Enhancing Grid Infrastructures with
Virtualization and Cloud Technologies

Reference Architecture for StratusLab Toolkit 1.0

Deliverable D4.1 (V1.0)
16 September 2010

Abstract

This document presents the architecture for StratusLab v1.0. The architecture consists of a set of services, components and tools, integrated into a coherent whole and a single distribution, providing a turnkey solution for creating a private cloud, on which a grid site can be deployed. The constituents of the distribution are integrated and packaged such that they can easily be installed and configured, using both manual and automated methods. A reference architecture is presented, fulfilling the requirements and use cases identified by D2.1 and further analysed in this document. The selection of the elements composing the distribution was made following an updated analysis of the state-of-the-art in cloud and a gap analysis of grid and cloud related technologies, services, libraries and tools. In order to enable cloud interoperability, several interfaces have also been identified, such as contextualisation, remote access to computing and networking, as well as image formats. The result of this work will be StratusLab v1.0, a complete and robust solution enabling the deployment of grid services on a solid cloud layer. The architecture will be updated at project month 15 in Deliverable D4.4.

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
Marc-Elian Bégin	SixSq	All
Eduardo Huedo	UCM	6.2.1, 6.2.2, 6.3, 6.5 and 3.3.5, 7, 8
Rubén S. Montero	UCM	6.2.1, 6.2.2, 6.3, 6.5 and 3.3.5
Vangelis Floros	GRNET	3.1

Document History

Version	Date	Comment
0.1	16 July 2010	Initial version for comment.
0.2	10 August 2010	Chapters 3 and 6.
0.3	30 August 2010	Added results from early sprints.
0.4	4 Sept. 2010	Added section on grid services.
0.5	5 Sept. 2010	Added section on grid services.
0.6	8 Sept. 2010	Updated State-of-theArt chapter and added Gap Analysis Chapter.
0.7	10 Sept. 2010	Content review by Charles Loomis.
0.8	13 Sept. 2010	Editing review by Louise Merifield.
1.0	14 Sept. 2010	Final version.

Contents

List of Figures

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 virtualisation 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 utilisation of the resource provider's physical infrastructure and facilitate the day-to-day tasks of the grid site administrator.

This document presents the architecture of the StratusLab v1.0 distribution. While this architecture is expected to evolve over the course of the project, it is important to define and agree on the high-level principals and the foundations on which StratusLab will be based. This document is therefore a starting point and the architecture it describes will be followed and updated as the project progresses. The document will be updated at PM15 with D4.4, which will add elements required for StratusLab v2.0.

The document first summarises requirements and use-cases gathered by WP2 with surveys performed during the summer 2010. These surveys were designed and conducted by WP2 and comprehensively analysed in deliverable *D2.1 Review of the Use of Cloud and Virtualization Technologies in Grid Infrastructures*. The set of requirements and use cases identified in D2.1 is then completed and consolidated in this document with previous work leading to the submission of the StratusLab proposal, and updated since, including quality, stability, constraint, performance, security and standardisation requirements. We then briefly analyse these in the context of StratusLab v1.0, to ensure that v1.0 meets the most urgent needs from our user communities.

An important goal of StratusLab is to produce a high-quality software solution, able to fulfil the requirements of our target user communities. Further, the project is relatively short (2 years) and composed of distributed and highly skilled teams, needing close and rapid co-operation. To better face this challenge, we have decided to adopt an agile development methodology. While this is not usual for FP7 projects, the StratusLab team has worked closely together for a number of years, leading to the submission of the project, with a resulting high level of trust among the team. We also realise that in order to produce high-quality software, strong engineering practices must be put in place, with a focus on process automation and testing. This methodology is presented in Chapter 4, including our engineering

process in terms of build, packaging, installation, configuration and test.

In Chapter 5, we then move on to present the reference component architecture capable of implementing the requirements and use cases identified in 3. We list the different elements needed to fulfil the requirements already identified, while remaining neutral in terms of concrete implementation. To better manage complexity in the installation and deployment of the StratusLab, but also to guide our testing efforts, we also present the reference deployment model assumed by default by our installation tools.

The actual realisation or implementation of the StratusLab architecture is presented in Chapter 6 where components are reviewed and selected to fulfil the identified functions. Where no existing solution is identified, fulfilling the StratusLab requirements, custom development is then identified for the project to realise. The core virtual machine management functionality is covered by OpenNebula, which also provides an abstraction to the virtualisation layer required to handle virtual machines. Other functions such as the Appliance Repository is implemented using the Apache HTTP Server, connected to a credentials server: LDAP. Other services and components are also selected. The glue required to bring these different entities into a coherent and single distribution will be developed by StratusLab.

Significant background analysis effort went into the selection of several of the components identified for implementing StratusLab, including strategies, technologies and standards. This background work is based on a thorough review of the state-of-the-art in the cloud computing eco-system. While innovative and ground breaking development has come from research projects, which are summarised in Chapter 7, we also realise that cloud computing, contrary to grid computing, is still dominated by the private sector. We therefore complete our updated state-of-the-art review with a summary of a survey conducted on cloud trends from the commercial world.

Before the summary, Chapter 8 brings together several gaps, but also opportunities for synergy, between grid and cloud, which will trigger further work which we hope will form part of the features for StratusLab v2.0. For example, monitoring and accounting must be integrated between the cloud and grid layers to ensure that the right level of reliable information is provided to system administrators and end-users. Further, these services form the foundation for more advanced features, such as auto-scaling, leveraging the power of cloud to provide a new scaling model for grid and virtual organisation services.

2 Introduction

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 virtualisation 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 utilisation of the resource provider's physical infrastructure and facilitate the day-to-day tasks of the grid site administrator.

This document presents the architecture of the StratusLab v1.0 distribution. While this architecture is expected to evolve over the course of the project, it is important to define and agree on the high-level principals and the foundations on which StratusLab will be based. This document is therefore a starting point and the architecture it describes will be followed and updated as the project progresses. The document will be updated at PM15 with D4.4, which will add elements required for StratusLab v2.0.

Chapter 3 first summarises requirements and use-cases gathered by WP2 with surveys performed during the summer 2010. This set of requirements is completed with previous work leading to the submission of the StratusLab proposal, and updated since, including quality, stability, constraint, performance, security and standardisation requirements. We then briefly analyse these in the context of StratusLab v1.0.

An important goal of StratusLab is to produce a high-quality software solution, able to fulfil the requirements of our target user communities. Further, the project is relatively short (2 years) and composed of distributed and highly skilled teams, needing close and rapid co-operation. To better face this challenge, we have decided to adopt an agile development methodology. This methodology is presented in Chapter 4, including our engineering process in terms of build, packaging, installation, configuration and test.

In Chapter 5, we then move on to present the reference component architecture capable of implementing the requirements and use cases identified in 3. We also present the reference deployment model assumed by default by our installation tools.

The actual realisation or implementation of the StratusLab architecture is presented in Chapter 6 where components are reviewed and selected to fulfil the identified functions. Where no existing solution is identified, fulfilling the StratusLab

requirements, custom development is then identified for the project to realise.

Significant background analysis effort went into the selection of several of the components identified for implementing StratusLab, including strategies, technologies and standards. This background work is based on a thorough review of the state-of-the-art in the cloud computing eco-system. Chapter 7 summarises our findings, and also includes trends from the commercial world.

Chapter 8 identifies several gaps, but also opportunities for synergy, between grid and cloud, which will trigger further work that will form part of the features for StratusLab v2.0.

3 Requirements

The StratusLab project and the respective cloud distribution 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 virtualisation in order to offer grid administrators the ability to set up and manage flexible, scalable and fault tolerant grid sites. This way we will enable the optimal utilisation of the resource provider's physical infrastructure and facilitate the day-to-day tasks of the grid site administrator.

The development of the StratusLab distribution is driven by user requirements. These requirements are of different natures - e.g. functional, constraints, performance, security. The following sections describe our methodology for defining and handling these requirements.

We start with grid site requirements gathered during the project preparation, followed by an analysis of the requirements and use cases identified in D2.1. We end this chapter with a summary of requirements coming from the need for standardisation.

3.1 Grid Site Requirements

Typically a computing grid provides four set of services:

- Workload Management
- Storage Management
- Information Management
- Security

These services are offered through a set of software components generally denoted as grid middleware. From the point of view of grid services support, the requirements imposed on cloud layer are more functional than technical. StratusLab should offer a stable cloud middleware for hosting all types of grid computing services as virtual machines. According to this requirement there should be no performance penalties or functionality compromises because of the cloud. Nevertheless, for the optimal interaction between the two worlds, the grid middleware should probably also adopt to the special characteristics of the cloud.

StratusLab clouds should be able to sustain high workload demands imposed by the provision of grid workload management services. The underlying cloud layer should allow virtual machine instances to utilise most of the underlying physical system processing power by minimising the interference between the two actors (virtual machine and physical machine) enabling optimal execution of application workloads and as a result fast execution turnaround. In order to achieve this, StratusLab should be compatible with hypervisor solutions that offer the best approach for accessing the raw computing power of a system.

One of the benefits offered by virtualisation is the ability to migrate a running instance from one host to another thus increasing the availability of a system. For example, in a case that system maintenance is required for a specific node which typically means bringing the node off-line, the provision of services from this node need not stop, but hosted VMs can be migrated elsewhere before proceeding with the node shutdown. StratusLab should support such on-demand migration of virtual machines.

Grid sites should exhibit a dynamic elasticity behaviour based on workload. This requires interoperation with the underlying grid middleware. For example, the addition or removal of a worker node from a grid site requires the update of the node's information in the cluster head node running the Local Resource Management System (LRMS). In the case of gLite this node is referred as a Computing Element (CE).

StatusLab should be able to satisfy the requirements for flexible management of mass storage. In particular a system administrator should be able to manage the virtual storage assigned for Storage Elements. In a typical use case a research group shares the results of an experiment by offering data volumes (e.g. in the form of Elastic Block Device) that other groups can replicate and attach to their VMs. This capability should be interoperable with the way popular grid middleware manages storage devices. For example, in gLite, DPM is used to access NFS, LVM and GPFS volumes. StratusLab should be versatile enough to support the creation of virtual data volumes that can be attached to VMs offering storage services (Storage Elements in gLite).

Finally StatusLab should be transparent to the end user and the security procedures engaged during the interaction of a user's job with the grid site.

For the information system the node should appear as a regular node reporting back all information about system load, memory usage, network bandwidth usage etc. The changes taking place in the cloud (node migration, dynamic expansion or reduction of available storage) should be reflected in the monitoring of the node in reasonable time in order to allow the components depending on the information system to have the most up-to-date information of the grid site status.

3.2 End-User and System Administrator Requirements

3.2.1 High-Level Requirements

Below are the requirements identified in the Deliverable D2.1. For each requirement a comment is provided explaining the level of compliance expected from StratusLab v1.0.

1. *"The project should support installation of the cloud distribution on RedHat and Debian systems, with RedHat systems having a much higher priority."* Our development strategy already includes the support of these two operating system families.

2. *"Integration of the cloud distribution with automated site configuration and management tools should be demonstrated with Quattor and/or Puppet, with Quattor being the more popular."* Quattor is already the default tool supported by StratusLab.

3. *"Demonstrations of grid services over the cloud should initially target core services of the gLite middleware."* The tests performed by WP5 of the StratusLab incremental deliveries focuses on using core gLite services.

4. *"The cloud distribution must supply stock images for popular Red-Hat and Debian-based systems."* Base/stock images form an important part of the standardisation effort StratusLab is undertaking, especially with respect to contextualisation.

