FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Web System For Creating And Managing Virtual High Performance Computing Environments

Pedro Adriano Pessoa Teixeira

Master in Informatics and Computing Engineering

Supervisor: Tito Carlos Soares Vieira (M.Sc.)

19^{th} July, 2011

Abstract

Cloud Computing is a subject that has recently become very discussed in the academic community. FEUP - Faculty of Engineering from the Oporto University - is no exception. Its informatic center CICA - Professor Correia de Araújo Informatic Center - is currently developing a private Cloud project. This project is split into two different projects and this document covers the front-end of the system.

This projects objectives consist in developing a web portal capable of creating and managing virtual high performance computing environments over private cloud computing infrastructures.

In this document, the state of the art of Cloud Computing related subjects is discussed, and the other half of this project (developed by another student as his Master thesis) is analyzed in various sections.

Resumo

A computação em nuvem é um tema que recentemente tem vindo a ser muito debatido na comunidade académica. A FEUP - Faculdade de Engenharia da Universidade do Porto - não é excepção. O seu centro de informática CICA - Centro de Informática Prof. Correia de Araújo - encontra-se neste momento a desenvolver um projecto de computação em nuvem privada. O projecto está subdivido em dois projectos e este documento foca-se no front-end do sistema.

Este projecto consiste em desenvolver um portal web capaz de criar e gerir ambientes de computação de elevado desempenho virtuais sobre infraestruturas de computação em nuvem privadas.

Neste documento é exposto o estado da arte da computação em nuvem e temas relacionados, sendo que a outra parte do projecto é analisada em diversas secções do documento.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations and Symbols

CICA	Centro de Informática Prof. Correia de Araújo
CPU	Central Processing Unit
FEUP	Faculdade de Engenharia da Universidade do Porto
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
PHP	PHP: Hypertext Processor
VD	Virtual Disk
VM	Virtual Machine
VO	Virtual Organization
VW	Virtual Workspace
WWW	*World Wide Web*

Chapter 1

Introduction

Firstly, it is necessary to understand that this project is part of a bigger project currently under development at CICA - Centro de Informática Prof. Correia de Araújo - at FEUP - Faculdade de Engenharia da Universidade do Porto - and as such, themes related to the whole project (and not just the one covered by this document) must be subject to analysis in this document.

Two areas of computing - Cloud and Grid - are initially discussed, as they establish the basis for the project. Other subjects are also covered, since the project is wide enough that it encompasses other themes, such as virtualization. In this document it is also defined the work plan and the methodologies to be followed in the development of the project, which will happen in the next semester.

Technologies which are related to Cloud and Grid computing are also described in this document, as they are used in some parts of the project. These technologies are used in the Cloud creation and management, as well as in the web portal to be developed and which will serve as an interface for the whole project.

In this first chapter the context of the project is described, as well as the motivation to work on this project and its objectives, which will serve as guidelines for the dissertation.

1.1 Context

Leonard Kleinrock (part of the team that developed ARPANET - Advanced Research Projects Agency Network, an early seed for the Internet) said in 1969:

> "As [...] computer networks [...] grow and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices around the country." [BYV+09]

Confirming Kleinrock's prediction, computing is migrating in a direction where people develop software for an incredible amount of people in order for them to use it as a service, instead

of running them on their personal computers. Different providers such as Amazon, Google, IBM and Sun Microsystems are now establishing data centers dedicated to hosting Cloud Computing[1] applications spread around the world in order to ensure redundancy and reliability in case one of the datacenters fails.

User requirements for Cloud services are complex and varied, so service providers need to know they can be flexible when delivering those services at the same time they keep the users clear from the infrastructure on which those services stand.

Computing services are available instantly when anyone needs them and the consumers only required to pay the providers when they actually access and use those resources. Consumers no longer have the need to invest in and maintain complex IT infrastructures and software developers are facing new challenges. They must create custom made software that will be used as a service, instead of the traditional practice of installing the software in the users' machines. Some people state this is the era of pervasive computing, where computation and information are available all the time. [McF]

Having this in mind, FEUP is currently developing a private Cloud project at CICA. Its main goals are to make FEUP's computing system more easily accessible and to appeal for a greater usage by FEUP's academic community.

The project aims at automatically creating custom made computing environments so that users can run their computing jobs in the desired hardware, by using dynamically created virtual clusters.

1.2 Project

In order to fully understand what will be developed throughout the next semester, it is needed to comprehend the full scope of the project. FEUP's private Cloud project is composed by two smaller projects which will be interlinked in the future.

Currently, FEUP's computing infrastructures are only accessed by people who have the technical capabilities to interact with the system. These people are the technicians whose area of expertise encompasses outsourcing computing resources to perform computing jobs. If someone from an area that is not Informatics wants to use the system as it is, they need to talk to any of the technicians and waste valuable time (for both parties) cutting through red tape.

As such, this project aims to simplify this whole process and to make FEUP's computing infrastructures more accessible to the academic community, without the users having to spend time studying about the technologies and how the system actually works.

Figure 1.1 describes the "big picture", the whole computing project being presently worked on at FEUP. As it is possible to see, the system should work in the following manner:

1. A Chemical Engineering investigator needs to run a job that requires a great deal of computing effort, more than his/her computer can handle;

[1] Using multiple server computers via a digital network as if they were a single computer.

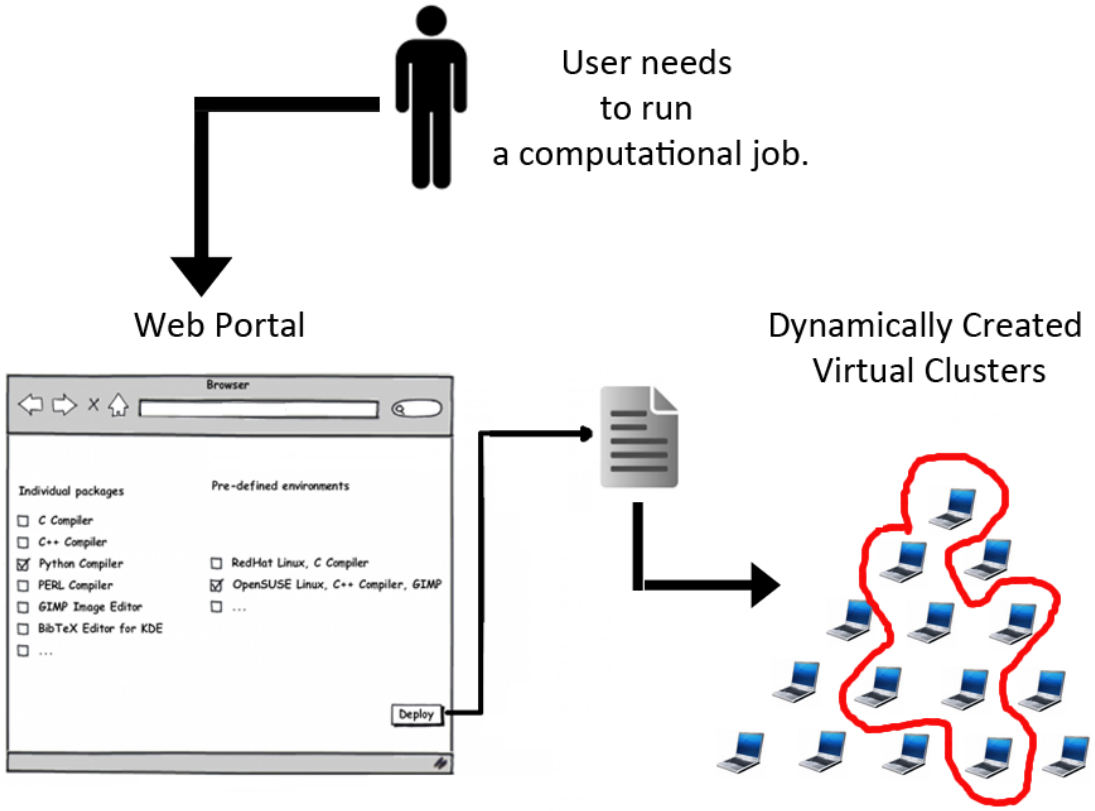

Figure 1.1: FEUP's Private Cloud Project.

