of the key players in global standardization in academia.

3.1 Cloud-like APIs Alternatives

In the Cloud-like APIS analysis, firstly, this document is focusedc on the work that is being done by the DMTF. The DMTF is the leading industry organization for the development, adoption and promotion of interoperable management standards and technologies for IT systems and infrastructures. Inside the DMTF, the Cloud Management Working Group (CMWG) is developing a set of prescriptive specifications that deliver architectural semantics as well as implementation details to achieve interoperable management of clouds between service requestors/developers and providers.

For the elaboration of this specifications, the CMWG is taking into account a set of APIs provided by several leading companies of the industry. Among them we can mention vCloud from VMWare, HP Insight Dynamics API from HP, Fujitsu IaaS API from Fujitsu, TCloud API from Telefónica, etc. Subsections 3.1.1 and 3.1.2 will discuss the vCloud and TCloud APIs as candidates for StratusLab.

On the other hand, the document describes the Open Cloud Computing Initiative (OCCI). The OGFs OCCI (Open Cloud Computing Initiative) Working Group is aiming to deliver an API specification for remote management of cloud computing IaaS. The proliferation of IaaS providers with their own APIs (Amazon AWS, GoGrid, Flexiscale, etc.) requires an standardization effort on defining an common API to avoid vendor-locking and to guarantee interoperability. Subsection 3.1.3 will explain the advantage of the OCCI approach.

3.1.1 vCloud from VMWare

vCloud [28] is an initiative led by VMware that counts with the collaboration of more than 100 partner including BT, Rackspace, SAVVIS, Sungard, T-Systems and Verizon Business. The main aim of vCloud is to provide a complete solution to deliver enterprise cloud services based on VMware technologies. The vCloud API will allow application providers to deploy their application among a cloud service provided by one of the vCloud partners. The vCloud API is based on the principles of Representational State Transfer (REST).

The vCloud resource model includes the following entities: the Organization that owns the applications, the Virtual Data Center (VDC) where different Resource Entities (virtual applications templates, media devices, etc.) and Networks are instantiated. Finally, Virtual Application or Appliance (VApp) is an aggregation of one or more virtual machines that work together for the consecution of the same service.

There are number of operations to query existing resources or summary of the Organizations resources and to delete an specific resource from a VApp. All these operations are based on typical HTTP operators (GET, POST, PUT, DELETE) and includes the location of the REST-oriented resource, through its URI.

vCloud application lifecycle model defines a number of steps where a number of operations can be performed:

- Provisioning: creation or clonning of VApps, instantiation of the VApp templates, upload or facilitate the location of VApp parts and instantiation of VApps defined in an OVF package (see section 4.3).
- Configuration: a client can reconfigure VApps by adding, removing or modifying OVF sections.
- Deployment: reservation of all the resources requested by the VApp.
- State operations: once the resources are reserved the VApp have to be started by powering on all the resources, and later on it can be powered off, reset, suspended, shutdown, rebooted, etc.

As some of the previous operations can take time, pending task can be retrieved and cancelled. There are some other administrative operations to manage Organizations, VDCs, users, etc.

As conclusion, we see that the basic service lifecycle management is very well covered, but advanced features like dynamic scalability, advance management capabilities, monitoring or usage accounting, etc. are not included.

3.1.2 TCloud from Telefónica

The TCloud API [24] 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 [28], published by VMware and submitted to the DMTF for consideration.

TCloud API defines a set of operations to perform actions over the following basic elements:

- Virtual Appliances (VApp). This resource represents, typically, a Virtual Machine running on top of a hypervisor, but TCloud leaves the vendor to determine the bounds of VApp concept.
- Virtual Data Centres (VDC). VApps resides in a VDC context. A VDC is a set of virtual resources (e.g. networks, computing capacities) which incarnate VApps.
- Organizations (Org). VDCs are owned by organizations. An organization represents any kind of independent unit which manages its own cloud resources (e.g. enterprises, divisions, groups).

TCloud API defines operations to perform actions over above resources categorized as follows:

- Self-Provisioning operations. This includes capabilities for instantiating VApps and VDC resources.

- Self-Management extensions. This includes capabilities to manage instantiated VApps. Some examples are adding more hardware to an existing VApp or shutting down a virtual machine.

TCloud API defines this set of core operations. More extensions may be applied by implementers to provide actions not considered in this specification. Along this core, TCloud defines the following extensions:

- Self-Monitoring operations. This includes capabilities for obtaining information of virtual resources used by VApps, VDCs or Organizations. Some examples are obtaining the amount of memory used by a VM or obtaining the used bandwidth for a VDC.

- Self-Administration extensions. This includes capabilities for managing organizations. Some examples are creating a new organization or adding more users to it.

TCloud is focused on adding network intelligence, reliability and security features to cloud computing empowered by enhanced telecom network integration [24]. Moreover, TCloud aims to extend current cloud computing models providing more flexibility and control to cloud computing customers. DMTF defines cloud computing as "an approach to delivering IT services that promises to be highly agile and lower costs for consumers, especially up-front costs". This approach impacts not only the way computing is used but also the technology and processes that are used to construct and manage IT within enterprises and service providers.

In essence, compatibility for the main operations and data types defined in vCloud are maintained in TCloud, but it provides extensions for advanced Cloud Computing management capabilities including additional shared storage for service data, network element provisioning (different flavors of load balancers and firewalls), monitoring, snapshot management, etc. All these extended capabilities will be very useful for covering the StratusLab objectives.

3.1.3 OCCI from OGF

The OCCI effort was initiated to provide a global, open, non-proprietary standard to define the 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.

The OpenNebula OCCI API is a RESTful service to create, control and monitor cloud resources based on the latest draft of the OGF OCCI API specification [18]. The following sections describe the current OCCI API implemented in OpenNebula 2.0.