5. *"The cloud infrastructure must be as operating system neutral (with respect to running virtual machines) as possible to maximise its utility."* This feature is provided by standard virtualisation technologies, such as KVM, Xen or VMware, which integrates with StratusLab (KVM and Xen for v1.0).

6. *"The application benchmarks must cover all of these types of applications: sequential, multi-threaded, shared memory, and parallel."* Benchmarks will be integrated with the StratusLab distribution, such that they can provide metrics covering these types of applications.

7. *"The application benchmarks should include workflow and master/worker applications."* Same as previous.

8. *"The application benchmarks must be parameterised to allow a wide range of input sizes, output sizes, and running times to be evaluated."* Same as previous.

9. *"The project must create application benchmarks (CPU-Intensive, Simulation, Analysis, Filtering, Shared Memory, Parallel, and Workflow) to measure quantitatively the performance of the cloud implementation for realistic applications."* Same as previous.

10. *"Performance benchmarks should also be created using packages like HEP-SPEC, Iozone, and iperf for CPU, disk IO, and network performance, respectively."* Same as previous.

11. *"The StratusLab cloud implementation must include access control mechanisms for stored data and must permit the use of encrypted data."* Data storage, as well as persistence is required as a basic feature. Rich access control will also be investigated.

12. *"The cloud must allow both file and block access to data, although file access is by far more important."* It is unclear where the boundary for application data access lies between grid and cloud services. This requires further analysis with users.

13. *"The cloud must allow access to data stored in object/relational databases."* Same as previous.

14. *"Short-term work (<12 months) should concentrate on developments for deploying cloud infrastructures and longer-term work should concentrate on their use."* This recommendation will drive the feature selection for StratusLab v1.0.

15. *"The StratusLab distribution must be simple enough for users themselves to configure their own resources as a cloud."* This is a strong motivation for providing simple, script-based installation and configuration methods for the StratusLab distribution.

16. *"The StratusLab distribution must allow both full-virtualisation and para-virtualisation to be used."* This is also a hypervisor issue, but the configuration of these technologies must be such that it enables these choices.

17. *"The cloud service must have a command line interface and a programmable API."* Both access patterns will be provided, with when appropriate the same implementation to reduce duplication and increase quality.

18. *"The cloud distribution must allow a broad range of grid and standard services to be run."* StratusLab v1.0 will focus on the core grid services, while more dynamic services (e.g. requiring auto-scaling) will be investigated for v2.0.

19. *"Quantitative performance evaluations must be done to understand the penalties in using virtualisation."* Benchmarks will be shipped with the StratusLab distribution, such that system administrators can execute these on their own resources. Further, StratusLab will conduct its own analysis.

20. *"The project must determine the criteria by which administrators and users can trust machine images."* This trust issue is paramount to the success of the

project. Different communities involved are being engaged with to ensure a solution is found which is acceptable to all. The architecture will support basic signing of machine images.

21. *"The project should consider all features listed in the surveys as valid requirements."* All listed are certainly considered valid.

22. *"Integration with site management tools is a critical short-term requirement."* The integration with Quattor and the ability to install and configure StratusLab using manual tools provides the minimum implementation of this requirement. Further integration in terms of monitoring and accounting will follow.

23. *"The cloud implementation must scale to O(10000) virtual machines."* This requirement will be taken into account as part of the performance requirements.

24. *"The implementation must sufficiently sandbox running machine images to prevent unintended or malicious behavior from affecting other machines/-tasks."* This security requirement is analysed in Section 3.3.4.

Further to these requirements gathered in the survey we can add:

25. *StratusLab shall be built out of open source components.* We have pledged, and our users require, that StratusLab shall be an open source distribution. While it should be possible to substitute open source components with commercial ones, it is critical that the entire feature set provided by StratusLab be available under an open source license. Whenever possible, every effort should be taken to select harmonious licenses. If components only available under commercial license are required, then all project members should be given equal access to the license for the duration of the project.

26. *The cloud layer exposed via the cloud API shall provide a fair share mechanism.* An important issue left out here is how to throttle the requests, such that each user gets a fair share of resources, and that the site remains elastic as much as possible. This aspect of the cloud API will be further explored later this year and reported in the next version of the architecture document.

27. *Cloud physical and virtual resources shall integrate with existing grid monitoring and accounting solutions.*

Further analysis of these requirements is provided in the remainder of the document.

3.2.2 Use Cases

The following extract from D2.1 accurately set the scope for the StratusLab distribution:

> The primary vision of the StratusLab project is to create a private cloud distribution that permits resource centre administrators to deploy grid (and standard) services over the cloud's virtualised resources and that allows scientists to exploit e-infrastructures through a cloud-like interface. This vision emerged from a set of use cases and scenarios developed by the project's partners and based on their experience with existing e-infrastructures.

The primary use cases are described below.

Adding to the requirements already identified and listed in Section 3.2.1, D2.1 also expanded this vision statement in the form of use cases. They are reported here, albeit slightly reformatted for ease of reading, with comments on the applicability and feasibility for v1.0, which is what concerns the current architecture. We also provide elements of solutions when possible, solutions that will be expanded in the rest of the document.

Grid Services on the Cloud *Grid services are numerous, complex, and often fragile. Deploying these over virtualised resources would allow easy (re)deployment or use of hot spares to minimise downtime. Efficiency of these services is important, so having benchmarks to measure the "virtualisation penalty" for real services is equally important."* Live migration of virtual machine will address the first part of this use case, while the benchmarks component of StratusLab provide the measure for the assessment of "virtualisation penalty".

Customised Environments for Virtual Organisations *The computing environment offered by the European Grid Infrastructure is homogeneous regarding the operating system (Scientific Linux [18] is nearly universally used) but inhomogeneous in terms of which software packages are available on a particular site. The first limits the appeal of the infrastructure to those people already using Scientific Linux (or a close relative) and the second increases application failures from missing dependencies. Allowing grid Virtual Organisations (VOs) to develop their own computing environments as Worker Node images would solve both of these issues.* Providing an API to grid users such that they can choose their runtime environment in terms of custom virtual machine will partly address this requirement. A complete solution will require integration of this API with grid services, such that both layers can support this use case. As a first step though, VO could specify the virtual machine containing the Worker Node in which jobs from that VO will be deployed.

Customised Environments for Users *Although allowing VOs to provide virtual machine images increases the utility and reliability of the grid infrastructure, individual users are also likely to require customised environments containing,*

for example, their own proprietary software and/or data. Extending the ability to create virtual machines to individual users further enhances European e-infrastructures. Already mentioned in the previous point, granting grid users with a new cloud API for specifying is required. Further, support for machine image creation is required. Also required is a standard solution integrating the cloud API with the grid API. While this feature was originally planned for StratusLab v2.0, the surveys have clearly shown that this matter is critical. For this reason, we are considering adapting our work programme to support, at least in part, this use case in v1.0.

Sharing of Dataset and Machine Images *Preparing a virtual machine or a dataset image requires significant effort, both in terms of creating it and validating it. Providing a mechanism by which users can share these images avoids duplicated effort and promotes the sharing of knowledge. Solid policy recommendations allowing people to trust those images/dataset must come hand-in-hand with technical solutions to enable the sharing (repositories, access control, security, etc.).* This use case bundles several requirements together: 1- the ability to create dataset and machine images, 2- validation/verification of these, 3- sharing (e.g. via an appliance repository). This use case reinforces the need for users to be able to conveniently and safely specify custom images.

Provision of Common Appliances *New virtual machines images are often built from existing ones. Providing simple, stock images of common operating systems lowers the barrier to creating customised images as well as improves the utility of the cloud infrastructure. In addition, appliances for grid services would facilitate their deployment at smaller sites or sites with inexperienced system administrators.* This again reinforces the need to simplify and lower the barrier for end-users to be able to create a custom runtime environment, in the form of custom machine images. This use case applies this generic feature to a new deployment model for grid services, which again should improve the ease of management of grid sites using StratusLab, or any compatible cloud solution.

VO/User Services on the Cloud *Because of security concerns and lack of tools for controlling network access, VOs and users cannot currently deploy services on the European Grid Infrastructure. VOs often have significant software infrastructures built on top of the grid middleware to provide specialised services to their communities. Users often run calculations that involve workflows or task management. Frameworks to execute those types of calculations usually require a central, network-accessible controller (service) to manage the deployment of tasks and collection of results. Deploying these types of services is possible with a cloud.* This use case can be implemented using the same feature proposed for the previous use case, which provides a single set of resources, alleviating the need for VOs to maintain their own infrastructure (or having it specifically maintained for them), giving them the opportunity to focus on their scientific work.

Deployment of a Group of Machines *Large calculations often involve the deployment of a group of machines. Examples include the deployment of a batch system, Hadoop [3], workflow engines with workers, and BOINC [8]. The cloud should facilitate the deployment of groups of machines to make deployment and configuration of these high-level systems as efficient as possible.* In v1.0 it will be possible to deploy several machines together, connected together via virtual networks. A high-level set of services, such as auto-scaling and load balancing, will be analysed and implemented in v2.0.

Hybrid Infrastructures *All resource centres have a finite amount of computing resources available, although the cloud model aims for the appearance of infinite resources ("elasticity"). To handle peaks and maintain the elasticity of a resource centre, the system administrator may want to offer resources from other clouds (e.g. a public cloud like Amazon Web Services [2]) via their cloud. This public/private "hybrid" cloud allows a site to maintain a high-level of service to its users.* StratusLab v1.0 will put in place the foundations for sites to run as private clouds. Meanwhile, the project will work on addressing the significant challenges in providing a workable, simple, controlled, affordable and secure solution for being able to offload peak demands to other clouds, public and or private.

3.3 Requirements Analysis

From the requirements and use cases presented thus far, this section analyses these, providing insights and trade-offs, when appropriate, from a technical perspective. We also identify recommendations for a technical solutions, when possible.

3.3.1 Quality and Stability Requirements

It is important that StratusLab is a high quality distribution, since it will form the foundation layer for grid services to run on. Building credibility is an important driver for our dissemination activities. It is also important to realise that StratusLab will largely be composed of existing tools, libraries and implementations, which have to be carefully selected. It is therefore important that each constituent of the distribution be well supported and mature. Ideally members of StratusLab will be part of or in contact with the developer communities behind these components.

3.3.2 Constraint Requirements

The initial system administrator community will come from the grid world. This community largely uses Scientific Linux [18] as the base operating system. While this is likely to remain the case initially, it is important that the StratusLab distribution be a multi-operating system. Therefore, right from the start, StratusLab shall support more than one operating system. The logical candidates are: CentOS [10] (close cousin of Scientific Linux [18] and also Red-Hat [38] based) and Ubuntu [9] (a Debian [43] based operating system). Supporting these two different camps in the Linux world of distribution should enable us to extend our support to other operating systems, for example SUSE, without too much effort.

Ease of installation and configuration is an important tenet of StratusLab. This probably means that StratusLab will have to be opinionated, which means that while customisation and extension shall be possible, assumptions will be made to facilitate the installation and configuration of the system. An important balance is required between flexibility and usability, which our user communities will be instrumental in reaching. It shall therefore be possible to install and configure StratusLab using manual tools and automated fabric management tools (e.g. Quattor [37]).

3.3.3 Performance Requirements

The survey conducted earlier and reported in D2.1 illustrates clearly that system administrators are expecting to be able to handle on the order of tens of thousands of virtual machines over thousands of physical hosts. These numbers are therefore defining clearly what is the expectation in terms of performance of the system.

Another important performance parameter is the need for the system on which StratusLab is deployed to remain elastic, from a user perspective. While there are physical limitations that limit the elasticity of any system, it is important that the system behaves elastically, which means a reasonable latency for all requests, without exposing a queue semantic to the user.