2. The investigator accesses the portal to be developed and is presented with a layout similar to the one that is shown in Figure 1.2 where the investigator will be able to either:

   - Choose which Linux distribution and which individual packages should be used, as seen on the left side of Figure 1.2;
   - Choose a pre-configured environment composed of a pre-selected distribution and already installed packages, as seen on the right side of Figure 1.2;

3. After choosing either one of the options, the investigator would then press the "Deploy" button, which would automatically generate an ISO image containing the information that was passed in the previous step. This ISO image would then be used by the back-end of the project;

4. On the back-end, the ISO image previously deployed would then be used by OpenNebula [2] to dynamically create the virtual cluster where the Chemical Engineering investigator would run his computing job;

[2] OpenNebula - Virtual Infrastructure Manager that handles the storage, network and virtualization techniques which enable the dynamic placement of interconnected virtual machines on distributed infrastructures.[Ope]

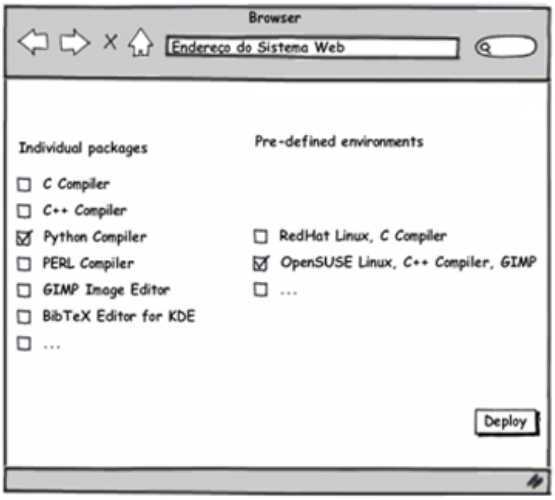

Figure 1.2: Web Portal Interface Mockup.

5. A `<username:password>` combination would then be returned to the investigator so that he could log into the system and run his computing job in the environment created as he desired.

Through this report, what will be referred as the "back-end of the project" consists on the second part of the project (after the ISO image is created), which is currently under development by Nuno Cardoso, a MIEIC - Mestrado Integrado em Engenharia Informática e Computação - student at FEUP, as his Master thesis theme.

1.3 Motivation and Objectives

Some computing infra-structures, namely Grids[3] and Clusters[4], can be rather inflexible when compared to Clouds, as the latter are supposed to allow the user to take advantage of a myriad of services, and not just computing power. [Sun]

However, as powerful as these infra-structures can be, they can be deemed useless if people who need to work with them, cannot do it because they have no knowledge of the technologies (one example could be the one used in the previous section, where a Chemical Engineering investigator needed to use the computing system but simply did not have the technical expertise required to interact with the system). As such, this type of issue causes a lack of growth in the use of FEUP's computing system.

This project aims at increasing the usability of the current computing system that exists at FEUP and with this, increase its usage and stop the lack of growth. In order to achieve this goal, a web portal will be developed that will simplify the access to the system. This portal will have

[3]Distributed systems that are loosely coupled, heterogeneous and geographically dispersed and act together to perform very large tasks.

[4]Group of linked computers working closely together as if they were a single machine.

a list of software packets and Linux distributions that the user can choose from and create an ISO image which will run the investigator's computing job.

A local software repository will be created and managed, so that the system can still run if there is a network failure and software packets cannot be retrieved from a remote repository. The ISO images will be created and stored in a local server in order to provide reusability, as one ISO image may be used in more than one work session and it would be unnecessary to create the same image again.

A requirements elicitation process will also be held in order to determine what is the best course of action to take when defining and implementing the user interface.

1.4 Dissertation Structure

Besides the introduction, this document contains three other chapters. In Chapter 2, the current state of the art is described and related work is discussed. In Chapter 3, the work plan is presented, along with the work methodologies to follow and the task scheduling to be followed next semester during the project development. Finally, in Chapter 4, a few conclusions regarding the research done are presented.

Chapter 2

State of the Art

In this chapter some of the concepts associated with Cloud and Grid computing are presented, as well as some technologies that are related to these fields. Firstly, the concept of computing is approached and it is followed by the concepts of Cloud and Grid computing. A comparison is made between these last two themes, as well as Utility computing, which is related to Cloud computing. The subject of Virtualization and Virtual Machines (VMs) is also analyzed along with some technologies that are used in the project to implement them. Web development tools are also considered in this section.

2.1 Virtualization and Virtual Machines

Throughout this project, Virtual Machines (VMs) are mentioned in great amount and as such, they deserve a special section.

Certain problems arise when the requirements of different virtual organizations (VOs) that need to use the same resources are in conflict or are incompatible with site policies. The software available on clusters cannot guarantee isolation of different communities and maintain resource availability while ensuring good utilization of those said resources. This is where Virtual Machines (VMs) come into play. They are emulations of lower layers of computer abstractions on behalf of the higher layers and allow the isolation of the applications from the hardware and neighbour VMs and customizing the platform so it suits the user's needs. [FFK+06, ZKFF05]

Virtualization benefits include an improvement in fault isolation and independence from guest VMs, performance isolation and simplifying the migration of VMs across different physical machines. These benefits enable VMs to share pools of platform and data center resources. [NS07]

The ability to serialize and migrate the state of a VM paves the way for better load balancing and improved reliability that cannot be achieved with traditional resources. Deploying virtual clusters - set of VMs configured to behave as a cluster and intended to be scheduled on a physical resource at the same time [ZKFF05] - of diverse topologies requires the ability to deploy many

VMs in a coordinated manner so that sharing of infrastructure, such as disks and networking, can be properly configured. This can become more costly than the deployment of single VMs.

In order to understand more, it is necessary to define virtual workspaces (VWs). These are an aggregation of an execution environment and the resources allocated to that specific environment. It is described by the workspace metadata, containing all the information needed for deployment. An atomic workspace, representing a single execution environment, specifies the data that must be obtained and the deployment information that must be configured on deployment. It is also needed to specify a requested resource allocation, something that describes how much of each resource should be allocated to the workspace.