3.1.3.1 Resources

There are two types of resources (see Figure 3.1) that resemble the basic entities managed by the OpenNebula system, namely:

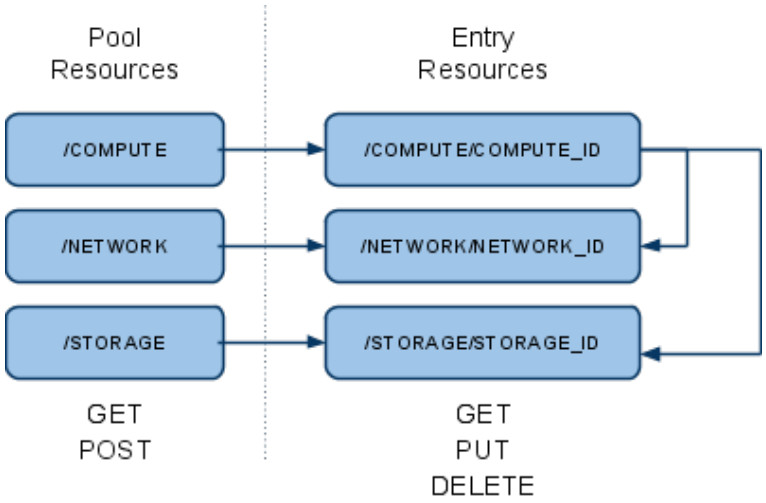

Figure 3.1: *OCCI resources and methods*

- Pool resources: Represents a collection of elements owned by a given user. In particular three pool resources are defined: COMPUTE_COLLECTION, NETWORK_COLLECTION and STORAGE_COLLECTION.
- Entry resources: Represents a single entry within a given collection: COMPUTE, NETWORK and STORAGE.

The NETWORK resource defines a virtual network that interconnects those COMPUTE elements with a network interface card attached to that network. The STORAGE resource defines a virtual disk that supports a VM block device. The COMPUTE resource defines a virtual machine by specifying its basic configuration attributes such as NIC or DISK, which include a NETWORK or STORAGE resource, respectively.

3.1.3.2 Operations

The methods associated with each resource type, as shown in Figure 3.1, are as follows:

- Pool resources:
  - GET: to list all the entry resources in that pool resource owned by the user.
  - POST: to create a new entry resource.
- Entry resources:
  - GET: to list the information associated with that resource
  - PUT: to update the resource (only supported by the COMPUTE resource).

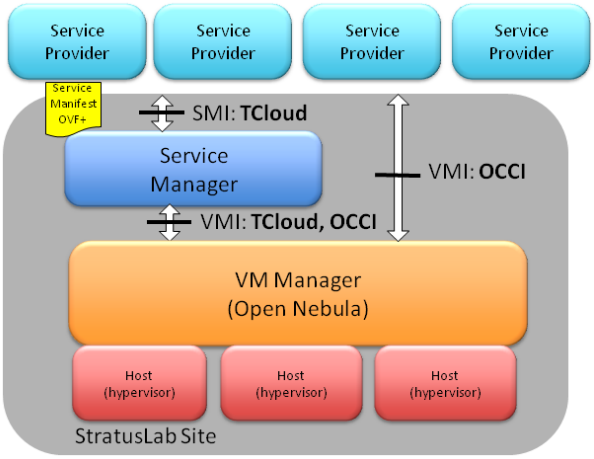

Figure 3.2: *StratusLab Cloud-like APIs*

- DELETE: to delete the resource.

The HTTP protocol does not provide means for notification, so this API relies on synchronous polling to find whether a VM update is successful or not.

Currently, the implementation uses HTTP Basic Access Authentication, therefore it is recommended that the server-client communication is performed over HTTPS to avoid sending user authentication information in plain text. Authorisation is handled by OpenNebula's user management module.

3.2 StratusLab Cloud-like APIs

Previous sections explained several candidates APIs to be adopted in the StratusLab project. Taking into account the conclusions, advantages and disadvantages described and their adaptation to StratusLab vision and requirements, StratusLab is going to use TCloud and OCCI as Cloud-like APIs. These two APIs have the same functionality (at the VMI level) and could be used without distinction. Thus, StratusLab decides to use both APIs to be aligned with the standardization work carrying out in the two main cloud standardization bodies: DMTF and OGF.

Figure 3.2 shows this double possibility. While, the SMI will be implemented by TCloud, the VMI will be able to be both TCloud and OCCI. In addition, StratusLab gives the opportunity to grid users to interact directly with the virtual machine manager by the direct usage of the VMI. In this second case, the OCCI standard will be used.

Thus, StratusLab offers two standard APIs for managing the services, virtual machines and resources. The current definition of both APIs will have to be extended to better cover all the StratusLab requirements but they represent a good start point for the project.

3.3 Grid and Cloud Integration

All services used by StratusLab must be integrated with a single identity system. OpenNebula has been already extended to support LDAP. It is important to ensure that all StratusLab services use such strategies to limit the propagation of specific credential mechanism across the distribution. Further, the grid uses a digital certificate mechanism to ensure proper integration of authentication and authorisation across its services and sites. StratusLab will need to integrate with this certificate-based mechanism in order to properly support the grid.

Another interesting solution with which we must consider is provided by the ARGUS [6] project, which develops a solution used in the grid world to provide authorisation decisions over distributed services.

OpenNebula comes with an internal user/password authentication and authorisation system and the ability to use an external driver that takes care of these duties. In this case, some support in the access APIs, e.g. OCCI and TCloud (see 3), could be needed, especially for authentication. Also, authorisation could be based on based on groups and group roles, using VOMS extensions.

The current authentication/authorisation module (from now on the "auth" module) has support for user/password and RSA private/public key authentication as well as user quota support. By default OpenNebula comes configured to use internal user/password authentication. Moreover, the auth module is designed so it can be easily modified or completely replaced, so it can be adjusted to a given need or to new authorisation methods, like grid certificates, including VOMS extensions.

Authorisation systems are classes with a method called `auth` defined. This method is called each time OpenNebula needs to authenticate a user. When OpenNebula is asked by a user to do something with one of its objects, an authorisation message with a series of tokens describing the objects and type of actions that will be performed. For an authorisation message to be successful, the user needs to have permissions to perform all the actions. When the action is permitted by the policy it should return true otherwise false or a string containing the reason for rejection is returned.

4 Virtual Appliance (Service) Language Definition

4.1 Virtual Appliance (Service) Language Description

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 a cloud environment, one key element of the interaction between Service Providers (SPs) and the infrastructure is the service definition mechanism. In IaaS clouds this is commonly specified by packaging the software stack (operating systems, middleware and service components) into one or more virtual machines, but one problem commonly mentioned [10] is that each cloud infrastructure provider has its own proprietary mechanism for service definition. This complicates interoperability between clouds and locks a service provider to a particular vendor.