Cloud computing is based on virtualisation and virtualisation means virtual images, which can be large files. We are therefore faced with manipulating large files, over potentially wide area networks. Since these images are largely 'write once and read many', a caching strategy is appropriate to avoid having to transfer the same files several times over the network.

3.3.4 Security Requirements

While the grid does not have as stringent security requirements as some commercial and military applications, it is important that the system provide a reasonable level of security.

It is important that the system administrators trust that running virtual machines on their infrastructure is not a threat to that infrastructure. Similarly, end-users deploying virtual machines on a remote infrastructure via the StratusLab cloud API must also feel certain that the execution of their machine will not be tampered with and that data used in that context can also remain secure.

Virtualisation technologies by their very nature already provide a powerful level of separation between the physical hosts, under the control of the system administrators, and the running of virtual machine instances, managed by the end-users. However, virtualisation technologies are complex and it is therefore important that their installation and configuration is performed such that it does not leave back doors open to malicious exploitation.

An important aspect of security is establishing a trust relationship between the system administrators and the end-users, such that a foreign virtual machine can be deployed on a remote infrastructure. For the cloud model to work in supporting the grid, it is fundamental that this trust relationship be established and negotiated

without requiring constant intervention of humans. It is therefore required that virtual machines be signed in some way such that the pedigree of the machine, its provenance and composition be tracked back to a trusted user.

Another important aspect to security that requires our attention is regarding credential management. All the services integrated in StratusLab shall integrate with a centralised and shared secure credential server, for example LDAP and/or OAuth. This means that the same credentials can be used to authenticate and authorise access to the different StratusLab integrated services. Furthermore, the StratusLab security model must integrate and/or be compatible with the grid security model, such that traversing from the grid to cloud layers and back does not open a security hole.

3.3.5 Standardisation Requirements

StratusLab sites will expose elastic behaviour exploiting the underlying capabilities of the StratusLab cloud in order to adapt to peak loads observed from various grid applications. The infrastructure will also be able to utilise external resources residing outside the borders of the participating institutes. Such resources will be for instance large public cloud providers, like Amazon's EC2 service, therefore demonstrating that the EGI infrastructure can operate in a hybrid public–private cloud platform, able to take advantage of external cloud resources when peak demand requires.

A hybrid cloud is an extension of a private cloud to combine local resources with resources from remote cloud providers. The remote provider could be a commercial cloud service, such as Amazon EC2 or ElasticHosts, or a partner infrastructure running the StratusLab cloud distribution. Such support for *cloudbursting* enables highly scalable hosting environments.

The site infrastructure running StratusLab will be fully transparent to grid and cloud users. Users continue using the same private and public cloud interfaces, so the federation is not performed at service or application level, but at infrastructure level. It is the infrastructure's system administrators who will take decisions regarding scale out of the infrastructure according to infrastructure or business policies.

However, the simultaneous use of different providers to deploy a virtualised service spanning different clouds involves several challenges, related to the lack of a cloud interface standard; the distribution and management of the service master images; the inter-connections between the service components, and finally contextualisation.

The following subsections describe several aspects related to interoperability which we plan to further explore in order to improve the level of interoperability of StratusLab with other cloud providers.

3.3.5.1 Cloud interfaces

Currently, there is no standard way to interface with a cloud, and each provider exposes its own APIs. Moreover the semantics of these interfaces are adapted

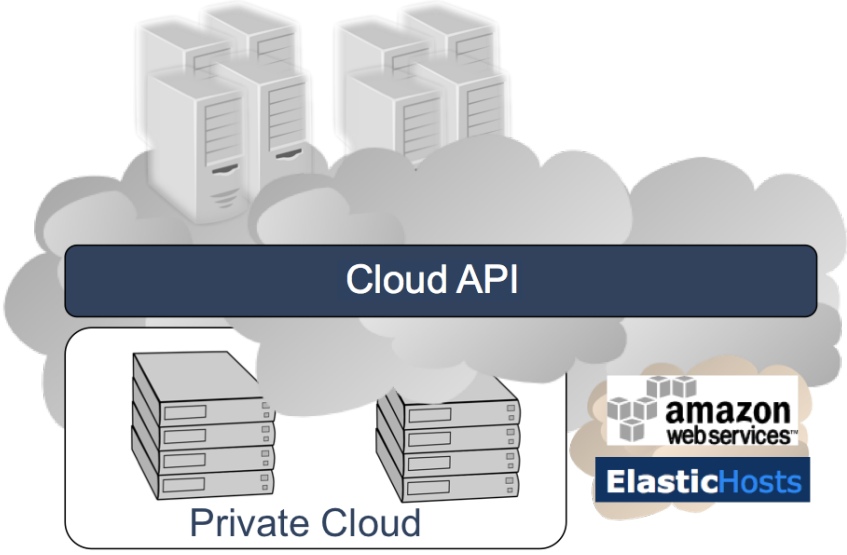

Figure 3.1: *Hybrid cloud.*

to the particular services that each provider offers to its costumer (e.g. firewall services, additional storage or specific binding to a custom IP).

A traditional technique to achieve interoperation in this scenario is the use of adapters [23]. StratusLab requires a pluggable and modular architecture, such that it can integrate with specialised adapters to interoperate with different hypervisor technologies (Xen, KVM, etc.), or different cloud providers, like Amazon EC2 and Elastic Hosts (EH) (see Figure 3.2).

Several standardisation bodies, such as the Open Grid Forum (OGF) and the Distributed Management Task Force (DMTF), have active working groups to produce standard cloud interfaces. For example, the Open Cloud Computing Interface (OCCI) specification is being developed by the OCCI Working Group inside the OGF community. The OCCI effort was initiated to provide a global, open, non-propietary standard to define the infrastructure management interfaces in the context of cloud computing. OpenNebula, one of our selected components, supports this standard.

The OCCI specification defines a simple (about 15 commands) and extensible RESTful API. It defines three types of resources: compute, storage and network. Each resource is identified by a URI, has a set of attributes and is linked with other resources. Resources can be represented in many formats such as OCCI descriptor format, Open Virtualisation Format (OVF) or Open Virtualisation Archive (OVA). Create, Read, Update and Delete (CRUD) methods are available for each resource, which are mapped to the typical REST methods over the HTTP protocol: POST, GET, PUT and DELETE. Other HTTP methods are also included by OCCI, like COPY, HEAD, MOVE and OPTIONS. Requests are used to trigger state changes

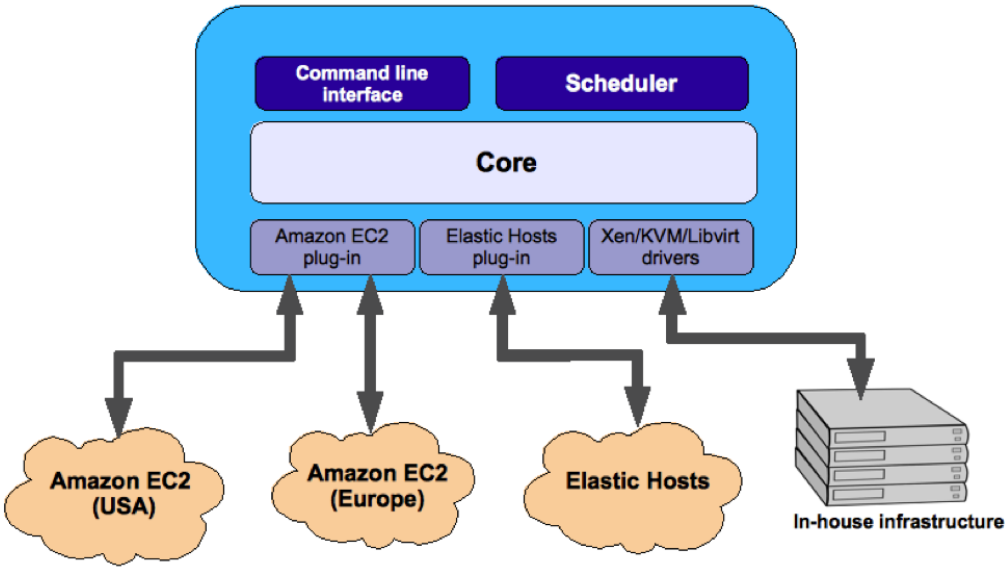

Figure 3.2: *Cloud brokering.*

and other operations (create backup, reconfigure, etc). A request is sent by a POST command to the resource URI, of which body contains the request and its parameters.

3.3.5.2 Image management

In general, a virtualised service consists of one or more components each one supported by one or more VMs. Instances of the same component are usually obtained by cloning a master image for that component, that contains a basic OS installation and the specific software required by the service.

Cloud providers use different formats and bundling methods to store and upload these images. We could assume that suitable service component images have been previously packed and registered in each cloud provider storage service. So when a VM is to be deployed in a given cloud the image adapters skip any image transfer operation. This approach minimises the service deployment time as no additional transfers are needed to instantiate a new service component. However, there are drawbacks with storing images in each cloud provider: higher service development cycles as images have to be prepared and debugged for each cloud; higher costs as clouds usually charge for the storage used; and higher maintenance costs as new images have to be distributed to each cloud.

Therefore, the project will investigate different options to resolve this important issue. Possible solutions could include the feasibility of offering a repository of reference images for cloud users, with demonstrated interoperability among the supported public cloud infrastructures, and following the existing standards in the areas of VM images and virtual appliances. Another idea would be to provide

an image creation engine, based on SlipStream for example, which can generate images from a higher-level recipe.

3.3.5.3 Network management

Resources running on different cloud providers are located in different networks, and may use different addressing schemes (public addresses, private addresses with NAT, etc.). However, some services require all their components to follow a uniform IP address scheme (for example, to be located on the same local network), so it could be necessary to build an overlay network on top of the physical network to allow the different service components to communicate. In this context, there are interesting research proposals (e.g. ViNe [48], CLON [31]) and some commercial tools (e.g. VPN-Cubed [12]), which provide different overlay network solutions for grid and cloud computing environments.

Virtual Private Network (VPN) technology could interconnect the different cloud resources with the in-house data centre infrastructure in a secure way. In particular, OpenVPN [36] software allows implementing Ethernet tunnels between each individual cloud resource and the data centre LAN, as shown in Figure 3.3. In this setup, which follows a client-server approach, the remote cloud resources, configured as VPN clients, establish an encrypted VPN tunnel with in-house VPN server, so that each client enables a new network interface which is directly connected to the data centre LAN. In this way, resources located on different clouds can communicate, as they are located in the same logical network, as well as accessing common LAN services (NFS, NIS, etc.) in a transparent way, as local resources do.