These atomic workspaces can then be combined to formed what is called a virtual cluster. Foster et al propose an aggregate workspace that contains one or more workspace sets - atomic workspaces with the same configuration. Cluster descriptions can be defined in ways that atomic workspaces can be constructed flexibly into more complex structures, organizing at the same time the infrastructure sharing between the virtual nodes. This deployment enables the user to specify different resource allocations for different members of aggregates defined like this. Foster et al consider that the trade-off they obtained is acceptable, as the slowdown suffered was of about 5% and considering that virtual machines offer unprecedented flexibility in terms of matching clients to resources. [FFK+06]

Zhange et al also approached the virtual cluster theme in an article written with both Foster and Freeman, where they combine virtual clusters with Grid technology. The authors only considered two types of node within the cluster: head-nodes and worker nodes. They optimized the loading of the virtual images through image cloning (only transferring one image for all the worker nodes and one image for the head-node, therefore cloning all the worker node images at either staging or deployment time) and they considered that the cost of virtual cluster deployment and management is a good justification for expecting that they may be used for VOs for large groups of short jobs and single long-running jobs. They also found that the cost of running batch jobs in a a virtual cluster was very acceptable.[ZKFF05]

Katarzyna Keahey and Tim Freeman introduce the term *contextualization* in order to describe the process of quickly deploying fully configured images and adapt them to their deployment context, for single VMs. The authors understand *contextualization* as the process of adapting an appliance to its deployment context (an appliance defining an environment as an abstraction independent of its deployment). They are deployed dynamically and are potentially associated with a different context. According to the authors, they can also fulfill three different roles:

1. Appliance providers - they configure environments, maintain them and guarantee their consistency;

2. Resource providers - they provide resources with limited configuration requirements that are designed to support appliances but no longer to provide end-user environments for multiple communities;

3. Appliance deployers - they coordinate the mapping of appliances onto available resource platforms and information exchange between groups of appliances to enable them to share information.[KF08]

There are different types of approaches, since providers can have the applications running inside VMs or provide access to the VMs as a service (Amazon Elastic Compute Cloud), enabling the users to install their own applications. With virtualization, companies are trying to save power by getting the most of what they consume. Running several operating systems inside one machine, they can run independently and CPU idle time is kept to a minimum.[Wei07]

Virtualized computing clusters offer the advantage of being able to transform themselves to the user's needs. However, as pointed out by Nishimura et al, previous work has shown that the system does not scale when increasing the number of VMs and their detailed configuration is not allowed. To counter this issue, Nishimura et al propose a new way of managing virtual clusters so that a flexible and fully-customizable system integration by creating VMs on-the-fly is achieved.

The authors also propose the creation of *virtual disk caches* (VD caches), in order to reduce software installation time. This VD cache is created when a user requests it and is automatically destroyed to keep the total cache size within the given space. What the authors did was that when an installation request is made by the user, the system selects physical resources to host a virtual cluster for the request, instantiates a set of VMs and installs the operating system and other requested software to them. The experiments conducted by the authors using a prototype implementation showed that installing a 190-node virtual cluster can be done in 40 seconds, indicating that the installation of a 1000-VM could be done in under two minutes.[NMM07]

Nathuji and Schwan addressed the issue of integrating power management mechanisms and policies with the virtualization techniques deployed in virtual environments. They propose the *VirtualPower* approach which aims to control and synchronize the effects of the power management policies applied by the VMs to the virtualized resources.

The authors propose this approach having in mind the current limitations in battery capacities and the power delivery and cooling limitations existent in data centers when they try to handle the constant demands of performance and scalability. The authors state that their approach can exploit the hardware power scaling and the methods that control the power consumption of the underlying platforms. It takes guest VMs' power management policies and coordinates them through the system in order to achieve the objectives.

The power management actions are encoded as a set of rules, these being based on a set of mechanisms which serve as a base to implement the power management methods. This approach aimed to present guest VMs with a set of power states and then use the state changes requested by the VMs as inputs to virtualization-level management policies, including those to use specific platforms and their power management capabilities, along with policies that take into consideration goals derived from the applications running through the whole system and from global constraints, such as rack-level limitations on maximum power consumption.

Their findings showed that it is possible to respond to specific power management goals and policies implemented in guest VMs without a need for application specificity to be established at the virtualization level. [NS07]

It is extremely important to discuss *OpenNebula*, as this project will interact with it and as such, it is approached in a later section. 2.6.1

2.1.1 Hypervisors

An hypervisor is a piece of software that emulates the functioning of certain hardware, a process called *Hardware Virtualization*. KVM - Kernel-based Virtual Machine - an open-source virtualization software[1] is used on the back-end of the project..

The following features for KVM were identified:

- Virtualization using hardware virtualization extensions, such as Inter-VT and AMD-V, thus enabling faster virtualization;
- Symmetric Multi Processor emulation - enables multiprocessor hardware emulation;
- Live migration of VMs between hosts, allowing VM relocation without downtime;
- Paravirtualized networking and block devices, which enables faster emulation of those devices. [Car11]

2.2 Grid Computing

Buyya et al believe that Grid computing facilitates the sharing, selection and aggregation of geographically dispersed resources, be it supercomputers, storage systems, data sources or even special assets owned by organizations for solving large-scale resource-intensive problems in different areas of expertise, and that was Grid computing's motivation. Buyya also created a definition for "Grid" at the 2002 Grid Planet conference held in San Jose, United States:

> "A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed 'autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements."[BYV+09]

Ian Foster, one of the most revisited authors regarding Grid computing, states that the "Grid" must be looked upon in respect of the applications it contains, the business value it generates and the scientific results it is capable of returning, instead of its architecture. Carl Kesselman and Ian Foster wrote the following definition in their book "The Grid: Blueprint for a New Computing Infrastructure":

[1] Nuno Cardoso compared XEN and KVM, but came to the conclusion that neither offered an advantage over the other, so KVM was chosen due to its simplistic installation process.

> "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities."[KF98]

Foster and Steve Tuecke redefined the definition, this time referring social and policy issues, affirming that Grid computing is related to resource sharing and problem solving in a coordinated manner and that these occur in dynamic, multi-institutional virtual organizations, the aspect to remember being the power to do something with the result. The authors also stated that they are preoccupied with the "direct access to computers, software, data and other resources."

As such, Foster proposes (as pointed out by the title of his article) a three point checklist that defines what a Grid system should be:

- The Grid should coordinate resources that are not subject to centralized control – Integration and coordination of both users and resources that live within different domains;
- The Grid should use standard, open, general-purpose protocols and interfaces, as this will allow the establishment of dynamic resource-sharing arrangements and the creation of something more than an agglomerate of incompatible and non-interoperable distributed systems;
- The Grid should deliver nontrivial qualities of service, such as response time, throughput, availability, security, co-allocation of multiple resource types to meet complex user demands, resulting in the utility of the combined system to be greater than just the sum of its parts.

Foster also states that the Web is not a Grid, as though its general-purpose protocols support the access to distributed resources; they do not coordinate their use to deliver qualities of service.

Some large-scale Grid deployments inside the scientific community abide by the three points described by Foster, such as NASA's Information Power Grid and the TeraGrid, which will link major U.S. academic sites, as they integrate resources from several institutions, use open and general-purpose protocols (Globus Toolkit, which will be discussed in further details later on this report) to negotiate and manage sharing and they address multiple dimensions of the quality of service, such as security, reliability and performance.[Fos02]

Stockinger started a survey where he contacted over 170 Grid researchers globally spread in order to obtain a general feel on how the Grid was being defined. The results showed that the Grid infrastructure should provide a set of capabilities, such as:

- Description of available resources, what they are capable of doing and how they are connected;
- Visibility into the state of resources, including notifications and logging of significant events and state transitions;
- Assurance of the quality of service across an entire set of resources for the lifetime of their use by an application;

- Provision, life-cycle management and decommissioning of allocated resources;
- Accounting and auditing of the service;
- Security.

The results also showed that a Grid should have a set of characteristics, including:

- Collaboration - sharing resources in a distributed manner;
- Aggregation - the Grid is more than just the sum of all parts;
- Virtualization - Services are provided in a way that the complexity of the infrastructures is hidden from the end-user through the creation of an abstract "layer" between clients and resources;
- Heterogeneity;
- Decentralized control, Standardization and Interoperability - supporting Ian Foster's definition;
- Access transparency - users should be able to access the infrastructure without having to preoccupy themselves how they are doing it;
- Scalability;
- Reconfigurability;
- Security - specially since the systems are often spread through multiple administrative domains. [Sto07]

The members of the EGEE (Enabling Grids for E-sciencE Project) also state that their Grid abides by some of the characteristics mentioned above, namely "decentralized control", "heterogeneity" and "collaboration" [BÓ8]. Their Grid is described in greater detail in the "Grids VS Clouds" section below.