For example, if a service provider has prepared a service to be deployed in Amazon EC2 (using the proprietary Amazon Machine Image format) then changing this service to another cloud IP (e.g., GoGrid) is not straightforward, as the image creation for each provider requires several different configuration steps. Therefore, there is a need for standardizing this virtual application distribution in order to avoid vendor lock-in and facilitate interoperability among IaaS clouds.

The selection of a service definition language is a key decision for the StratusLab project. Next section describes and compares several alternatives.

4.2 Virtual Appliance Language Alternatives

There are several standards to be used as the basis for the description of services to be deployed on IaaS clouds. This section enumerates the three most relevant ones.

Configuration Description, Deployment, and Lifecycle Management Specification (CDDLM) [1] is a format proposed by the Global Grid Forum (now part of the Open Grid Forum) as a standard for the deployment and configuration of grid services. CDDLM gives total control about the file format: tags names and content. Although the creation of ad-hoc tags allow cloud users to specify any requirement,

so that to be flexible, it forces to find an agreement about how to represent and interpret all the configuration parameters (such as hardware requirements, instance configuration, elasticity rules, KPIs, etc).

Another emerging standard (from OASIS) is the Solution Deployment Descriptor (SDD) [17]. SDD is oriented to the description of the management operations of software in the different stages of its typical lifecycle: installation, configuration, maintenance, etc. SDD does not address neither how the deployment is accomplished, nor how the configuration data must be interpreted. Thus, there must be a previous agreement about how the requisites and conditions are declared and interpreted. For example, the SDD proposal includes a Starter Profile [16] based on the definitions provided by the DMTF Common Information Model (CIM) [8] to promote interoperability. However, this format lacks 'native' features to describe the requirements closer to the infrastructure level (hardware specification).

Last but not least, the Open Virtualization Format (OVF) [9] is a Distributed Management Task Force (DMTF) [26] standard which specifies how to package and distribute software to be run in virtual machines. OVF enables the easy packaging of applications as a collection of virtual machines in a portable, secure and technology independent way [9]. This means that the application should be easily distributed in different data centers and interoperability problems are minimized.

Regarding the standards related with service specification, CDDLM and SDD have limitations when considered to be used in IaaS clouds, and by extension, in the StratusLab platform. CDDLM is not a deployment format itself, but a frame to define configuration formats. Given that the main goal of this work was to extend an already present standard, not creating a completely new format, it can be concluded that CDDLM does not suit our requirements. Regarding SDD, it is inherently flexible, but it forces service and infrastructure providers to reach an agreement on the deployment and data models before any deployment can be performed, while we intend the service description document to be self-contained, without the need of using third-party specifications. In addition, SDD has some expressiveness problems (in particular, to express infrastructure requirement).

The conclusion is that between CDDLM, SDD and OVF, the latter is the most appropriated choice to base a service specification for covering the StratusLab requirements.

4.3 Open Virtualization Format

Section 4.2 proposes several service definition languages and concludes that the best option for the StratusLab platform is the Open Virtualization Format (OVF) [9]. This section provides a high level view of OVF structure and objectives.

The Open Virtualization Format (OVF) objective 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 ven-

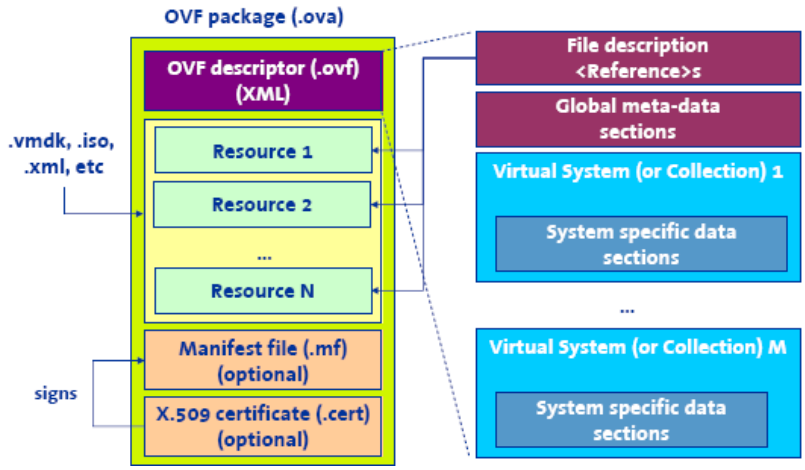

Figure 4.1: *OVF package and descriptor structure*

dor 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.

Figure 4.1 shows the structure of an OVF package. This file includes an OVF descriptor (an XML file describing the VApp), resources used by the VApp (virtual disk, ISO images, internationalization resources, etc.) and, finally, an optional manifest and X.509 certificate files to ensure integrity and authenticity. The OVF specification describes also the OVF environment (a guest-host communication protocol which is used by the deployment platform to provide configuration parameters to guests at deployment time). OVF defines some procedures to fill automatically these parameters during the boot process using the OVF Environment document and the Activation Engine script (procedures with some limitations and that is not completely standardized yet).

Thus, OVF specifies how to package and distribute software to be run in virtual machines. Our contribution is focused on utilizing OVF as basis for a service definition language for deploying complex Internet applications in a StratusLab grid-cloud infrastructure. These applications consist of a collection of virtual machines (VM) with several configuration parameters (e.g., hostnames, IP addresses and other application specific parameters) for software components (e.g., web server, application server, database, operating system) included in the VMs. Most of these parameters are unknown before the deployment time because the service provider does not know particularities of the IaaS cloud or datacenter in which the deployment takes place.