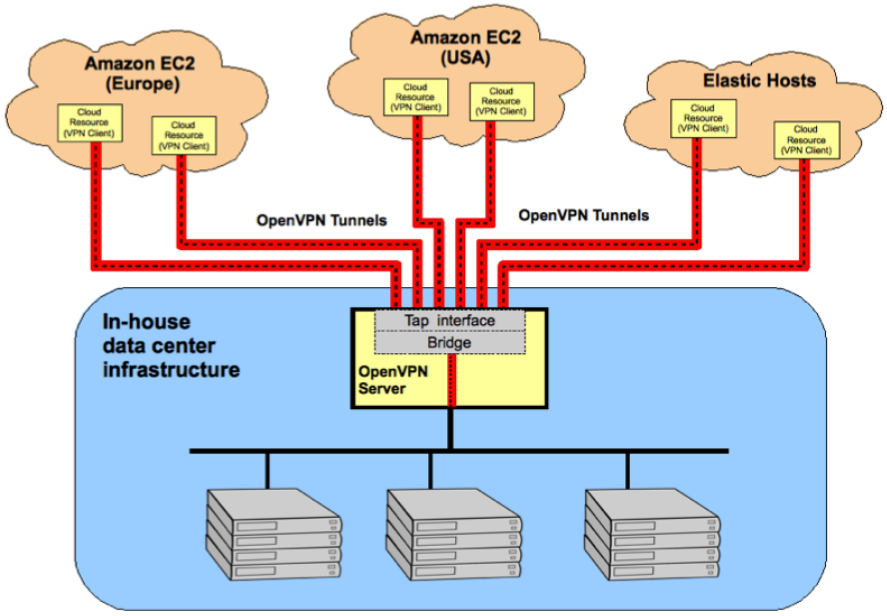

Figure 3.3: *VPN-based network configuration for a multi-cloud infrastructure*

4 Development Process and Strategy

This chapter describes our development process and strategy, including engineering practices and the mechanisms we have put in place to control and track the execution of our programme of work.

4.1 Agile Development Process

Faced with a challenging and fast moving eco-system around web and distributed technologies, we need to be able to quickly react to changes without losing focus nor impacting our performance with constant disturbances to our programme of work. We also need to remain closely focused on listening to our user communities, without losing focus on our obligation to deliver. We also need to avoid at all costs the tunnel vision side effect that over-engineering can cause when disconnected from reality.

Agile development processes are gaining in popularity in industry and government development. Most project members have also a long history of collaboration. We therefore have decided to apply an Agile methodology called Scrum to drive the development, integration and release efforts of the project. This allows us to define small units of work, able to add value to our users, on a quick turnaround time scale. Users can then provide valuable feedback which we can use to improve the project output, improve dissemination and ultimately the impact the project will have. This process includes the entire chain from collecting user requirements, architecture, design, implementation and testing, but also dissemination of our releases, and active pursuit of new user communities and management of feedback.

Figure 4.1 illustrates the Scrum process. For StratusLab, we have chosen to adopt a three week sprint duration. This means that every three weeks, the results of the sprint are demonstrated to all available project members (and any guests the project deems relevant). The two main objectives of the demonstration meeting are first to force the integration of the work performed in terms of releasable increment, and second to gather feedback from all stake-holders. Not required by Scrum but appropriate for a distributed project like StratusLab, and in accordance with the proposed management structure, the demo meeting is followed, normally the day after, by a session of the TCSG. This session is an opportunity to (re)assess the architecture of the system, review important feedback gathered during the demo and

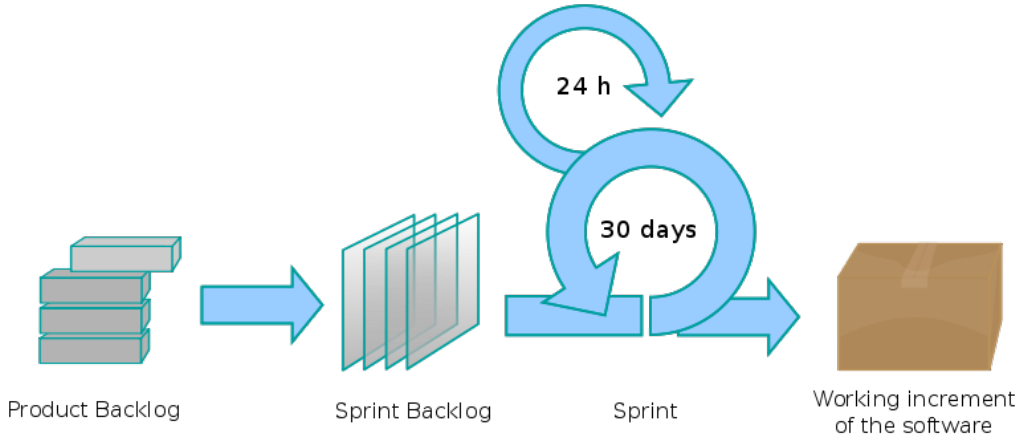

Figure 4.1: *Scrum process*

any other feedback received, and redefine the priorities of the project. Following on from this event, a *planning meeting* is held, reviewing the user stories proposed for the next sprint. During the planning meeting, the different technical teams estimate and commit to realising the user stories. This ends the Scrum cycle.

4.2 Users

We have identified several categories of users (or actors) that will interact with the system. While real users might fulfil several roles at once, it is important to separate and identify the concerns that these different types of users will have in the context of StratusLab. The user stories (see Section 4.3) refer explicitly to a user identified in this section.

- **Scientists**: End-users that take advantage of existing machine images to run their scientific analyses.
- **Software Scientists and Engineers**: Scientists and engineers that write and maintain core scientific community software and associated machine images.
- **Community Service Administrators**: Scientists and engineers that are responsible for running community-specific data management and analysis services.
- **System Administrators**: Engineers or technicians that are responsible for running grid and non-grid services in a particular resource centre.
- **Hardware Technicians**: Technicians that are responsible for maintaining the hardware and infrastructure at a resource centre.

Adding to this list of external users, we add 'StratusLab Team Members', since we are our first users. This allows us to target project member in user stories, which is useful when building our own support infrastructure and tools.

4.3 User Stories

Chapter 3 provided the requirements and use cases driving the technical work of the project. While these are critical in making sure that the project fulfils the need of our target communities, they are not in a format that can easily be translated into technical tasks that can be scheduled for sprints. For example they can be purposefully vague, leaving the technical decision to the project members responsible for their implementation, they can also be complex.

In order to decompose these into smaller elements, agile methodologies in general, and Scrum in particular, propose to create *user stories*. User stories are a high-level description of a need by a specific user or actor from the system, expressed in the form of an action and a resulting benefit. Most user stories are captured according to the following template: "As a *actor/user* I can *action* in order to *benefit*". The scope and complexity of each story is such that it can be implemented within a single development iteration (or sprint in the Scrum agile methodology we are using). This guarantees that entire stories can be implemented and delivered in a short time scale, such that it can be demonstrated and evaluated by project team members and/or target users. Here is an example taken from our JIRA tool:

> As a Sys Admin I can deploy a virtualised Worker Node (WN) that register to a Grid Computing Service

We do not list all user stories in this document since it would represent a significant effort and would cause duplication. StratusLab uses a collaboration tool called JIRA [6], augmented by the GreenHopper [5] plugin which provides JIRA with the added features for agile and Scrum support, including management of user stories. As already mentioned, user stories are assigned to sprints at the start of every sprint. At the end of every sprint, completed and accepted user stories are declared 'Done' and closed. Others are either rejected as not complete, and eventually rescheduled in a future sprint if still relevant .

While the detail of every user story is outside the scope of this document, it is important the the reader understand the process by which we are executing the StratusLab programme of work.

4.4 Build and Packaging

As mentioned before, all the StratusLab software, including its dependencies, are available via packages. These are constructed using Maven [4] and integrated in our continuous integration server Hudson. This means that the StratusLab software is continuously built, packaged and tested, which provides rapid feedback to the project members as the distribution progresses.

StratusLab will support RPM and DEB packages, popular with Red-Hat, Debian and SuSE-like operating system distribution.

4.5 Installation

All required dependencies for StratusLab will be expressed as dependencies in the StratusLab packages. Further, all dependencies will be on packages, either pointing to official and supported package repositories (preferred option), or in the case were these dependencies are not available in package format, or if the wrong versions are available, we will package and maintain these packages ourselves. All the StratusLab packages will be available in our dedicated YUM and APT package repositories.

The installation itself will be available via two mechanisms:

- **Automated**: Using the Quattor fabric management system, already the defact standard for several grid sites
- **Manual**: Using dedicated `stratus-*` system administrator commands

The source of metadata for both systems will be shared and maintained together with each release of StratusLab.

4.6 Configuration

Virtualisation technologies support a wide range and rich set of options and parameters. In order to facilitate instantiating virtual machines that will work with contextualisation and provide interoperability between the different StratusLab sites, we need to reduce the scope of options and better guide the user.

The configuration of the StratusLab system will be performed via a single configuration file, including parameters grouped in sections. No duplication will exist in the configuration file, reducing the chances of conflicts, which could result in a faulty system. Further, since certain parameters, for example the choice of the hypervisor or mechanism to transfer images from the OpenNebula front-end to the nodes (e.g. NFS, SSH), could result in installing different packages, some installation techniques could decide to make those choices up-front. This would have the consequence for system administrators of reducing complexity in installing the system, at the price of reduced flexibility.

4.7 Testing

Testing the StratusLab distribution is required to ensure that the advertised functionality works, and as we release new features and fix bugs, that we do not introduce regressions. The best way to ensure that each release of StratusLab is of the expected quality and that no regression is introduced is to ensure adequate engineering practices and a good test suite, built over each sprint, which covers the main features. In order to detect a mistake introduced by a commit quickly after the commit, the test suite must be able to run often. The test suite must therefore

be able to be executed unattended and automatically. However, a challenge in testing distributed systems, something made even more challenging in a cloud system designed to deploy distributed systems, is the setup required for the test suite to perform on a meaningful deployment of the system.

StratusLab already has a series of tasks automated using a continuous integration server called Hudson [22]. This server fires build, deployment and test tasks (called jobs in Hudson) soon after new or updated files are committed in our version control system.

Further, as more complex deployment scenarios are required, SlipStream (from SixSq) will be used to deploy complex multi-node systems automatically, without having to write special code for this. The execution of the SlipStream deployment scenario will also be integrated in Hudson jobs, which in turn will be triggered by commits in our version control system.

5 Component Architecture

Following from the analysis reported in Chapter 3, in this chapter we first establish the generic reference architecture required to implement the system, and then move on to choose components which will allow us to realise StratusLab v1.0.

5.1 Anatomy of a Cloud

The anatomy of an IaaS (Infrastructure as a Service) cloud consists of several main components. These components are illustrated in Figure 5.1 and described in Table 5.1.

These elements are required for building a complete cloud distribution able to fulfil the requirements and use cases identified in Chapter 3.

5.2 Reference Deployment Model

Figure 5.1 also presents a deployment of the components required to build an IaaS cloud. This proposed deployment, or assignment of components to sites and hosts, enables us to ensure that we can deploy the StratusLab software onto a functioning system. This *reference model* will be implied by our installation and configuration tools. This makes it possible for StratusLab to be easily installed and configured, with little up-front knowledge of its internal details, while giving the opportunity for advanced users to explore different deployment models.

All of the cloud components in Table 5.1 will be deployed as part of a standard site running StratusLab, except the appliance repository, credentials manager, and user tools. The user tools will be deployed on the user's work station. The other two are centralised services shared by several or all sites.

Table 5.1: *IaaS Cloud Components*

Hypervisors	running on top of the physical resources, virtualisation is provided by hypervisors in which virtual machines execute, including network, storage and compute
Virtual machine manager	core of the system, the manager orchestrates requests and manages the allocation of resources, as well as the life-cycle of running virtual images
Appliance repository	is the persistent source of virtual images, both machine and datasets
Cloud interface	that provides the users with a simple abstraction to manage VMs.
Administration tools	tools providing system administrator the means to install, configure and manage the cloud
User tools	tools for interacting with the different cloud services
Credentials manager	provides authentication and authorisation services to the StratusLab integrated services and users
Monitoring system	provides real-time visibility and information on resources on the cloud
Accounting system	provides historical and aggregated information on the utilisation of cloud resources
Cache	provides caching of large datasets, such as machine and dataset images, at different points in the infrastructure.