Bote-Lorenzo et al also identified some core Grid characteristics that coincide with Stockinger and Ian Foster's definitions. These include scalability, heterogeneity, resource coordination and dependable, consistent and pervasive access. The propose the following definition for a Grid:

> "... large scale geographically distributed hardware and software infra-structure composed of heterogeneous networked resources owned and shared by multiple administrative organizations which are coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. these applications can perform either distributed computing, hight throughput computing, on-demand computing, data-intensive computing, [...]"[BLDGS04]

Baker et al say that the Grid has evolved from something static and carefully configured, to what has been witnessed in the past years, where it became a seamless and dynamic virtual environment, capturing the attention from the industry and thus making an impact on the Grid's architecture and protocols and standards.

The authors also describe a few standards and organizations that have been actively present in the Grid's environment over the past years. These include the Global Grid Forum (GGF), a community-driven set of groups which goal is to develop standards and best practices for wide-area distributed computing. The GGF creates a group of documents that provide some information to the Grid community, dividing its efforts into several categories, including architecture, data and security.

The authors also approach the World Wide Web Consortium (W3C), an international organization created to promote common and interoperable protocols. This organization was responsible for creating the first Web Services specifications in 2003, such as SOAP and the Web Services Description Language (WSDL). According to the authors, the most important Grid standard to appear recently is the Open Grid Services Architecture (OGSA), which goal is to define a common, standard, and open architecture for Grid-based applications. It was announced by the GGF at the Global Grid Forum in 2002 and in March 2004 it was declared by the GGF to be the flagship architecture.[BAFB05]

Iosup, Dumitresco and Epema analyzed four Grid implementations and the differences on their workload:

- Firstly, they covered the LHC Computing Grid, which testbed has 25,000 (twenty five thousand) CPUs and 3 PetaBytes of storage. Jobs are managed and routed to resources via a Resource Broker, which tries to conduct the job matchmaking and balance the workloads at the global level. The site used by the authors had around 880 CPUs;

- Secondly, they looked at the Grid3 testbed, representing a multivirtual organization environment that sustains production level services required by various physics experiments. It is composed by more than 30 sites with 4500 (four thousand five hundred) CPUs;

- Thirdly, they analysed the TeraGrid system - used for scientific research - which has over 13,6 TeraFLOPS of computing power and can store 450 TeraBytes of data;

- Finally, they reviewed the DAS-2 environment, which has 400 CPUs spread over five Dutch Universities and its workload ranges from single CPU jobs to very complex ones. These can be submitted either via the local resource managers or to Grid interfaces that communicate with them.

They discovered that while Grid research focuses on complex application types, most of the applications encountered were extremely easy to run in parallel (embarrassingly parallel applications).

The authors identified two large problems, a scale (origin and size of the data that must be collected) and a methodological (missing components of the information) problem. In order to address the first problem, the information should ideally come from three different sources:

- Local and Grid scheduler - without these logs, job arrival and dependency information can be lost and an analysis of site-related performance metrics cannot be done;
- Grid AAA (authentication, authorization and accounting) modules - these modules provide the information regarding the link between jobs and their owners;
- Monitoring systems - without the information these systems provide, it is impossible to understand how the applications are running within the Grid and to quantify the system utilization.

The authors concluded that a small number of VOs and users control the workload in terms of submitted jobs and consumed resources, system evolution can appear at the system, VO and user level and should be considered when provisioning resources.[IDE+06]

Malcolm Atkinson from the National e-Science Center in the United Kingdom, says the following:

> "With Web Services we allow a thousand flowers to bloom. With a Grid we organize the planting and growth of a crop of plants to make harvesting easier."[Sto07]

Iosup et al end their article with the following quote:

> "[...] conclude that Grids are not yet utilized at their full capacity."[IDE+06]

which serves as a conclusion for this section.

2.3 Cloud Computing

Similarly to the Grid, many definitions arise when one talks about the Cloud. Presently, it is considered normal to obtain access to content spread over the Internet without a reference to the hosting infrastructure that lies underneath it. This infrastructure is made of data centers that are being monitored by service providers.

Buyaa et al state that Cloud computing extends this paradigm in where the capabilities of the applications are viewed as complex services that can be accessed over a network. The authors also believe that the Cloud is an infrastructure from where businesses and users can access applications from anywhere in the world anytime they want. Cloud computing's services need to be reliable, scalable and sufficiently autonomic to support omnipresent access, dynamic discovery and they need to support composability, as they must permit to be reassembled and selected in any order to comply to the user's requirements.

The authors have a definition of their own:

> "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." [BYV+09]

They believe that Clouds are the new datacenters with hypervisor technologies such as VMs, with services provided on-demand as a personalized resource collection in order to meet the service-level agreement, which should be established *à priori* with a "negotiation" and accessible as a composable service via Web Service technologies.

Vaquero et al state that the paradigm of Cloud computing shifts the infrastructure to the network in order to reduce the costs that are normally associated with the management of hardware and software resources.

Having in mind Gartner's Hype Cycle[2], Vaquero et al state that Cloud computing is now in its first stage – Positive Hype – mixing every definition that appears into an overly general term that confuses every single person. The same thing that happened to Grids can be applied here. There are no widely accepted definitions (Foster's being the most accepted one) and a clear definition can help transmit what it actually is and how businesses can reap benefits from it.

There are many Cloud definitions, but they all focus on certain technological aspects. Thus, Vaquero et al try to analyze all the features of Cloud computing in order to reach a clearer definition.

The authors try to distinguish the different actors and scenarios that can arise:

The actors:

Service Providers make services accessible to Service Users through Internet-based Interfaces. The computing infrastructure is offered "as a service" by the Infrastructure Providers, moving computing resources from the SPs to the IPs, in order to give the firsts flexibility and reduced costs.

The scenarios:

- Infrastructure as a Service – IPs are responsible for the management of a large set of computing resources, such as storing and processing capacity. If they use virtualization, they can split, assign and dynamically resize the resources to build ad-hoc systems as the customers (SPs) demand, by deploying the software stacks that run their services.

- Platform as a Service – Clouds offer an additional abstraction layer – they can provide the software platform where systems run on. The sizing of the hardware resources demanded

[2] Graphic representation of the maturity, adoption and social application of specific technologies. It has five phases: 1 Product launch that generates interest; 2 - Frenzy of publicity generates over-enthusiasm and unrealistic expectations; 3 - Technologies fail to meet expectations and become unfashionable; 4 - Press stopped to cover the technologies, but some businesses continue to experiment and understand its benefits; 5 - Benefits become widely demonstrated and accepted. Technology becomes stable and evolves.