4.4 Open Virtualization Format in StratusLab

As already mentioned, current Infrastructure as a Service (IaaS) Clouds present their own mechanisms for service definition. This complicates interoperability among Clouds. In addition, the current service definition mechanisms have some limitations, mainly they only consider conventional IT infrastructure as the target deployment platform. Therefore, there is a need for standardizing a service definition language in order to avoid vendor lock-in and facilitate interoperability among IaaS Clouds. In order to solve this interoperability problem, our proposal is to base the service definition on open standards and extend it when needed. It is worth mentioning that we are not only proposing to use OVF as a way of solving the cloud interoperability problem. We are going a step further, addressing important issues for IaaS clouds

Standard OVF was designed considering conventional IT infrastructure (i.e. data centers) as the target deployment platform, so it presents some flaws when applied directly to service deployment in clouds [12] and by extension in a StratusLab infrastructure. There are several important issues that are not completely solved (or not considered at all) in the current OVF specification, such as:

- Self-configuration: how virtual machines composing the service are dynamically configured, e.g. IP addresses.
- Custom automatic elasticity: how service providers could specify rules and actions to automatically govern the scaling up and down of the service
- Performance monitoring: how service providers define the key performance indicators that are monitored by the cloud infrastructure, e.g. to trigger the elasticity actions.

One of the key activities in the StratusLab project will be to solve all these problems using extensions to the OVF standard to achieve StratusLab goals.

5 Virtual Machine Contextualization

Another key element is working towards the contextualization definition. Contextualization is defined as the process by which a virtual machine instance is configured based on virtual machine master image. In general, contextualization consists of passing arbitrary data to the virtual machine at boot time. The goal of this process is to configure generic installations of system services (e.g. sshd) and it is the basis for implementing a *"install once, deploy many"* strategy.

However, contextualisation currently lacks standardisation. While this is an aspect where StratusLab will engage the standardisation organisations over the project lifetime, we currently need to define conventions that will guarantee virtual machine interoperability between sites running the StratusLab distribution initially, and eventually to develop solutions for letting users and sites utilise other cloud services beyond the StratusLab frontier, including public and commercial clouds.

Instance level contextualization is based on the availability of data not contained in the base image but available in the instance. In its most basic form it is achieved via the use of a CD-ROM device, or attached disk. The CD-ROM content is dynamically created and mounted within the instance. This basic process is compliant with OVF standards and is used in OpenNebula.

In general, a VM consists of one or more disk images, which contain the operating system and any additional software or data required. When a new node is needed, the images are transferred (cloned) to a suitable physical resource and a new VM is booted. During the boot process the VM is contextualised, i.e. the base image is specialised to work in a given environment by, for example, setting up the network or the machine hostname. Different techniques are available to contextualise a VM, for example a context server [11], or accessing a disk image with the context data for the VM.

It is important for StratusLab to support standard image formats. We chose ISO images (OVF recommendation), also supported by OpenNebula, to provide configuration parameters to a newly started VM. This method is network agnostic, so it can be used also to configure network interfaces. In the VM description file, the user can specify the contents of the ISO image (files and directories), tell the device how the ISO image will be accessible, and specify the configuration parameters that will be written to a file for later use inside the VM.

Another issue that arose in StratusLab is the integration with DHCP for net-

work management. DHCP is used on most data centres and, despite OpenNebula tries 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.

6 Service Scalability

Cloud computing promises an easy way to use and access a large pool of virtualized resources (such as hardware, development platforms and/or services) that can be dynamically provisioned to adjust to a variable workload, allowing also for optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by means of customized SLAs [27].

Therefore, cloud computing automated provisioning mechanisms can help applications to scale up and down systems in a way that performance and economical concerns are balanced. 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)" [4]. The actions to scale may be classified in [4]:

Vertical scaling by adding more horsepower (more processors, memory, bandwidth, etc.) to equipment used by the systems. This is the way applications are deployed on large shared-memory servers.

Horizontal scaling by adding more of the same software or hardware resources. For example, in a typical two-layer service, more front-end nodes are added (or released) when the number of users and workload increases (decreases). This is the way applications are deployed on distributed servers.

As commented in the requirements in Chapter 2, StratusLab will work on VM *reconfiguration*. The vertical scaling can be considered as reconfiguration in the sense that the virtual machine features are changed. For us, the reconfiguration will imply stopping the current VM and deploying a new one with different VM features. StratusLab will also work on the horizontal scaling, so that, it will be able to scale up or down hardware and software resource instances in case they are required.

6.1 Service Scalability Framework Alternatives

Grid applications deployed over cloud technologies should benefit from scalability at service level, which conceals low level details from the user. There are several off-the-shelf alternatives in the industry when it comes to scalable cloud infrastructures: Amazon Web Services, GoGrid and RightScale are among the best known.

Although they are well-tested solutions, these alternatives exhibit some disadvantages:

1. Non-abstract, they deal with VM management instead of service management
2. Non-standard interfaces (which leads to poor portability) on top of a single infrastructure (which hinders federation with another clouds)
3. Lack of flexibility, in the sense of:
    - Reduced set of actions available (basically limited to VM launching/removing)
    - Impossibility of combining metrics (neither arithmetically nor logically)

Thus, in order to overcome with these limitations the Claudia project [25] arises being the chosen solution to be used in StratusLab.

6.2 Claudia

A means to solve those problems is to introduce a layer on top of current IaaS clouds that provides users with the required abstraction level and allows them to manage the service as a single entity, closer to user needs. This solution 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. Claudia follows such paradigm and is the only open source solution that meets these requirements.

The Claudia platform [25] 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 can deploy services in a public cloud (Amazon, Flexiscale, GoGrid, etc.) or in a private cloud using a Virtual Infrastructure Manager (such as OpenNebula, Eucalyptus, etc.) through a plug-in driver mechanism that will orchestrate the virtual resource allocation [25], as Figure 6.1 shows.

The main component in Claudia is Clotho, which involves the Service Lifecycle Manager (SLM) and the Scalability and Optimization Module presented in Figure 6.1. Clotho allow for creating services and managing monitoring events and scalability rules. It 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 business rules protection techniques.