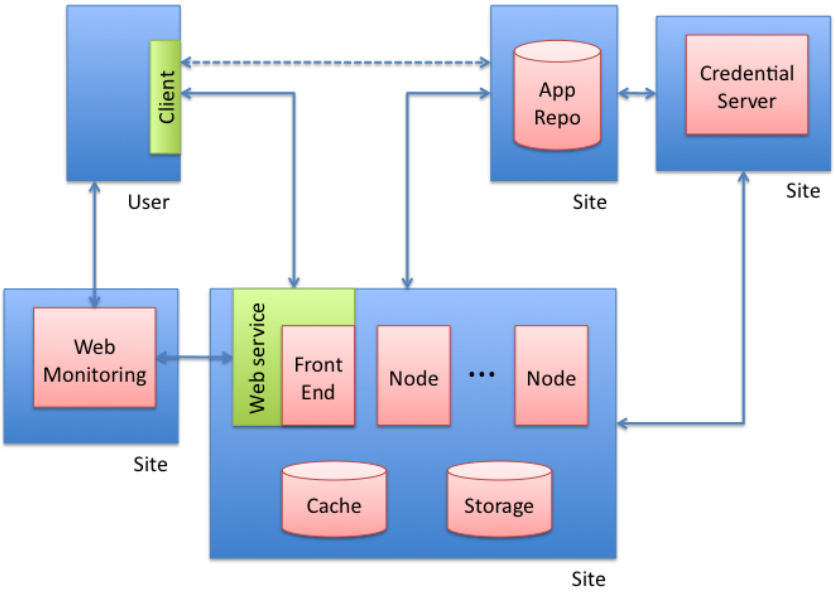

Figure 5.1: *High-level architecture.*

6 Selected Components

Chapter 5 identified several components that need to be part of the StratusLab distribution. This section reviews, analyses and selects candidates for these components. The result of this selection process is summarised hereafter, and illustrated in Figure 6.1 and described in Table 6.1.

As shown on Figure 6.1 OpenNebula is at the heart of the system and provides VM management. In order to grant remote access to the cloud, a set of client and web applications are deployed, such that only standard web protocols are used to monitor and control the cloud. The Appliance Repository is also shown, running from a remote site and shared by several sites. Storage and cache is also required at the site to provide image caching and storage.

These choices are explained in the following sections.

6.1 Virtualisation

Over recent years, a constellation of virtualisation products have emerged. So, a variety of hypervisors have been developed and greatly improved, most notably KVM, Xen and VMware. The current open source standards are KVM and Xen. It is too early to tell which of these two technologies will dominate, therefore, we have chosen to support both. The libvirt library provides a convenient front-end to these virtualisation, making it easier to integrate with both. Amazon standardises on Xen, while several other developments support KVM.

A feature important when using cloud systems to build, deploy and test cloud systems, is the ability to run virtual machines inside an already virtualised machine. This is possible using para-virtualised Xen machines inside KVM. Hence, adding to the need to support both types.

In order to support both virtualisation technologies, we need to be able to ensure that virtual machine images uploaded into the appliance repository are compatible with both.

As mentioned above, VMware is also taking an important role in the virtualisation market. While support for VMware is less important to our targeted communities, it should remain on our radar for future support.

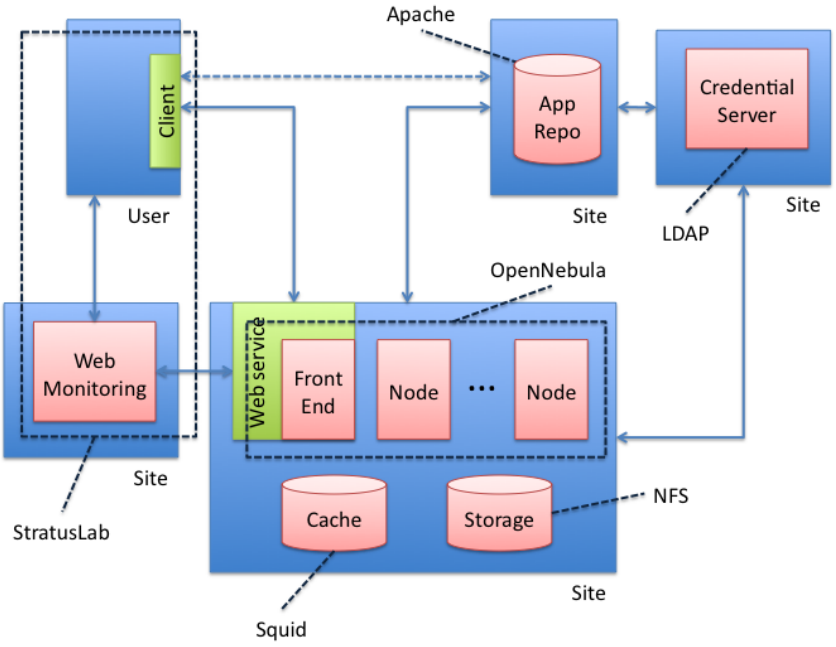

Figure 6.1: *High-level architecture with components assignment*

Table 6.1: *IaaS Cloud Components*

Hypervisors	KVM and Xen. Eventually VMware.
Virtual machine manager	OpenNebula
Appliance repository	Apache HTTP Server
Cloud interface	Custom interface to start with, inspired from Amazon EC2. Standard interfaces will follow
Administration tools	StratusLab specific development
User tools	StratusLab specific development
Credentials manager	LDAP
Monitoring system	StratusLab specific development, building on OpenNebula's XMLRPC feature
Accounting system	StratusLab specific development, building on an extended OpenNebula's XMLRPC feature
Cache	Squid

6.2 Virtual Machine Manager

Several 'Virtual Machine Manager' technologies that cover the functionality outlined in Chapter 5 above have appeared, like Platform VM Orchestrator, VMware DRS, or Ovirt. On the other hand, projects like Globus Nimbus [50], Eucalyptus [49] or OpenNebula [30] (developed by UCM in the context of the RESERVOIR project), or products like VMware vSphere, that can be termed cloud toolkits, can be used to transform existing infrastructure into an IaaS cloud with cloud-like interfaces.

Several of the above mentioned solutions are commercial, closed or focusing on a subset of the functionality required from SlipStream.

Key differentiating factors of OpenNebula with other commercial virtual infrastructure managers are its open and flexible architecture and interfaces, which enable its integration with any existing product and service in the Cloud and virtualisation ecosystem, and its support for building any type of Cloud deployment. Further, compared to other open source alternatives, OpenNebula provides superior functionality on a wider range of virtualisation technologies for building private and hybrid clouds. It has been designed to be integrated with any networking and storage solution and so to fit into any existing data centre. For these reasons, OpenNebula has been chosen as the VM manager for the StratusLab cloud distribution.

OpenNebula manages VMs and performs life-cycle actions by orchestrating three different management areas, namely: **networking** by dynamically creating local area networks (LAN) to interconnect the VMs and tracking the MAC addresses leased in each network; **image management** by transferring the VM images from an image repository to the selected resource and by creating on-the-fly temporary images; and **virtualisation** by interfacing with physical resource hypervisor, such as Xen or KVM, to control (e.g. boot, stop or shutdown) the VMs. Moreover, it is able to contact cloud providers to combine local and remote resources according to allocation policies.

In order to provide the virtual management solution we require, we need an open source and open architecture toolkit, scalable and available, backed-up by an active developer community, with which we can engage. Our choice is therefore OpenNebula.

In the following subsections, you will find further details on the capability of OpenNebula that StratusLab will leverage in order to provide the comprehensive cloud solution required by our user communities.

6.2.1 Computing

A VM within the OpenNebula system consists of:

- Capacity in terms memory and CPU.
- A set of NICs attached to one or more virtual networks (see Section 6.2.2).
- A set of disk images. In general, it could be necessary to transfer some of

these image files to/from the execution host (see Section 6.3).

- A state file (optional) or recovery file, that contains the memory image of a running VM plus some hypervisor specific information.

The above items, plus some additional VM attributes like the OS kernel and context information to be used inside the VM, are specified in a VM template file. OpenNebula manages VMs by interfacing with the physical resource virtualisation technology (e.g. Xen or KVM).

The scheduler module is in charge of the assignment between pending VMs and known hosts. The OpenNebula scheduling framework is designed in a generic way, so it is highly modifiable and can be easily replaced by third-party developments. This way, it can be used to develop virtual resource placement heuristics to optimise different infrastructure metrics (e.g. utilisation or energy consumption) and to fulfil grid service constraints (e.g. affinity of related virtual resources or SLA).

OpenNebula uses a set of managers to orchestrate the management of VMs. In turn, these managers are helped by a set of pluggable modules that decouple the managing process from the underlying technology, such as virtualisation hypervisors, operating systems, file transfer mechanisms or information services. These modules are called drivers in OpenNebula, and they communicate with the OpenNebula core using a simple ASCII protocol; thus simplifying the development of new drivers.

OpenNebula, virtualisation libraries and tools provide a very wide range and rich set of parameter settings. In order to reduce the range of parameters exposed to the system administrator in StratusLab, we are proposing to expose only a subset of parameters, taking into account reasonable assumptions on the setup.

6.2.2 Networking

VMs are the basic building blocks used to deliver IT services of any nature, from a computing cluster to the classic three-tier business application. In general, these services consists of several inter-related VMs, with a Virtual Application Network (VAN) being the primary link between them. OpenNebula dynamically creates these VANs and tracks the MAC addresses leased in the network to the service VMs. Other TCP/IP services such as DNS, NIS or NFS, are the responsibility of the service (i.e. the service VMs have to be configured to provide such services).

The physical hosts that will conform the fabric of the virtual infrastructures will need to have some constraints in order to effectively deliver virtual networks to the VMs. Therefore, from the point of view of networking, we can define our physical cluster as a set of hosts with one or more network interfaces, each of them connected to a different physical network.

Figure 6.2 shows two physical hosts with two network interfaces each, thus there are two different physical networks. There is one physical network that connects the two hosts using a switch, and another one that gives the hosts access to the public internet. This is one possible configuration for the physical cluster, and

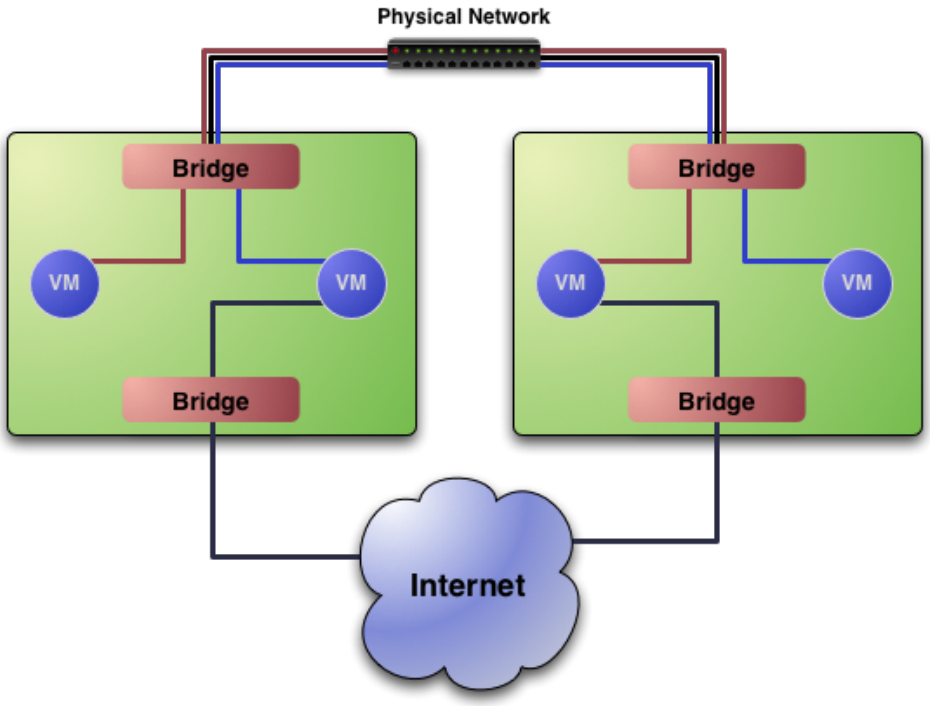

Figure 6.2: *Networking model.*

it is the recommended one since it can be used to make both private and public VANs for the VMs. Moving up to the virtualisation layer, we can distinguish three different VANs. One is mapped on top of the public internet network, and a couple of VMs take advantage of it. Therefore, these two VMs will have access to the internet. The other two (red and blue) are mapped on top of the private physical network. VMs connected to the same private VAN will be able to communicate with each other, otherwise they will be isolated and will not be able to communicate.