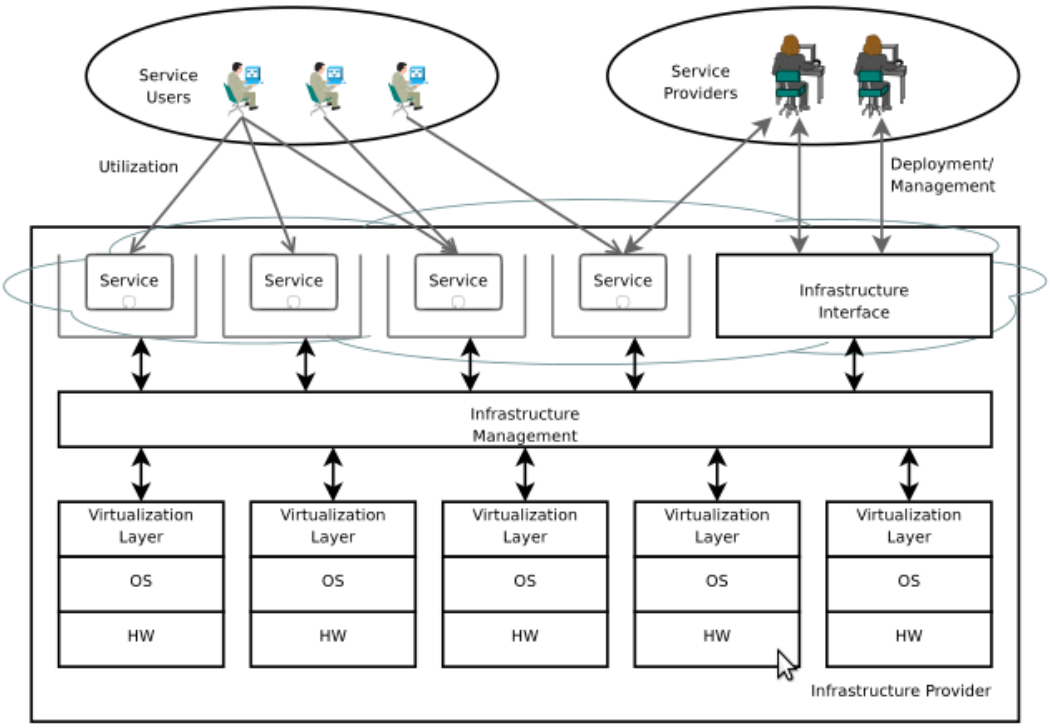

Figure 2.1: Cloud Actors [BYV+09].

by the execution of the services is made in a transparent manner. The applications developed are run on the provider's infrastructure and are delivered through the Internet from the provider's servers. The Google Apps Engine is a very good example.

- Software as a Service – Cloud systems can host many services that users can be interested in, such as online word processors or even Google Apps. [KG09, VRMCL08]

In the article written by Vaquero et al, many Cloud definitions are gathered. Markus Klems states that the key elements for the Cloud are immediate scalability and the optimization of resources usage, these being bring provided by increased monitoring and automation of resources management. Jeff Kaplan and Reuven Cohen prefer to focus on the business model, paying more attention to the collaboration and pay-as-you-go and reducing the costs of investment. Douglas Gourlay and Kirill Sheynkman define the Cloud as being simple virtualized hardware and software, combined with monitoring and provisioning technologies. [GKC+, VRMCL08]

McFedries believes that the basic unit of the Cloud is nothing other than data centers - huge collection of clusters - that can offer a large supply of computing power and storage simply by using whatever resources they can spare.[McF]

Kevin Hartig defines Cloud Computing as being able to access resources and services needed to perform certain tasks with needs that are constantly changing. The application or user requests

access from the cloud rather than on a specific endpoint of the network or a resource. The Cloud becomes a virtualization of resources that is both self maintainable and manageable, a view also shared by Jan Pritzker, who focuses his definition on virtualization and on-demand resource allocation. Other authors such as Reuven Cohen, Praising Gaw, Damon Edwards and Ben Kepes (to name a few) are strong believers that Cloud computing is nothing more other than a buzz word, grouping concepts such as deployment, load balancing, provisioning and data and processing outsourcing.[GKC+]

Having this in mind, Vaquero et al believe that the Cloud is a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services), these being dynamically reconfigured to adjust to a variable load (scaling) also allowing for an optimum resource utilization. The pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service-Level Agreements. The authors also state the set of features that resemble this minimum definition would be scalability, pay-per-use utility model and virtualization. [VRMCL08]

Brian Hayes states in his article that even though the future of Cloud computing is still unclear, there are a few directions in which it can go. One of those directions is Web based services, such as Google Docs or even Photoshop Express. Salesforce.com also offers a variety of online applications and its slogan is actually "No software!".

As mentioned earlier in the report, Amazon.com also ventured into this new paradigm, offering data storage and computing capacity, each of these services being able to expand and contract as the users need (elasticity) and Google has its App Engine, providing hosting on Google server farms.

There is great concern in terms of scalability in the Cloud, as it might be necessary to organize resources so that the program runs flawlessly even though the number of concurrent users increases might arise. Hayes also mentions that Cloud computing raises questions in terms of privacy, security and reliability, since personal documents are being delivered to a third-party service.[Hay08]

Nicholas Carr writes in his book that a shift is happening, where the Cloud is becoming similar (if not equal) to the electric grid, as we can connect to the Cloud and get data, storage space and processing power cheaply and instantly (Utility Computing). [Car09]

Aaron Weiss writes in his article that the Cloud is robust, even self-healing, as it has many sources from where to get the power to recover from whatever accident occurs. Weiss states that the Cloud is also very power consuming, as roughly 50 percent of the energy it consumes comes from the cooling process alone. Giants such as IBM and Microsoft are also scouting locations where the hydroelectric power is cheaper and greener, so they can establish their cluster centers.[Wei07]

2.3.1 Utility Computing

Computing is going in such a direction that the services that are made available to the user are being done in such a way that computing is becoming equal to traditional utilities such as water, gas, electricity and telephone services.[BYV+09]

In an article published by InfoWorld (formerly the Intelligent Machines Journal), Utility computing is mentioned as being a form of Cloud computing, where storage and virtual servers are being offered and can be accessed on demand, such as the services offered by Amazon.com, Sun or IBM. [BÓ8]

IBM Global Services provide the following definition for Utility Computing:

> "Utility computing is the on demand delivery of infrastructure, applications, and business processes in a security-rich, shared, scalable, and standards-based computer environment over the Internet for a fee. Customers will tap into IT[3] resources - and pay for them - as easily as they now get their electricity and water." [Rap04]

Utilities have the following characteristics:

- Necessity - Users depend on utility services to fulfill their day-to-day needs. It takes time for distribution networks to spread and costs to decline, as it also takes time for users to adapt to the service. Once they do, the service may grow in importance as users begin to find new ways to reap benefits from it;
- Reliability - The service must be readily available when and where the user requires it, as a temporal or intermittent loss of service may cause several issues to the user. Redundancy must be built into production capacity in order to make up for hypothetical service failures;
- Usability - Users have a "plug-and-play" mentality and they need to feel at ease with whatever feature they are using;
- Utilization rates - Utilities are driven by a necessity to carefully manage utilization rates. User demands for utility serves mays vary over time and across service regions. This may lead to spikes in utilization of the service and under-utilization in off-peak periods. Service providers must have in mind that how the service is billed may influence how users use that service;
- Scalability - As production capacity grows, the unit cost of production shrinks. It might be expected that as the demand for the service rises, the quality of service may decline or vice-versa [Rap04].

Bhattacharya and Vashistha state that utility based computing allows computing resources to be available for a customer on demand, as the customers subscribe to the services of the utility provider and only pay for the quantum of the resources used. This allows any customer to cut down on IT infrastructure spendings as they can simply subscribe to the provider's services and use the computing resources at will, only paying for as much as they use. Typical measures of usage include metered CPU hours and memory space usage [BV08].

Ross and Westerman write in their article that utility computing relies on several important technical capabilities to deliver what it promises - services available on-demand. The authors

[3] Information Technologies

believe that for most firms, the impact of utility computing will be on the extent and nature of outsourcing. The benefits that can be obtained only enhance the current benefits of IT and business processes outsourcing: lower cost, variable capacity and increased strategic focus. On demand capacity leads to firms to invest less in computing capacity. Advances in autonomic computing may reduce the number of people needed to monitor operations and thus reduce labor costs.

The authors believe that firms will be able to do more with less and will be able to allocate their most strategic resources to their most strategic opportunities [RW04].

2.4 Grids VS. Clouds

As one knows, Grids and Clouds share a few goals, such as reducing computing costs and increasing flexibility and reliability through the use of third-party operated hardware.