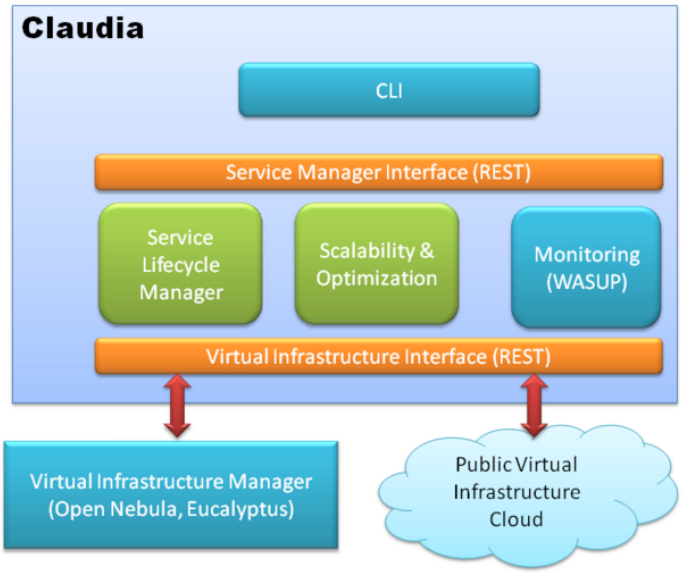

Figure 6.1: *Claudia High Level Architecture*

Clotho provides a means for users to specify their application behavior in terms of adding or removing more of the same software or hardware resources [4] by means of *elasticity rules* [5]. 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.

As Clotho APIs, Claudia uses the TCloud API (see 3.1.2) as a cloud-like API for both SMI and VMI. Thus, the TCloud Server component implements a subset of the TCloud API (see 3.1.2), acting as both as SMI and VMI.

WASUP, the monitoring system in Claudia, is the component that stores and distributes the status of the deployed services data to the rest of the service manager components and to the cloud users. This component aggregates the information collected by others monitoring systems, for instance, placed in the virtual machine. Still, it is required an analysis task to see if StratusLab monitoring tools offer the same functionality as WASUP. In this case, WASUP would be disable.

Finally, Claudia offer a command line client used to manage and to test the other components in the Claudia Client component. It integrates a simple TCloud API client for service deployment features and monitoring utilities.

6.3 Using Claudia in StratusLab

The service manager provides a more flexible and easier to use platform to deploy grid-oriented services with all the benefits of the IaaS cloud model. As it is stated in both scenarios, now grid service providers can define the configuration and dynamic behavior of their services in a easy and standard way. The SM hides platform complexity to the Service Providers, avoiding complicated management processes and allowing Service Providers to focus on the service definition.

Thus, this section tries to illustrate how the service manager will work in StratusLab with the rest of StratusLab components. Concretely, we are going to exemplify how the StratusLab architecture works when a service provider decides to use the StratusLab platform to deploy one of its services. Once this service is working, the rest of use cases in StratusLab can be deployed in the Final StratusLab architecture including the technology developed in WP6.

6.3.1 Deployment Scenario

Figure 6.2 shows a high level flow describing the deployment of the service into the StratusLab platform. The main steps are:

1. As a previous step before the beginning of the deployment process, the service provider must define the service structure, conditions and characteristics in a OVF descriptor. It includes the description of all the service's VMs (OS, hardware characteristics, networks, etc.).

2. In addition, the service provider has to generate all the VMs involved in the services and store them into the Appliance Repository.

3. At this point, the service provider is ready to request the service deployment. The interface through the service manager (SM) implements the TCloud API for supporting the needed operations at this level. The deployment request will include the OVF descriptor.

4. The SM revises the OVF descriptor in search of useful information for the service lifecycle management. The startup order of all the VMs of the Service, scalability rules, resource requirements, etc. are some examples of the available information within the OVF Descriptor. The SM also translates the OVF Descriptor into the needed OpenNebula templates (for both networks and VMs).

5. The next step is to provide the needed networks and to start the deployment of VMs in the right order, by interacting with OpenNebula.

6. At this point, the OpenNebula engine has all the information to deploy the service into the StratusLab Platform. It localizes the images in the Appliance Repository and installs them in the corresponding resources.