6.2.2.1 Virtual Network

OpenNebula allows for the creation of Virtual Networks by mapping them on top of the physical ones. All Virtual Networks are going to share a default value for the MAC prefix, set in the oned.conf file.

There are two types of Virtual Networks in OpenNebula:

- Fixed, which consists of a set of IP addresses and associated MACs, defined in a text file.
- Ranged, which allows for a definition supported by a base network address and a size, either as a number or as a network class (B or C).

Other approaches, like configuring a DHCP server for the datacentre, are also possible.

6.2.2.2 IP Address Assignment

Using OpenNebula, we have great flexibility in terms of network configuration. By default, during the installation of StratusLab, two virtual networks are created: a private and a public network. Also by default, all virtual machines started on the site will be visible on these networks. Further, we can request a specific IP address to be associated to a virtual machine, a feature similar to the Amazon EC2 Elastic IP. Furthermore, custom networks can be created, such that new VM instances can be configured and attached to these.

6.2.3 Extensibility

Demonstrating our growing experience with OpenNebula, several members of the StratusLab project are now contributors to the OpenNebula project. The latest development on the Web Monitor, which adds the monitoring capability of an OpenNebula installation over the web, showed the power of the XMLRPC API exposed by OpenNebula. Being able to extend more easily the data available via that interface will improve the rate at which we can extend and better integrate OpenNebula with existing and new services. The availability of the XMLRPX API is very important, since most cloud interfaces (like OGF OCCI, AWS EC2 or VMware vCloud APIs) provide basic monitoring of virtualised resources, while this API provides extended monitoring of virtualised and physical resources.

Other extensibility mechanisms are likely to be identified over the course of the project, which we hope we will be able to exploit in collaboration with the main

developers of these systems.

6.2.4 Image Management

VMs are supported by a set of virtual disks or images, which contain the OS and any other additional software needed by the service. The image repository (see Section 6.4) holds the base image of the VMs. Also, the images can be shared through NFS between all the hosts, or transferred through SSH between them.

OpenNebula uses the following concepts for its image management model:

- Image Repository, refers to any storage medium, local or remote, that holds the base images of the VMs. An image repository can be a dedicated file server or a remote URL from an appliance provider, but they need to be accessible from the OpenNebula front-end. See Section 6.4 for more details about the image repository to be used in StratusLab.
- Virtual Machine Directory, is a directory on the cluster node where a VM is running. This directory holds all deployment files for the hypervisor to boot the machine, checkpoints, and images being used or saved, all of them specific to that VM. This directory should be shared for most hypervisors to be able to perform live migrations.

6.3 Storage

Transient and persistent storage are required in a cloud. This section focuses on the need for persistent storage and caching to improve performance in managing transient data, like images.

6.3.1 Persistent Storage

It is important to be able to persist data independently from running virtual images. Amazon has a feature called Elastic Block Store, which allows a cloud user to define a persistent volume, attach it to a running instance via a device, which can be mounted on the local file system. Even if the instance stops or crashes, the data behind this volume is persisted and can be re-attached to a different running instance.

This feature also allows cloud users to be able to create complex data-sets and manage them independently from the machine images. For example, a data-set might be maintained for testing, which can be applied to different versions of a system and composed at runtime, without being ‘baked’ into each image needing this data-set.

This feature can be achieved in StratusLab using the current contextualisation feature of OpenNebula, where this disk image is saved and made available to the user via the Appliance Repository (Section 6.4). If true persistence is required (data surviving beyond a physical node crash) the disk image could be associated with a distributed or replicated file system. Performance and network consideration have to be take into account for this advanced feature. Further, other options also

exist, which we will explore for StratusLab v2.0 in collaboration with our user communities.

6.3.2 Caching

In order to save network bandwidth when transferring large image files (even compressed) a caching mechanism can be used. Technologies such a Squid [44] can provide a caching feature in a controlled manner. For example, each site could setup a caching system, where the front-end and nodes have a configured cache, thus saving bandwidth inside the site and between the site and the Appliance Repository.

6.4 Appliance Repository

The StratusLab Appliance Repository requires to provide remote access via the web. This can simply be implemented using the Apache Web Server, configured to authenticate with the project LDAP server. While this is a convenient and well supported solution, any web server, capable of handling large files could be used instead.

The repository needs to be organised and structured. Here we propose to adopt the convention provided by the Maven repository structure. Here is an example of this convention: https://appliances.stratuslab.org/images/base/ubuntu-10.04-i686-base/1.0/.

To seed the foundation of trust between authors of virtual images and system administrators running sites, metadata must be available describing the content and origin of the virtual image. This metadata is provided in the form of a manifest file (`ubuntu-10.04-i686-base-1.0.img.manifest.xml`), describing an image (`ubuntu-10.04-i686-base-1.0.img.gz`), and placed in the same 'folder' in the repository. To reduce the file size and bandwidth when accessing images, the image file should be compressed as a single file (i.e. not archived).

The currently proposed manifest format is an XML document. Here is an example of a manifest file:

```
<manifest>
    <created>2010-08-18 20:34:28.334763</created>
    <type>base</type>
    <version>1.0</version>
    <arch>i686</arch>
    <user>A. Joseph</user>
    <os>ubuntu</os>
    <osversion>10.04</osversion>
    <compression>gz</compression>
</manifest>
```

As we explore features like the Persistent Storage (see Section 6.3.1), we might then extend the definition of 'Appliance', which could mean to also add data images

to the appliance repository.

6.5 Virtual Machine Contextualisation

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 [20], or accessing a disk image with the context data for the VM (Open Virtual Format, OVF [14], recommendation).

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

With this in place, each VM in the Appliance Repository are guaranteed to work correctly in all StratusLab sites, and will work with the StratusLab command-line tools, such as `stratus-run-instance` and `stratus-create-image`.

6.6 Tools

In order to access, monitor and control the cloud, both from an end-user and a system administrator point of view, a set of tools must be available. While web solutions will be discussed in the Section 6.8 for monitoring and Section 6.9 for accounting, this subsection focuses on remote command-line tools. The tools supporting installation and configuration of the cloud itself are briefly described in Sections 4.5 and 4.6.

Several of the tools presented in this section have already been developed as part of the first sprints of the project, in order to validate several assumptions made for the StratusLab architecture. This explains why some of the descriptions describe future work, while others describe existing results.

6.6.1 Virtual Machine Execution

Chapter 3 clearly illustrated the need by the users to be able to remotely manage virtual machines, as is standard in public clouds such as Amazon EC2. To comply with this requirement, a set of command-line tools are required. Already, early versions of the StratusLab distribution include, for example, the `stratus-create-instance` command (the name is likely to change in the near future), which allows the user to remotely request the instantiation of new virtual machines by a given site.

These types of commands can easily be built using the OpenNebula XML-RPC mechanism. This is another strong point in favour of OpenNebula in terms of

openness. The command provides all the necessary information for OpenNebula to download the image from the Appliance Repository, allocate a node for running the instance, create the new instance and contextualise it. The monitoring commands, also provided in the form of `stratus-describe-instance` and `stratus-describe-node` allow the user to monitor his or her virtual machines. This means that no SSH access to the OpenNebula front-end is required in order to let users start new virtual machines.

6.6.2 Virtual Image Creation

In clouds, the fundamental building block for users is the virtual machine. We therefore need to facilitate the creation of virtual machine images, which will work with our contextualisation (see Section 6.5). This section describes several aspects of this creation and how StratusLab can provide support to users for image creation.

6.6.2.1 Base Images

To facilitate the creation of new virtual machine images, StratusLab is developing and maintaining a set of base images. These images are meant as a starting point for building more specialised images. These images typically only contain a clean standard operating system installation, as well as the required configuration to support the StratusLab contextualisation strategy. For example, IP addressing will work according to the StratusLab convention, a user defined public key will be installed in the VM during contextualisation such that users can ssh into their instances.

StratusLab maintains this set of base images in the StratusLab public Appliance Repository (https://appliances.stratuslab.org/images/base).

6.6.2.2 Custom Image Creation

While StratusLab provides online documentation for users to create their own images from scratch (see Subsection 6.6.2.1), StratusLab also provide a simpler way to customise images. The command-line tool `stratus-create-image` offers a convenient way to augment an existing image (i.e. a base image or another image fulfilling the contextualisation requirements). Following the SlipStream [42] proposed convention, this command takes a list of packages to be installed as well as a script to further configure the machine. Once the machine is successfully configured, the command saves the image and uploads it, alongside the image manifest, to the appliance repository. From this point on, the image can be used as any standard StratusLab image.

The advantage of this technique is that the image creation process is standardised and will eventually be reproducible for other clouds, thus taking an important step towards cloud interoperability.

6.7 User Management

All services used by StratusLab must be integrated with a single identity system. OpenNebula is already being 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 need to engage is provided by the ARGUS [13] project, which develops a solution used in the grid world to provide authorisation decisions over distributed services.

6.8 Monitoring

In order to track and monitor the state of the cloud, we need monitoring tools. OpenNebula already provides local monitoring command-line tools, such as `onehost` and `onevm`. While useful, these tools only work locally on the front-end machine. Meanwhile, OpenNebula also provides an XMLRPC API. StratusLab builds on top of this API to provide both a set of command-line tools (`stratus-describe-node` and `stratus-describe-instance`), as well as a web application.

The command-line tools are packaged together, while the web monitor is packaged individually.

These tools are useful in their own right, however, as cloud sites are interconnected via the grid, we will also need more powerful tools able to bridge the boundaries of single cloud sites, both for monitoring and accounting (see Section 6.9 for details). Grid monitoring tools already provides cross-site monitoring with which StratusLab monitoring of the cloud layer will have to integrate.

6.9 Accounting

Both cloud and grid services require adequate tracking of resource usage, per user, and in the grid context per VO. While OpenNebula maintains several key accounting metrics, they are not readily available (see Section 6.2.3) to accounting services. In a grid context, it is important to be able to extract these values and aggregate them. It is therefore important to make the raw accounting parameters available via a convenient API, such that they can be securely queried and retrieved.

These metrics, as for accounting values, must be integrated with existing grid accounting services.

7 State of the Art

This chapter is an updated version of the state-of-the-art section we wrote for the StratusLab proposal.

The potential benefits that cloud and virtualisation technologies can bring to current e-Infrastructures require a common framework that bridge grid and cloud computing models. Various solutions have been proposed to take advantage of these new technologies in a grid environment, from its direct application to encapsulate the execution of each job, to the advance provisioning of Virtual Organisation clusters. In Section 7.1 below, we summarise the current state of the art in these areas and then in Section 7.2 we detail the status of the ecosystem of cloud technologies. Finally, Section 7.3 shows a commercial perspective.