Vaquero et al lay out a very comprehensive list of features and discuss the similarities and differences between them. The list includes resource sharing, heterogeneity, virtualization, security, the offer of high level services such as metadata search, the awareness of architecture, dependencies and platform, software workflow, scalability and self management, standardization, payment model and quality of service. The list is shown in Figure A.1 which is in Appendix A.

The authors also believe that Grids are meant to be user friendly, virtualized and automatically scalable utilities, something that steps into the Clouds' path, but they still need to be able to incorporate virtualization techniques in order to obtain some advantages already present in the use of Clouds, like migrability and hardware level scalability.[VRMCL08]

A few members of the Enabling Grids for E-sciencE (EGEE, now part of the European Grid Infrastructure) performed a comparative analysis on Grids and Clouds, focusing two implementations of both: the EGEE project for Grid and the *Amazon Web Service* (AWS) for Cloud, using metrics such as performance, scale, ease of use, costs and functionality, amongst others. The Grid in use by the EGEE runs on gLite, an open source software which had development funding from the EGEE, described in a later section of this document, as it is used in some extent by FEUP's cluster system.

When comparing both EGEE Grid and the *Amazon Web Service*, the authors of the analysis encounter a set of differences and similarities:

- The AWS does not expose how they operate their data centers and how they implement the user interfaces, execute the user requests and maintain their accounting, its back-end is still a grey area;
- The EGEE Grid exposes both user interface as well as the resource interface to permit providers to connect their resources. The AWS hides this second interface;
- The authors assume that on the resource side, both systems work in similar manner, as both cases require a queueing mechanism whether the data center is dispatching a grid job via a batch system or is requested to instantiate a new virtual machine;

- The greatest benefit of the Cloud proposed by Amazon is its interfaces and usage patterns, focused on simplicity;
- Both services are not fail-proof, but the authors consider that a centralized Cloud might not be able to provide the resilience that the distributed nature of EGEE does;
- Grids are typically used for job execution - limited duration execution of a program, part of a larger set of jobs, consuming or producing a significant amount of data. Clouds, even though they support a job usage pattern, they seem to be more often used for long-serving services;
- Amazon bills users for computing resources usage with a minimum of one hour usage. This stops being efficient when dealing with a large number of small jobs;
- Elasticity in the Grid is made by adding worker nodes at a site or adding new sites;
- The complexity in the Cloud is kept server-side, which makes its entry point very low, something that is still considered a goal to achieve for Grids. [BÓ8]

2.5 Developments, Applications and Services

With the shift of the computing industry towards a provision of Platform as Service and Software as a Service, consumers can access resources on-demand without having to preoccupy themselves with time and location, Buyya et al believe that there will be an increasing number of Cloud platforms being developed. [BYV+09] One of those platforms is *OpenNebula*, an open-source tool kit for Cloud management. In this report, the Amazon computing service (2.5.1) will also be analysed, as well as *Google's App Engine* (2.5.2) and *Microsoft Azure* (2.5.3). *Rackspace Hosting* and *NASA*'s *OpenStack*

2.5.1 AWS - Amazon Web Services

The *Amazon Web Services* consist of several components, but only two will be taken into consideration in this document, as they are the most relevant to the work discussed: *Amazon's Simple Storage System* (2.5.1.1) and *Amazon's Elastic Computing Cloud* (2.5.1.2).

2.5.1.1 Amazon's Simple Storage Service

The core service for the *Amazon Web Services* is the *Amazon's Simple Storage Service*, that gives the user the power to store large amounts of data in a reliable way which does not hinder its availability. Data is accessed through protocols such as SOAP[4] and REST[5], while also being able

[4] Simple Object Access Protocol - Used to exchange information in the implementation of Web Services in computer networks.

[5] Representational State Transfer - Style of software architecture for distributed hypermedia such as the World Wide Web.

to be accessible via normal web browsers. The storage model runs on a two-level hierarchy, where the users can create *buckets* and place data *objects* in those buckets. Strings are used as keys for both buckets and objects, thus being able to be easily incorporated in URLs. Users are charged 15 US cents per Gigabyte per month, each user being able to have up to 100 buckets and each can hold up to 5GB of data.[Haz08]

2.5.1.2 Amazon's Elastic Computing Cloud

Physically speaking, the *Elastic Computing Cloud* (EC2) is a large number of computers on which Amazon provides time to paying customers, these computers being spread all over the United States. EC2 is based on the XEN virtualization technology, which allows one physical computer to be shared by several virtual ones, each with its own operating system.

Through the use of virtualization, the users create an image of their software environment using the tools provided. This will be used to create and instance of a machine in Amazon's Cloud. Customers can freely choose configuration templates for their instance and they can create and destroy the instances at will, enabling the software to scale itself to the amount of computing power it needs.[BÓ8, Haz08]

Amazon has released *Elastic IPs* (Static IPs for Dynamic Cloud Computing), which allows the assignment of static IPs to dynamic resources that are deployed via EC2, as well a service that enables users to request EC2 instances to be geographically distributed, as a response to the demand for EC2 IP addresses in a static range for application range for applications like email service hosting, as well as providing a safety net in case the operations of an Amazon Web Services data center go awry.

Amazon provides a variety of ways of requesting the EC2 instances, namely through the use of Web Services, supporting Buyya et al's Cloud definition previously mentioned in the document.

Amazon has also introduced its own performance unit named "EC2 Compute Unit". Since Amazon ventured into the Utility computing field, model it follows differs from the traditional way developers were formatted to think about CPU resources. Instead of renting a certain processor for several months or years, it is now rented by the hour. One EC2 Compute Unit provides the CPU equivalent of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.[Ama]

2.5.2 Google Cloud - Google's App Engine

The *Google Cloud*'s official name is *App Engine* or *Appengine*. It gives developers the ability to run web applications on Google's infrastructure, the same that is being used by *Google* for *GMail* and *Google Docs*. The Cloud appears to be a platform accessible over the Internet with limitless hardware, the latest software and abundant storage for deploying web applications. The *App Engine* has the following features:

- Automatic horizontal scaling and load balancing;

- APIs[6] for authenticating users with Google Accounts and for sending emails. No system administration is needed by the user to set up or allow access to these APIs;
- Fully featured Eclipse developed environment that simulates *Google App Engine* on the localhost for development and testing;
- Persistent storage and support for transactions and queries using the standard JDO[7] and JPA[8] APIs;
- Generous free quotas, which allow small universities to have access to the same hardware and software as large industries. Each user can have 10 applications created, each with 10 versions, which totals an effective development environment of 100 applications. A free account supports six and a half CPU hours a day, with 1GB of stored data and sending email to 2000 recipients a day and a max of 5 million page views a month;
- It is free, with no contracts to sign, no hardware expense and no system administration costs for maintaining, updating, patching or backing up *App Engine*;
- Eclipse plug-in available for Apple, Linux and Windows, which allows standard debugging using Eclipse debug tools. It provides menu based functionality to automatically upload the application to the Google App Engine;
- Requires no system administration;
- Simple web based, user friendly console.[HP10]

2.5.3 Microsoft Azure

Microsoft Azure platform is a cloud computing platform which offers a set of cloud computing services similar to those offered by Amazon Web Services. *Windows Azure Compute* (Microsoft's counterpart to Amazon's EC2), only supports Windows virtual machines and offers a limited variety of instance types when compared with Amazon's EC2. Its instance type configurations and cost scales up linearly from small to extra large and its instances are available in 64 bit x86_64 environments.