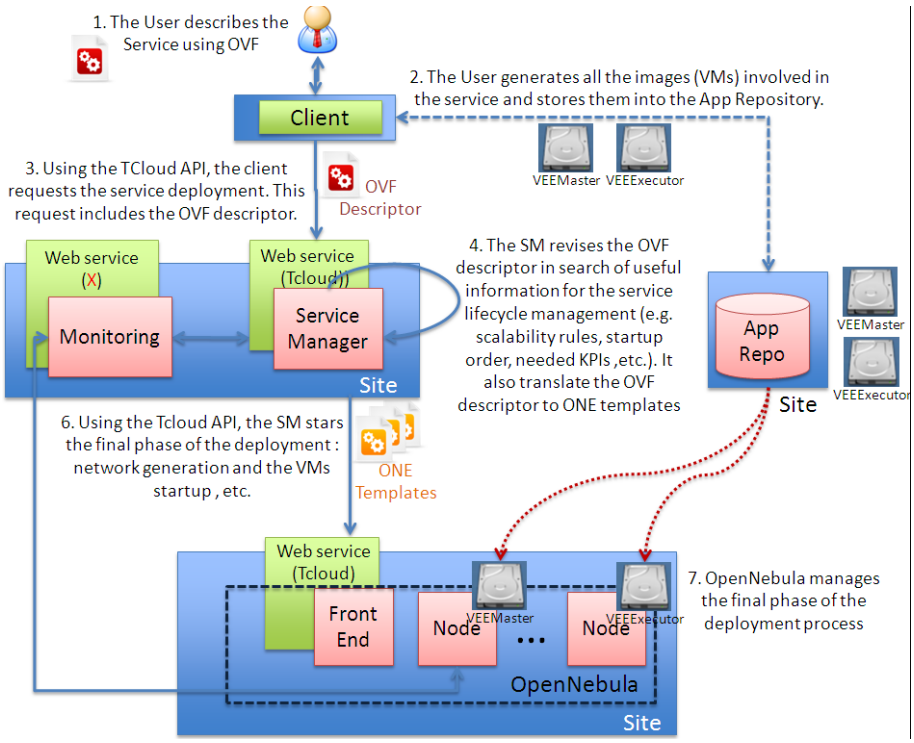

Figure 6.2: *Deployment Scenario*

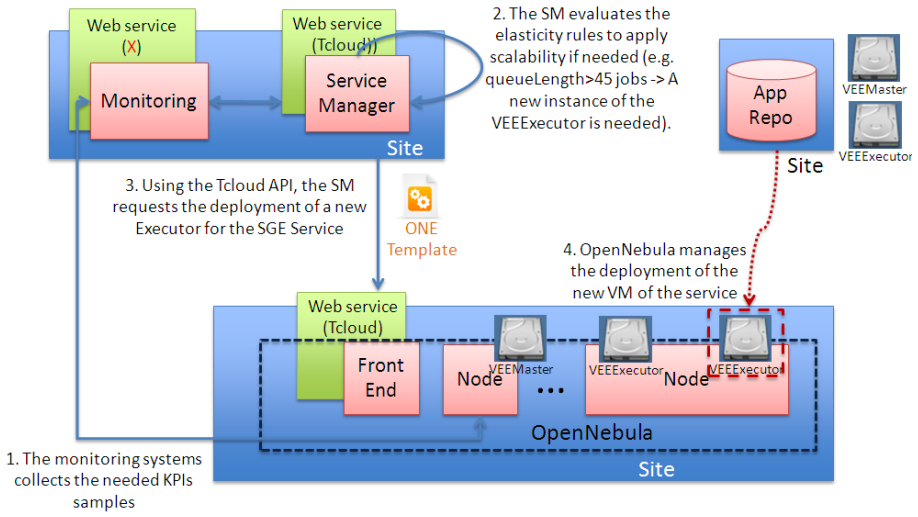

Figure 6.3: *Scalability Scenario*

6.3.2 Scalability Scenario

As commented previously, the service manager will provide grid service elasticity. Thus, service providers define elasticity rules in the service manifest to assure the right operation of their services and/or the final user experience. Moreover, the Service Provider has defined elasticity rules which adds a new instance of a concrete virtual machine to avoid the system overload.

Figure 6.3) shows how the StratusLab platform manages the scalability of the a service.

The main steps are:

1. The StratusLab Monitoring System (or a grid monitoring tool) provides KPIs values for the service to the SM.

2. The service manager evaluates, in real time, the elasticity rules of the service. When the corresponding KPI exceeds the given threshold, the SM starts the deployment process of a new VM instance.

3. Using the TCloud API, the SM requests for a new replica to be deployed. The steps here are the same as in the deployment Section 6.3.1.

6.4 Next steps in Claudia for StratusLab

Although Claudia is well-suited for Stratuslab as a service manager according to the requirements identified so far, Claudia follows a process of continuous growth and could be extended with new requirements identified. In contact with the com-

munity of users, we will analyze the differen grid services to be deployed and new grid users requirements which can appear with the usage of Claudia in StratusLab.

For instance, previous experience exists in the deployment and the scalability of the Sun Grid Engine application. However, more grid services, such as Torque or gLite grid services, will be used as use cases to be deployed in the StratusLab infrastructure by using the service manager, which for sure, will provide new requirements.

In addition, the service manager will need to be extended to consider the deployment and configuration of load balancers. So that, the service manager, automatically, can deploy a load balancer, which is part of the application to be deployed, configuring it in case new balanced replicas have been instantiated.

The integration of Claudia with management tools popular in the grid community (e.g. Quattor) through TCloud will be another subject of study to be done in nexts steps.

Finally, when there will be a decision in monitoring tools, Clotho will be extended to obtain the monitoring data from the chosen tools, or to populate WASUP database with the obtained data.

7 Grid Accounting and Monitoring

7.1 Monitoring

In this section, we talk about the monitoring modules that will be used for the purposes of the project. We consider three levels of monitoring, according to the infrastructure or service we are interested in:

Physical Infrastructure In this level, we must be able to monitor the usage of the physical bare-metal infrastructure, i.e., the machines that are the VM containers. A very popular monitoring tool that can easily collect information for arbitrarily large clusters is Ganglia [14] (other open-source alternatives include the Nagios [3] or the Munin [15] Monitoring Systems). Ganglia is a scalable distributed system monitor tool for high-performance computing systems. It allows the user to remotely view live or historical statistics (such as CPU load averages, network or memory or disk space utilization) for all machines that are being monitored. The presentation is done through a web front-end that provides a view of the gathered information via real-time dynamic web pages. Moreover, this information could be used to feed OpenNebula with monitoring information about physical 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.

Virtual Machines Infrastructure In this layer, the StratusLab administrator must be able to have an overview of the total virtualized resources that are currently deployed. The internal OpenNebula XML-RPC API provides extended monitoring of virtualized resources through some simple command-line utilities. These virtualized resources may include the total number of virtual machines that each actual node stores, the amount of allocated physical memory, the number of allocated physical CPU cores, or the bandwidth consumption per virtual machine. Currently, OpenNebula utilizes ssh commands that connect through the Cluster Front-End to each physical machine to collect virtual-machine related information which is then stored in the OpenNebula database. Instead of this, we could integrate OpenNebula with Ganglia, so that Ganglia collects virtual-machine related information and

disseminates this information back to OpenNebula. In the case where Ganglia integrates with OpenNebula, we will avoid connecting with each virtual machine container from the Cluster Front-End. Instead, virtual-machine related metrics will be locally fed to Ganglia through each virtual machine container, and Ganglia will disseminate this information to the Cluster Front-End in a scalable and efficient manner. What is more, in this situation, the cloud infrastructure administrators will have a unified GUI through the Ganglia web-based front end where they can monitor both the physical and its virtual machine infrastructure.

Grid Services In this level we must be able to monitor the Grid Sites that are deployed in the StratusLab infrastructure. Monitoring software that is used in actual grid sites can be re-used in this situation, since grid applications are unaware that they are being executed inside virtual machines.

7.2 Accounting

Accounting is a fundamental aspect for the provisioning of computing resources regardless the type of infrastructure (cloud, grid, etc). As such, it plays in important role for the provisioning of cloud computing resources in the context of StratusLab project. In a nutshell, an accounting subsystem is responsible for keeping track of the amount of computing resources consumed per user over a period of time. These information can facilitate among other things a fair share of computing resources among the users, through a provision mechanism that applies a set of accounting policies defined by the resource provider.

7.2.1 Cloud infrastructure accounting requirements

Cloud resource providers deploying the StratusLab distribution will need to keep track of usage for a diverse set of resources. A typical list of these resources is the following:

Number of Virtual Machines The number of VMs per user is the basic metric kept from a cloud accounting system.

Number of CPUs It is important to measure the total number of CPUs allocated at each user. The physical infrastructure has a finite number of actual CPUs per host machine, and their utilization must be carefully taken care of.

RAM size The total amount of RAM that is assigned to the user's virtual machines must be included in the accounting metrics and should its use must be regulated through the accounting policy.