7.1 Application of Virtualisation and Cloud to Grid Computing

In the last decade we have witnessed the consolidation of several transcontinental grid infrastructures that have achieved unseen levels of resource sharing. In spite of this success, current grids face several obstacles that limit their efficiency, namely:

- An increase in the cost and length of the application development and porting cycle. New applications have to be tested in a great variety of environments where the developers have limited configuration capabilities.
- A limitation on the effective number of resources available to each application. Usually different VOs require different software configurations, so an application can be only executed on those sites that support the associated VO. Moreover, the resources devoted to each VO within a site are usually static and cannot be adapted to the VO's workload.
- An increase in the operational cost of the infrastructure. The deployment, maintenance and distribution of different configurations involves specialised, time consuming and error prone procedures. Even worse, new organisations joining a grid infrastructure needs to install and configure an ever-growing middleware stack.

This situation often leads to a struggle between the users, who need more control over their execution environments, and grid operators, who want to limit

the heterogeneity of the infrastructure. As a result, several alternatives to reconcile both positions have been explored in the past. For example, the SoftEnv project [45] is a software environment configuration system that allows the users to define the applications and libraries they need. Another common solution is the use of a custom software stack on top of the existing middleware layer, usually referred as pilot-jobs. For example, MyCluster [51] creates a Condor or Sun Grid Engine cluster on top of TeraGrid services; and similarly over other middleware we may cite DIRAC [47], glideinWMS [41] or PanDa [46].

Additionally, several projects have investigated the partitioning of a distributed infrastructure to dynamically provide customised independent clusters. For example, COD (Cluster On Demand) [11] is a cluster management software which dynamically allocates servers from a common pool to multiple virtual clusters. Similarly, the VIOcluster [40] project supports the dynamically adjustment of the capacity of a computing cluster by sharing resources between peer domains.

However the most promising technology to provide virtual organisations (VO) with custom execution environments is virtualisation. The dramatic performance improvements in hypervisor technologies has made it possible to experiment with virtual machines (VM) as basic building blocks for computational platforms. Several studies (see for example [53] and [52]), reveal that the virtualisation layer has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters.

The first works in this area integrated resource management systems with VMs to provide custom execution environments on a per-job basis. For example Dynamic Virtual Clustering [15] and XGE [17] for MOAB and SGE job managers respectively. These approaches only overcome the configuration limitation of physical resources since VMs are bound to a given resource and only exist during job execution. A similar approach has been implemented by UCM at Grid level using the Globus GridWay Metascheduler [39].

More general approaches involve the use of virtual machines as workload units, which implies the change in paradigm from building grids out of physical resources to virtualised ones. For example, the VIOLIN project proposes a novel alternative to application-level overlays based on virtual and isolated networks created on top of an overlay infrastructure. Also I. Krsul et al. [29] propose the use of a service (VMPlant) to provide the automated configuration and creation of VMs that can subsequently be cloned and instantiated to provide homogeneous execution environments across distributed grid resources. On the other hand, the In-VIGO [1] project adds some virtualisation layers to the classical grid model, to enable the creation of dynamic pools of virtual resources for application-specific grid-computing. Also in this line, several studies have explored the use of virtual machines to provide custom (VO-specific) cluster environments for grid computing. In this case, the clusters are usually completely built up of virtualised resources, as in the Globus Nimbus project [19], or the Virtual Organisation Clusters (VOC) proposed in [32]. UCM has explored a hybrid model, so the cluster combines physical, virtualised and cloud resources [30].

It is important to note that the previous works also highlight that the use of virtualisation in grid environments can greatly improve the efficiency, flexibility and sustainability of current production grids. Not only by extending the classical benefits of VMs for constructing cluster, e.g. consolidation or rapid provisioning of resources [33], [16]; but also grid-specific benefits, e.g. support to multiple VOs, isolation of workloads and the encapsulation of services, as has been published by SixSq [7].

Finally, some initiatives, like the collaboration between the RESERVOIR and EGEE projects, with StratusLab initiative participation, are studying a cloud-like provisioning model for grid-sites. In this way, a grid site exposes a typical cloud interface to instantiate and control virtual machines. This would allow to access grid Computing resources as cloud providers.

7.2 State of Cloud Computing Technologies and Services

Cloud Computing was arguably first popularised in 2006 by Amazon's Elastic Compute Cloud, which started offering virtual machines (VMs) for $0.10/hour using both a simple web interface and a programmer-friendly API. Although not the first to propose a utility computing model, Amazon EC2 contributed to popularising the *Infrastructure as a Service* (IaaS) paradigm, which became closely tied to the notion of Cloud Computing. An IaaS cloud enables on-demand provisioning of computational resources, in the form of VMs deployed in a cloud provider's datacenter (such as Amazon's), minimising or even eliminating associated capital costs for cloud consumers, allowing capacity to be added or removed from their IT infrastructure in order to meet peak or fluctuating service demands, while only paying for the actual capacity used.

Over time, an ecosystem of providers, users, and technologies has coalesced around this IaaS cloud model. More IaaS cloud providers, such as GoGrid, FlexiScale, and ElasticHosts have emerged. A growing number of companies base their IT strategy on cloud-based resources, spending little or no capital on machines to manage their own IT infrastructure (see http://aws.amazon.com/solutions/case-studies/ for several examples). Other providers offer products that facilitate working with IaaS clouds, such as rPath's rBuilder, which allows dynamic creation of software environments to run on a cloud.

In general, an IaaS cloud consists of three main components, namely: a virtualisation layer on top of the physical resources including network, storage and compute; the virtual infrastructure manager (VIM) that controls and monitors the VMs over the distributed set of physical resources; and a cloud interface that provides the users with a simple abstraction to manage VMs. In recent years a constellation of technologies that provide one or more of these components have emerged. So, a variety of hypervisors have been developed and greatly improved, most notably KVM, Xen and VMware. Also several VIM technologies that cover the function-

ality outlined above have appeared, like Platform VM Orchestrator, VMware DRS, or Ovirt.

On the other hand, projects like Globus Nimbus [27], Eucalyptus [34] or OpenNebula[30] (developed by UCM in the context of the RESERVOIR project), or products like VMware vSphere, that can be termed cloud toolkits, can be used to transform existing infrastructure into an IaaS cloud with cloud-like interfaces. Key differentiating factors of OpenNebula compared with other commercial virtual infrastructure managers are its open and flexible architecture and interfaces, which enable its integration with any existing product and service in the Cloud and virtualisation ecosystem, and its support for building any type of Cloud deployment. Further, compared to other open-source alternatives, OpenNebula provides superior functionality on a wider range of virtualisation technologies for building private and hybrid clouds. These technologies include a cloud interface like Amazon EC2 or the vCloud API; and the functionality needed to orchestrate the virtual infrastructure. Finally we would like to note that there are some initiatives to standardise the Cloud Interface component (see the work of the OCCI WG, co-founded by UCM, in OGF); or to provide interoperability between different interfaces like Red Hat's Delta-cloud project.

On top of this IaaS layer some providers, such as Elastra and Rightscale, focus on deploying and managing services, including web and database servers that benefit from the elastic capacity of IaaS clouds, allowing their clients to provision services directly, instead of having to provision and setup the infrastructure themselves. In this case, the users may need to configure their services based on specific parameters that are only known at deployment time. Thus, advanced service contextualisation tools are required to ease the automatic deployment of services that are to be deployed on top of virtualised infrastructures in a shared security context.

Automatic service deployment requires a configuration phase performed simultaneously to the deployment of the service, or to deploy fully configured images and adjust the configuration of context-sensitive applications after deployment. These options may result in a long deployment time for nontrivial systems, which hinders the responsiveness and the scaling potential of the deployed services [28]. Also, the deployment time configuration of a service presents the same challenges that VMs customisation [21].

In order to start these automatic management features, an appropriate description language is needed. Nephele (developed by TID), is an open-source implementation of a service manager tool located on top of the virtual infrastructure management solutions. Nephele has been developed and continues to be enhanced as part of the European Union's RESERVOIR Project. However, Nephele needs extensions to fully cover the requirements above and provide an integral scalability to the deployed grid services.

Earlier this year, NASA and Rackspace launched a new project, called OpenStack [35]. This new project created significant buzz on the web. Part of it was caused by one of the reasons why NASA decided to create OpenStack: a rift with Eucalyptus. Following scalability issues, which have also been observed by Stra-

tusLab partners, NASA was unable to contribute improvements to the Eucalyptus software stack, uncovering that not all of Eucalyptus is open source [26]. OpenStack is interesting for StratusLab since both projects share similar goals. StratusLab will therefore engage with OpenStack in order to share, and eventually collaborate, since both projects could to be the only truly open source cloud distribution to date. However, the architecture of the OpenStack, and its open source nature, is not yet clear. Further, according to Foran, OpenStack has very challenging scalability goals, with millions of physical hosts, and tens of million of managed virtual machines. These goals exceed the ones from StratusLab.

7.3 Commercial Perspective

Despite the fact that cloud computing has become a standard resident of the distribution computing world, with the launch of EC2 by Amazon in 2006, commercial offerings have dominated the scene. It is only recently that projects such as OpenNebula, and to a certain extent Eucalyptus, have made progress in providing open source software for delivering cloud capabilities.

This dominance by the commercial market is clearly illustrated by a couple of articles published by Jeff Vance [24] [25]. While the accuracy of the report in terms of ranking and the presence of some companies can be challenged, the overall message makes it clear that, at least in North-America, cloud computing is mainly a private sector affair, with largely proprietary software stacks. Large scale deployment of these solutions have not been reported in this survey, which again would probably impact the results. Here are some of the names reported in these two articles as the main players:

- IBM
- AT&T
- Elastra
- OpSource
- Cisco
- Enomaly
- Amazon
- Salesforce.com
- Google
- Microsoft
- CA
- Rackspace

- Eucalyptus Systems
- Terremark
- GoGrid
- RightScale

It is important to note that this study is not limited to IaaS type clouds. It will be interesting to better understand, in the future, what these players plan to use in terms of underlying software and the standards they require.

8 Grid / Cloud Technology Gap

Although the virtualisation eco-system is now rich and reasonably mature, a gap still exists in order to integrate a single IaaS cloud distribution, accessible from outside the infrastructure, over a well-defined service API. This is critical since, for example, in a grid context users cannot be granted direct remote access to the machines running the virtualisation management engine.

Further, in order to expose a cloud-like API to grid users, an integration of this API is required between the grid and cloud layer, as well as an adaptation of the grid API such this new functionality is available to the users. StratusLab will explore, in partnership with user communities and grid middleware providers, how can this be best realised.

The creation and contextualisation of virtual machine images is also an important aspect where we need to fill a gap. This is important in order to be able to reuse virtual images between sites running StratusLab. 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 in the first place, and eventually develop solutions for letting users and sites utilise other cloud services beyond the StratusLab frontier, including public and commercial clouds.

Another missing functionality that requires our attention is that existing grid sites are not able to express the required scaling actions and reconfigurations. StratusLab will develop a new framework for service elasticity, also referred to as autoscaled services, which will be exploitable by grid and VO services. Further, SLA-powered services will also be explored, such that service developers can include new semantics when deploying their services in the cloud. Also, placement heuristics to optimise different infrastructure metrics and to fulfil grid service constraints will be evaluated.

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). We will also explore provisioning models (e.g. pay per use) when, for example, a site uses a hybrid cloud strategy to offload peak requests to another cloud in order to honour an elastic SLA.