It has been speculated that the clock speed of a single CPU core in *Azure*'s terminology is approximately 1.5 GHz to 1.7 GHz.[GWQF10] Windows Azure enables developers to build, host and scale applications in Microsoft datacenters, not requiring upfront expenses, long term commitment and users only pay for the resources they use. Windows Azure relieves the user from the effort of configuring load balancing and failover, is designed to let developers build applications that are continuously available, even if they need software updates and hardware failures occur.[Mic]

[6] Application Programming Interface
[7] Java Data Objects
[8] Java Persistence API

2.6 FEUP's Computing System

In this section FEUP's computing system is analysed in detail. The cluster system and the technologies it uses in its management are described and detailed. FEUP's cluster system currently uses three different technologies, *Moab*, *gLite* and *Condor*. The Cloud creation and management tool is also described, as it is a vital part in FEUP's computing system and is the connecting link between the front-end and the back-end.

2.6.1 OpenNebula

OpenNebula is an academic project developed by the *Universidad Complutense Madrid*. It was created in order to control several hosts and machines in a unified environment. It has been released as a software package in *Ubuntu* 9.04 and is compatible with both Xen and KVM, two *platform virtual machines* that emulate the whole physical computer machine. It has its own command-line tools and gives the user different configuration scripts which enable a simple and flexible way to design and manage running virtual machines. OpenNebula's architecture is shown in 2.2 [Pin10].

Figure 2.2: OpenNebula's Architecture [CGH09].

OpenNebula does not have a built-in utility to create VMs from scratch, but its templates allow the VMs to boot an ISO image, leaving the user with just creating an empty hard disk image.

It provides monitoring capabilities which become rather useful when there is a need to troubleshoot, scale or control resource allocation scenarios. *OpenNebula* exports drivers that communicate directly with the hypervisor (KVM - 2.1.1) and return useful data, such as the amount of CPU used, reserved and used memory and network traffic.[CGH09]

Figure 2.3 sums up the pros and cons of using *OpenNebula*.

Pros	Cons
Highly configurable VM templates	Limited Network Handling
Scriptable through the drivers	Limited to 1 cluster
Cloud Bursting (EC2 management)	No default VM connection enabled
Finely tunable configuration scripts	No GUI

Figure 2.3: Pros and Cons of using *OpenNebula* [CGH09].

2.6.2 OpenStack

OpenStack is a global collaboration of developers and cloud computing technologists producing the open source cloud computing platform for public and private clouds. The aim of this project is to deliver solutions for all types of clouds by being simple to implement, massively scalable and filled with features.

It was founded by *Rackspace Hosting* and NASA[9] and it has grown to be a global software community of developers collaborating on a standard open source cloud operating system.

OpenStack's mission is to enable any organization to create and offer cloud computing services running on standard hardware.

All of its code is available under the *Apache* 2.0 license and as such, anyone can run it, build applications on it or submit changes back to the project.

OpenStack has three major components:

- **Compute** - Designed to provision and manage large networks of virtual machines, creating a redundant and scalable Cloud computing platform. It has built-in control panels, APIs required to create and manage a Cloud, including running instances, managing networks and controlling access through users and projects;

- **Swift** - *OpenStack* Object Storage - Designed to create redundant and scalable object storage using clusters or standardized servers to store PetaBytes of accessible data. It is a long-term storage system for a permanent type of static data that can be retrieved, leveraged and then updated if necessary, such as virtual machine images, photo storage, email storage and backup archiving;

- **Glance** - *OpenStack* Image Service - Provides discovery, registration and delivery services for virtual disk images. It relies on a REST[10] interface for querying information. It currently allows uploads of private and public images which include those created via KVM, VMWare and VirtualBox.

[9] North American Space Agency
[10] Representational State Transfer

It is a project still in its early stages, as it was first released in October 2010 and is only in its third version [CCa].

This project is under close surveillance by CICA as it is viewed as a possible substitute for *OpenNebula* (2.6.1).

2.6.3 Clusters

Three different technologies are currently in use by FEUP's Cluster system: *Moab*(2.6.3.1), *gLite*(2.6.3.2) and *Condor*(2.6.3.3).

2.6.3.1 Moab Cluster Suite

Moab Cluster Suite is a proprietary tool for high performance computing systems, developed by the company *Cluster Resources*. It has built-in modules for work management, Cluster administration and monitoring, report creation. It is composed by three essential components:

- **Moab Workload Manager** - scheduling and workload management engine;
- **Moab Cluster Manager** - graphical interface for Cluster administration, monitoring and report analysis;
- **Moab Access Portal** - web based portal for job management and submission, directly focused on the end-user.

A resource manager supplies the system with basic functionalities for initiating, stopping, canceling or monitoring jobs. *Moab Workload Manager* uses a resource manager's services to get information about the state of the resources and the node workload. It is also used to manage jobs and to send information on how they should be run and it can be configured to manage more than one resource manager simultaneously. Its composing nodes can be split into three groups:

1. **Master node** - Manages the resources;
2. **Submissive/Interactive nodes** - Allow users to manage and submit jobs into the system;
3. **Computing nodes** - Execute the submitted jobs.

Is is also possible to split the nodes into two groups - source and destination nodes. The first ones are the nodes where there users, portals or other systems can submit their jobs and the latter ones are where the jobs are executed. Jobs originate in a source node and are transferred to the destination nodes. Decisions are made in the source nodes, so it is possible to choose which nodes will execute the submitted jobs. Moab also allows for the establishment of connections between several Grid systems, which permit access to additional resources.[Pin10, Res]

2.6.3.2 gLite

gLite consists in a set of components designed with the objective of building a Grid computing infrastructure for resource sharing developed by the project EGEE (Enabling Grids for E-sciencE), also mentioned in an earlier section. *gLite* is based on four core concepts:

- **Computing Element (CE)** - Set of local computational resources, namely a Cluster. It is composed by three components:
  - **Grid Gate** - generic interface for the Cluster who receives jobs and submits it to the Local Resource Management System (LRMS);
  - **Local Resource Management System (LRMS)** - Sends the jobs to the worker nodes for execution;
  - **Worker Nodes** - Cluster nodes where jobs are executed.
- **Storage Element (SE)** - Supplies access to data storage resources;
- **Information Service (IS)** - Resource research is made through this component, which is also responsible for supplying information regarding resources and their state;
- **Workload Management System (WMS)** - Receives jobs from users, appropriately allocates a CE, saves jobs states and gets the final results.[Pin10, CER]

2.6.3.3 Condor

Condor is a free and open-source workload management system, developed by the *Condor Research Project*. It has built-in job queueing mechanisms, scheduling and priority policies, resource monitoring and management. Users submit jobs to *Condor*, which puts them in a queue, chooses when and where to execute them based on defined policies, carefully monitors their progress and informs the user when the jobs are finished. *Condor* can manage dedicated nodes or harness the CPU energy wasted in workstations that are turned on but unused. If the system detects the machine became suddenly unavailable, *Condor* can migrate the state of the job into a different machine and resume work. It offers an extremely flexible structure to assign resources to jobs, allowing these to have specific requisites and resource preferences, as well as enabling the resources to specify preferences over jobs to execute. Each machine from *Condor* can play several roles:

- Central Manager - Machine that collects information and makes the negotiation between resources and resource requests. All resource requests go through the *Central Manager*. There can only be one *Central Manager* in a *Condor* infrastructure;
- Execute - Machine which executes jobs, therefore allowing the network to take advantage of its resources. Any machine can be configured to take this role;
- Submit - Machine responsible for the job reception and submission to the *Central Manager*. Any machine can be configured to take this role;

- Checkpoint Server - Machine which stores checkpoint files for all submitted jobs.[Pin10, Tea]

2.7 Software Packages

Webopedia[11] describes *Software Packages* as:

> "A special method of distributing and installing software (or software upgrades) to a computer." [Web]

The package is provided as compiled code, with its installation and removal handled by a package management system (PMS). Each package contains meta-information such as package description, version and dependencies.