Storage space The accounting module must be able to assign and "charge" users according to the storage space they will allocate.

Network I/O The accounting module must measure the Internet traffic that is generated by each user, i.e., the data transfer in and out of the StratusLab physical cluster data-center (traffic that is generated between StratusLab virtual machines should not be taken into account).

OpenNebula is implementing in release 2.0 an Accounting Toolset [20] that visualizes and reports resource usage data, and allows their integration with charge-back and billing platforms. This add-on can be used as the basis for the accounting module and will be further developed in order to potentially satisfy specific requirements stemming from the project.

7.2.2 Integration of Cloud and Grid accounting information

StratusLab aims to facilitate and support the provisioning of grid resource over cloud infrastructures. This is expected to impact the type and amount of accounting information being kept. The cloud and the grid are residing in two distinct layers of the StratusLab architecture. In principle the hosting of grid services is independent of the underlying infrastructure being used whether this is a set of physical machines or virtualized resources within a public or private cloud environment. Grid sites already implement their own services for keeping various accounting information. Currently APEL [2] and DGAS [7] are the two most popular accounting frameworks integrated in gLite and are used in the context of EGI. These accounting tools collect information related to resource consumption from grid users. The primary focus is placed on information regarding grid jobs. The job is the primary abstraction for resource consumption that encapsulates an application, information about data produced and consumed, and the resource requirements for its proper execution. For grid sites it is interesting to know the number of jobs submitted, completed or failed. Information can be extracted about the users submitting jobs. The users can be grouped into VOs. For this reason aggregated information about VO usage can be derived. From that additional information can be extracted e.g. statistics per application domain, per NGI, per project etc.

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 [13] are working on the systematic collection of grid usage data with the purpose of building ontology for grid domain knowledge.

In the context of StratusLab we are interested in collecting information about resource utilization from grid resource providers that exploit a cloud infrastructure for deploying virtualized grid sites. We identify four different actors:

The grid user Is 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.

The VO manager Is 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.

The 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.

The cloud provider Is managing 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).

In the context of StratusLab a number of partners will be acting as cloud providers offering cloud capabilities based on the physical infrastructure committed for the project using the cloud distribution that the project integrates. Although the exact provision policies are expected to be defined in the coming months, and in particular in D5.2, the grid site manager is expected to be the main contact point for allocating resources for grid sites. Site managers may wish to deploy a complete set of grid services on top of virtualized resources. Alternatively they may wish to deploy only a subset of the resources that they control as a part of a larger grid installation. For example a physical grid site may wish to expand the range of WNs offered by utilizing a number of WNs from the cloud.

From the point of view of the cloud, the grid site manager will be the cloud user and the entity for which accounting information will be kept (number of VMs, number of CPUs) etc. Usage quota and resource usage restrictions will be enforced on the grid site level. These restrictions will be 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.

The grid site itself will probably need to keep information of cloud resource usage per grid user. This will be especially important in the case the grid site exhibits some kind of elasticity behavior. For example the instantiation of a new WN VM may be the result of a users requesting additional CPU cores to run their applications. A grid site will probably need to keep track of this usage thus the grid middleware should inform the cloud middleware about the grid user using a specific resource. Since grid users are identified by a digital certificate it will be enough to associate the usage of a specific could resource with a certificate DN.

Inversely, the grid accounting system will need to expand in order to include information about consumption of virtualized resources. Thus alongside the grid job information regarding memory and CPU consumption grid, accounting will keep information about the number and type of VMs instantiated, the amount of time used, the total network traffic among virtualized resources, the amount of virtualized storage used etc. The information will be kept per site and will help derive statistics for virtualized grid resources utilization per user and per VO.

Apparently, the above requirements need to be further developed by discussions with two more projects that are directly involved with the provision of grid services in Europe, namely EGI-InSPIRE and EMI. In particular, with EGI-InSPIRE we will discuss the accounting procedures and policies for grid sites. Finally, with EMI we will need to look into the technical requirements from the point of view of the middleware be it the cloud middleware integrated in StratusLab or the grid middleware developed by EMI.

8 Updated StratusLab Architectural Design

Figure 8.1 presents a high level design of the StratusLab architecture with all the components required to build an IaaS cloud with advanced functionalities provided in the JRA package. It is possible to see that the picture is quite similar to the one presented in D4.1 [23], plus the inclusion of the service manager component and the usage of the web APIs to access to both the service manager and OpenNebula.

OpenNebula is the virtual machine manager and it is charge of orchestrating requests and managing the allocation of resources, as well as the lifecycle of running virtual images. It will be deployed in a site and will manage the required resources (cluster, machines, storage, nodes, etc.).

The service manager works on top of OpenNebula. The SM is responsible for managing the overall service, not only the virtual machines of which it is composed. The input of the service manager is the service manifest formalized in the OVF language. It contains information about the virtual appliance to be deployed, that is to say, service and virtual machines information, as well as network and storage features. Each one of the virtual machines and networks is deployed in the physical infrastructure by OpenNebula, which will be in charge of managing them and sending the information to the SM which continues with the service management.

In order to obtain service scalability according to service providers' requirements, the service manager is able to manage the service scale up or down process following elasticity rules defined by the service providers. These rules are formalized again in the OVF using its extension capabilities.

The access to both OpenNebula and service manager is through the TCloud API (explained in Section 3.1.2). This means that both components are exposed as services, so that, they can be accessed via the web.

Finally, accounting and monitoring functionalities are provided unified at the physical and virtualization layer by a single component (see Section 7). This component also interacts with the accounting and monitoring facilities of the grid services running inside the virtualized environment provided by OpenNebula. The accounting and monitoring component also interacts with the service manager in order to feed the latter with the relevant information needed for enabling the fair share of virtualized resources. This picture is still rather abstract since decisions on accounting and monitoring tools remain to be made, and will be reflected in the next version of this document.