Finally, StratusLab will extend the resource sharing capabilities of a grid site

by adding new grid-aware cloud interfaces to provide a Grid as a Service; and these interfaces will be integrated with existing Grid services. Scheduling policies and heuristics will be extended to work on a multi-cloud environment.

'Advanced' cloud features as described here (e.g. auto-scaling, SLA-powered provisioning, elastic response due to hybrid model) could lead, in turn, to a new breed of grid and VO services, such that a simpler, more reliable and resilient infrastructure can be offered to European researchers.

9 Summary

This document presents the architecture of the StratusLab v1.0 distribution. As explained throughout the document, including in the analysis of the requirements, the choices of components for the implementation of StratusLab, the review of the state-of-the-art and the gap analysis between grid and cloud, cloud computing is a very dynamic topic. This document aims at laying the foundation for the architecture of StratusLab v1.0 distribution. Building on the knowledge of implementing StratusLab, we will undoubtedly learn and gather feedback from our user communities using early versions of the distribution. This document will be updated at PM15 with D4.4, which will add elements required for StratusLab v2.0.

We summarised the requirements and use-cases gathered by WP2 with surveys performed during the summer 2010. Adding our collective knowledge, we then analysed these requirements to formulate a reference component architecture of the StratusLab v1.0 distribution.

We also described our agile development process and engineering practices needed to face the challenge of delivering StratusLab in a changing and distributed world. While unusual in FP7 projects, this agile method has already proven to be effective and will be improved throughout the execution of the project.

The choices of components for the implementation of the StratusLab architecture was presented in Chapter 5 where components were reviewed and selected following analysis. Where no existing solution were identified, fulfilling the StratusLab requirements, custom development was then identified for the project to realise.

Chapter 7 presented an updated version of the state-of-the-art that we produced in the project proposal, which also fed into the component analysis and selection that was presented in Chapter 6.

Gaps, but also opportunities for synergy, between grid and cloud were discussed in Chapter 8, identifying work to be done to better integrate grid and cloud, with a short and long term perspective.

This architecture, we believe forms solid foundations on which the StratusLab project can base its incremental releases in order to provide system administrators and resource providers the functionality that will enable the efficient exploitation of computing resources for the provision of grid services. Reusing best-of-breeds in modern software and technology, StratusLab will capitalise on the inherent attributes of cloud computing and virtualisation 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 utilisation of the resource provider's physical infrastructure and facilitate the day-by-day tasks of the grid site administrator.

Glossary

Appliance	Virtual machine containing preconfigured software or services
Appliance Repository	Repository of existing appliances
DCI	Distributed Computing Infrastructure
EGEE	Enabling Grids for E-sciencE
EGI	European Grid Infrastructure
EGI-TF	EGI Technical Forum
Front-End	OpenNebula server machine, which hosts the VM manager
GPFS	General Parallel File System by IBM
Hybrid Cloud	Cloud infrastructure that federates resources between organizations
IaaS	Infrastructure as a Service
Instance	see Virtual Machine / VM
iSGTW	International Science Grid This Week
Machine Image	Virtual machine file and metadata providing the source for Virtual Images or Instances
NFS	Network File System
NGI	National Grid Initiative
Node	Physical host on which VMs are instantiated
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
Virtual Machine / VM	Running and virtualized operating system
VM	Virtual Machine
VO	Virtual Organization
VOBOX	Grid element that permits VO-specific service to run at a resource center
Web Monitor	Web application providing basic monitoring of a single StratusLab installation
Worker Node	Grid node on which jobs are executed

References

[1] S. Adabala, V. Chadha, P. Chawla, et al. From virtualized resources to virtual computing grids: the In-VIGO system. *Future Generation Computer Systems*, 21(6):896–909, June 2005.

[2] Amazon. Amazon Web Services. Online resource. http://aws.amazon.com.

[3] Apache. Apache Hadoop. Online resource. http://hadoop.apache.org.

[4] Apache Software Foundation. Apache maven. Online resource., 2010. http://maven.apache.org/.

[5] Atlassian Pty Ltd. Greenhopper plugin. Online resource., 2010. http://www.atlassian.com/software/jira/.

[6] Atlassian Pty Ltd. Jira. Online resource., 2010. http://www.atlassian.com/software/jira/.

[7] M. Bégin. An EGEE Comparative Study: Grids and Clouds - Evolution or Revolution. Technical report, EGEE-III NA1, 2008. Available at http://edms.cern.ch/file/925013.

[8] U. Berkeley. BOINC. Online resource. http://boinc.berkeley.edu.

[9] Canonical Ltd. Ubuntu linux. Online resource., 2010. http://www.ubuntu.com/.

[10] centos.org. Centos. Online resource., 2010. http://www.centos.org/.

[11] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic Virtual Clusters in a Grid Site Manager. In *Proc. 12th Intl. Symp. High Performance Distributed Computing (HPDC 2003)*, June 2003.

[12] cohesiveFT. VPN-Cubed. Online resource. http://www.cohesiveft.com/vpncubed.

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

[14] DMTF. Virtualization Management Initiative (VMAN). Online resource. http://www.dmtf.org/initiatives/vman_initiative.

[15] W. Emeneker, D. Jackson, J. Butikofer, and D. Stanzione. Dynamic Virtual Clustering with Xen and Moab. In *Proc. Frontiers of High Performance Computing and Networking, ISPA 2006 Workshops*, volume 4331 of *Lecture Notes in Computer Science*, pages 440–451, 2006.

[16] W. Emeneker and D. Stanzione. Dynamic virtual clustering. *IEEE Cluster*, Sept. 2007.

[17] N. Fallenbeck, H. Picht, M. Smith, and B. Freisleben. Xen and the Art of Cluster Scheduling. In *Proc. 1st Intl. Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*, 2006.

[18] Fermilab, CERN, others. Scientific linux. Online resource., 2010. https://www.scientificlinux.org/.

[19] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual clusters for grid communities. In *Proc. 6th IEEE Intl. Symp. Cluster Computing and the Grid (CCGrid 2006)*, May 2006.

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

[21] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, L. Vaquero, , and M. Wusthoff. Service Specification in Cloud Environments Based on Extensions to Open Standards. In *Proc. 4th Intl. Conf. Communication System Software and Middleware*, 2009.

[22] Hudson Labs. Hudson. Online resource., 2010. http://hudson-ci.org/.

[23] E. Huedo, R. Montero, and I. Llorente. A modular meta-scheduling architecture for interfacing with pre-WS and WS Grid resource management services. *Future Generation Computer Systems*, 23(2):252–261, 2007.

[24] Jeff Vance. Cloud computing leaders: Ten to watch. Online resource., 2010. http://itmanagement.earthweb.com/netsys/article.php/11075_3887691_1/Cloud-Computing-Leaders-Ten-to-Watch.htm.

[25] Jeff Vance. Ten cloud computing leaders. Online resource., 2010. http://itmanagement.earthweb.com/netsys/article.php/12297_3884311_1/Ten-Cloud-Computing-Leaders.htm.

[26] Joseph Foran. Comparing open source cloud platforms: Openstack versus eucalyptus. Online resource., 2010. http://searchcloudcomputing.techtarget.com/generic/0,295582,sid201_gci1518097,00.html.

[27] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming*, 13(4):265–276, 2005.

[28] K. Keahey and T. Freeman. Contextualization: Providing One-Click Virtual Clusters. In *IEEE 4th Intl. Conf. eScience*, pages 301–308, 2008.

[29] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. VMPlants: Providing and managing virtual machine execution environments for grid computing. In *Proc. 2004 ACM/IEEE Conference on Supercomputing*, 2004.

[30] I. M. Llorente, R. Moreno-Vozmediano, and R. S. Montero. Cloud Computing for On-Demand Grid Resource Provisioning. In *Proc. High Performance Computing workshop 2008, High Speed and Large Scale Scientific Computing*, volume 18 of *Advances in Parallel Computing*, pages 177–191. IOS Press, 2009.

[31] M. Matos, A. Sousa, J. Pereira, and R. Oliveira. Clon: overlay network for clouds. In *WDDM '09: Proceedings of the Third Workshop on Dependable Distributed Data Management*, pages 14–17, New York, NY, USA, 2009. ACM.

[32] M. Murphy, B. Kagey, M. Fenn, and S. Goasguen. Dynamic Provisioning of Virtual Organization Clusters. In *Proc. 9th IEEE Intl. Symp. Cluster Computing and the Grid (CCGrid 2009)*, 2009.

[33] H. Nishimura, N. Maruyama, and S. Matsuoka. Virtual clusters on the fly - fast, scalable, and flexible installation. In *Proc. 7th IEEE Intl. Symp. Cluster Computing and the Grid (CCGRID 2007)*, May 2007.

[34] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-source Cloud-computing System. In *Cloud Computing and Its Applications*, October 2008.

[35] OpenStack Cloud Software. Online resource. http://www.openstack.org.

[36] OpenVPN. Online resource. http://openvpn.net.

[37] Quattor Community. Quattor. Online resource., 2010. http://quattor.org/.

[38] Red Hat, Inc. Red-hat linux. Online resource., 2010. http://www.redhat.com/.

[39] M. Rodriguez, D. Tapiador, J. Fontan, E. Huedo, R. S. Montero, and I. M. Llorente. Dynamic Provisioning of Virtual Clusters for Grid Computing. In *Proc. 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 2008), in conjunction with Euro-Par 2008*, volume 5415 of *Lecture Notes in Computer Science*, pages 23–32, 2009.

[40] P. Ruth, X. Jiang, and S. G. D. Xu. Virtual Distributed Environments in a Shared Infrastructure. *IEEE Computer, Special Issue on Virtualization Technologies*, May 2005.

[41] I. Sfiligoi. Making science in the Grid world: using glideins to maximize scientific output. In *Proc. Nuclear Science Symposium Conference Record*, pages 1107–1109, 2007.

[42] SixSq Sàrl. Slipstream. Online resource., 2010. http://www.sixsq.com/products.

[43] SPI. Debian. Online resource., 2010. http://www.debian.org/.

[44] Squid Project. Squid. Online resource., 2010. http://www.squid-cache.org/.

[45] Teragrid User Support. Managing Your Software Environment. Available at https://www.teragrid.org/web/user-support/environment. Accessed April 2010.

[46] The PanDA Production and Distributed Analysis System. Available at https://twiki.cern.ch/twiki/bin/view/Atlas/Panda. Accessed April 2010.

[47] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees. DIRAC: A Scalable Lightweight Architecture for High Throughput Computing. In *Proc. 5th IEEE/ACM Intl. Workshop on Grid Computing (GRID'04)*, pages 19–25, 2004.

[48] M. Tsugawa and J. Fortes. A virtual network (ViNe) architecture for grid computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 10 pp., 25-29 2006.

[49] S. B. University of California. The EUCALYPTUS open source software infrastructure. Online resource. http://eucalyptus.cs.ucsb.edu.

[50] University of Chicago. Globus Nimbus. Online resource. http://workspace.globus.org.

[51] E. Walker, J. P. Gardner, V. Litvin, and E. Turner. Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment. In *Proc. IEEE Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2006)*, July 2006.

[52] L. Youseff, K. Seymour, H. You, J. Dongarra, and R. Wolski. The Impact of Paravirtualized Memory Hierarchy on Linear Algebra Computational Kernels and Software. In *Proc. High Performance Distributed Computing (HPDC)*, June 2008.

[53] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Paravirtualization for HPC Systems. In *Proc. Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), in conjunction with Intl. Symp. Parallel and Distributed Processing and Application (ISPA 2006)*, Dec. 2006.