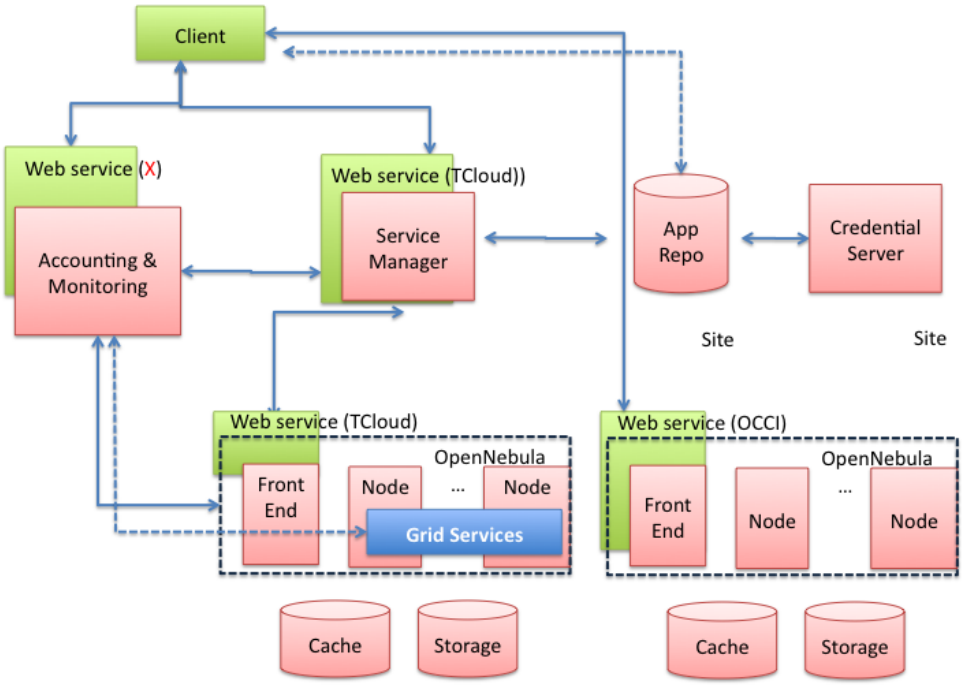

Figure 8.1: *Updated StratusLab Architecture*

9 Conclusions and Future Work

This document has been a starting point of the activities that are going to be carried out in the research workpackage. In a way, it is an input for grid service providers in StratusLab to know what the research activities are able to do, so that, they can provide requirements and feedback to guide the WP6 work in next months. On the other hand, it has provided a updated version of the StratusLab architecture including WP6 activities, and it is a starting point for technical discussion (e.g. monitoring tools in StratusLab).

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. For the definition of this service behavior as well as the service, virtual machines and networks features, the document has analyzed the OVF standard. Contextualization information for VM has been defined and used in OpenNebula. 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 alternatives to be the APIs to access to the Service Manager and Virtual Machine Manager. Regarding monitoring and accounting, some alternatives have been identifies as a starting point for discussion next months.

Next steps in this WP will be the deployment of the Service Manager and the APIs server in the StratusLab testbed, to test its functionality and to check its advantages. Some user stories will be created following the Scrum methodology which StratusLab is following. Thus, the user stories will be described using a narrative style, for instance *as a grid service provider I want to deploy my grid service just specifying all my requirements in a xml file* or *as a grid service provider I want to scale up my service when...*, or *as a grid service provider I want to the balanced virtual machines in my services are scaled up when by balancer needs it that is when ....*

To start with testing, we will use some grid existing service application, for instance the Grid Sun Engine (since this application has already been tested with the service manager and all the required artifact has been created). Other applications more suitable for StratusLab will be also considered like Torque. In addition, we will analyze how to integrate gLite services and the Quattor application with

the service manager, that is, to see how these services can be deployed by using Claudia.

Finally, this document will be updated in the D6.4 Cloud-like Management of Grid Sites 2.0 Design Report at PM15.

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	Virtual 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] P. Anderson and E. Smith. System administration and cddlm. In *Proceedings of the GGF12 CDDLM Workshop*. Global Grid Forum, 2004.

[2] APEL. Accounting Processor for Event Logs. http://goc.grid.sinica.edu.tw/gocwiki/ApelHome.

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

[4] 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.

[5] 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.

[6] Collective. Argus authorization service. Online resource, 2010. https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework.

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

[8] DMTF. CIM Infrastructure Specification. Specification DSP0004 v2.6.0. Technical report, Distributed Management Task Force, Mar 2010. http://dmtf.org/sites/default/files/standards/documents/DSP0004_2.6.0_0.pdf.

[9] 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.

[10] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, November 2008.

[11] T. Freeman and K. Keahey. Contextualization: Providing One-Click Virtual Clusters. In *Proceedings of the eScience08 Conference*, December 2008.

[12] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero. Service specification in cloud environments based on extensions to open standards. In *COMSWARE '09: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, pages 1–12, New York, NY, USA, 2009. ACM.

[13] Grid Observatory. http://www.grid-observatory.org.

[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] Munin. Munin (network monitoring application). Online resource. http://munin-monitoring.org/.

[16] OASIS. Oasis sdd starter profile. In *Specification, Organization for the Advancement of Structured Information Standards*. Organization for the Advancement of Structured Information Standards, 2008.

[17] OASIS. Oasis solution deployment descriptor (sdd). In *Specification, Organization for the Advancement of Structured Information Standards*. Organization for the Advancement of Structured Information Standards, 2008.

[18] OCCI-WG. Open Cloud Computing Interface Specification. Technical report, Open Grid Forum, 2009. http://forge.ogf.org/sf/go/doc15731.

[19] OGF. Open Grid Forum (OGF). Online resource. http://www.ogf.org/.

[20] OpenNebula. OpenNebula Accounting Toolset. http://www.opennebula.org/documentation:rel2.0:accounting.

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

[22] Stratuslab Consortium. Deliverable 2.1 Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures. Online resource., 2010. http://stratuslab.eu/lib/exe/fetch.php?media=documents:stratuslab-d2.1-v1.2.pdf.

[23] Stratuslab Consortium. Deliverable 4.1 Reference Architecture for StratusLab Toolkit 1.0. Online resource., 2010. http://stratuslab.eu/lib/exe/fetch.php?media=documents:stratuslab-d4.1-v1.0.pdf.

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

[25] TID. The claudia project. Online resource., 2010. http://claudia.morfeo-project.org/wiki/index.php/Main_Page.

[26] DMTF. The distributed management task force webpage. Online resource., 2010. http://www.dmtf.org.

[27] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.

[28] 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.