2.7.1 Debian Packages

There are two types of *Debian packages*:

- Binary packages - Contain executables, configuration files, copyright information and other documentation.The packages are distributed in a *Debian*-specific archive format and they are usually distinguished by having a '.deb' file extension. Then can be unpacked by using the *Debian* utility *dpkg*;
- Source packages - A '.dsc' file which describes the source package, a '.orig.tar.gz' file which contains the original unmodified source in *gzip*-compressed tar format and a '.diff.gz' file that contains the *Debian*-specific changes to the original source.

Installing the software via the package system uses "dependencies" which are carefully designed by the package maintainers. They are documented in a special *control* file associated with each package.

Debian's packaging tools can be used to:

- Manipulate and manage packages or parts of packages;
- Administer local overrides of files in a package;
- Aid developers in the construction of package archives;
- Aid users in the installation of packages which reside one a remote FTP site.[Debb]

Working with packages in *Debian* uses three main utilities:

- **apt** - Advanced Package Tool - Main package manager in *Debian* systems used for retrieving/installing, removing or searching for packages;

[11] http://www.webopedia.com

- **dpkg** - package manager for *Debian* - Predecessor of *"apt"*. Tool to install, build, remove and manage *Debian* packages. Controlled entirely via command line parameters.[Rev]

- **dselect** - menu driven front-end that uses both *"apt"* and *"dpkg"*.

2.7.1.1 Repositories

A *Debian* repository is a set of *Debian* packages organized in a special directory tree which contains a few additional files containing indexes and checksums for the packages. If a user adds a repository to the **/etc/apt/sources.list** file, all the packages available can be viewed and installed as if they were the packages contained in *Debian*.[Par]

A complete list of the current "stable" *Debian* packages is located at `http://packages.debian.org/stable/`. Currently, FEUP also provides this list at `http://mirrors.fe.up.pt`.

The *apt*-utilities can retrieve the packages from DVDs or the Internet via *HTTP* or *FTP*, which means that a system can be fully updated via an Internet connection.

There are several guides on how to install and manage a *Debian* repository, so the process will not be described in this document, as the information would be redundant and time-wasting. Some of these guides can be found at [Deba, Con].

2.8 ISO Images

An ISO file is and "image" of an entire CD or DVD. All the contents of a disc can be represented perfectly in a single ISO file. They are often used to distribute large programs over the Internet as all the files can be contained in one single file - the ISO [Sup].

These are the files that will be created when using the web portal discussed earlier in this document.

There are several tools available for creating such files - Symantec's GHOST (2.8.1), Suse Studio (2.8.2) and *Ubuntu*'s VM Builder (2.8.3).

2.8.1 Symantec's GHOST

GHOST[12] is a disk cloning program distributed by Symantec. It is the tool used by CICA to create its custom Linux distributions that are installed in FEUP's computers. [Sym]

It supports a process called "hot imaging", which allows the user to create images from a 'LIVE' operating system and still be able to change the files. It combines the GHOST executable file and the image file it creates (with a '.gho' extension). and gives the user the ability to restore the system to a working configuration in minutes. A tutorial on creating a GHOST image can be found in [Rad].

[12] General Hardware-Oriented System Transfer

2.8.2 Suse Studio

SUSE Studio is a web-based user interface to build software appliances. These are pre-configured combinations of an application and its configuration, and includes an operating system. All these parts are integrated into a single image and can be deployed.

SUSE Studio offers a series of utilities, all of them web-based. It is possible to specify where to get the software from (i.e. choosing the software repositories) and which type of image it is (VMWare, KVM, Live CD, XEN image, etc.) [Nov, Lif].

2.8.3 Ubuntu's VM Builder

VMBuilder is a Python-based software package for creating VM images of free software GNU/Linux-based operating systems. It is under development by the *Ubuntu* Virtualization Team[13] and it simplifies the process of building and installing VMs [Ubub].

VMBuilder currently supports KVM, XEN and VMWare as hypervisors. Command line options can be used to add and remove packages, choose which version of *Ubuntu* to run, which mirror to use, among others [Ubua].

A tutorial for working with VMBuilder can be found at [Debc].

[13] https://launchpad.net/vmbuilder

Chapter 3

Workplan and Methodologies

In this section the workplan and methodologies to be followed during the development of the project over the next semester will be presented and detailed.

3.1 Technologies to be used

Since this project consists on developing a web portal, it is necessary to detail which technologies will be used to build it. There are two major programming languages candidates for this development, *Ruby* on the *Rails* platform and *Python* on the *Django* platform.

3.1.1 Ruby on Rails

> "Ruby on Rails is a breakthrough in lowering the barriers of entry to programming. Powerful web applications that formerly might have taken weeks or months to develop can be produced in a matter of days." -Tim O'Reilly, Founder of O'Reilly Media [HH]

Ruby on Rails is a Web 2.0 framework that attempts to combine PHP's simple immediacy with Java's architecture, purity and quality. It forms an environment and provides all the tools to create business-critical, database-supported web applications. Its basic objectives are simplicity, reusability, expandability, testability, productivity and maintainability.

It implements a MVC (Model-View-Controller) architecture, which clearly separates code according to its purpose.

Ruby code is easy to read and is based on languages such as Python, Perl and Lisp [BK07].

Ruby on Rails official website offers a wide range of APIs, guides and books, which make for an extremely well documented framework [HH].

3.1.2 Python - Django

Django is a high-level Python web framework that encourages rapid develop and clean, pragmatic design. It was designed to handle two challenges: intensive deadlines and the requirements of the experienced web developers who wrote it.

Just like *Ruby on Rails*, *Django* focuses itself on the DRY[1] principle, which states:

> "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." [CCb]

Django's documentation is extremely extensive, and can be found in its official website [2] and in the online version of the *Django Book*, a free book about the *Django Web Framework* [3].

Python and *Django* are the best candidates to be used, as VMBuilder is a suitable choice for the needs of this project, since it can run inside *Ubuntu* and is *Python* based.

Python will also be used in any scripting necessary, unless it cannot be done specifically in *Python*. The back-end scritping is written in *Bash*, something that can be easily integrated into Python modules [sei].

3.2 Tasks to complete

This project will be developed in phases, each of which corresponds to a task that should be completed.

3.2.1 Package Repositories

This task is composed by two sub-tasks - creating a local repository and accessing and managing this said repository. It was chosen to take this path because as it was mentioned earlier, without a local repository and there is a network failure, the whole system is rendered useless because the system cannot access the software packages to create the ISO image.

A connection to a remote server - *Debian*'s official repository - will also be made, to ensure redundancy.

3.2.2 Software Packages

This task is also composed of two sub-tasks - downloading the packages and managing them. This should be done right after the previous task was completed. On a first stage the packages should be listed and downloaded and then managed, as in, they should be ready to be packed with the kernel of a Linux distribution for the next phase (3.2.3).

[1] Don't Repeat Yourself
[2] https://docs.djangoproject.com/en/1.3/
[3] http://www.djangobook.com

3.2.3 Creating custom ISO images

This is the core task of the system. The packages collected from the previous task should be packed with a Linux kernel and an ISO image should be created.

3.2.4 Reviewing the State of the Art

As this is a relatively new subject that is constantly evolving and new developments are being made, is is obligatory to keep up with the new discoveries and projects. As such, some time should be taken periodically in order to do a research for papers and projects that could appear during the development of the project. A suggested periodicity would be on a weekly or fortnight basis.

3.2.5 System testing

As with every technical project, it is necessary to carry out tests on the system, to ensure that everything is being developed in a correct fashion. System tests are critical for identifying bugs on the interface or on the software development process.

3.2.6 Writing the documentation

The dissertation and all the related documentation should be written on time and with time to spare, in order to avoid delays on the deployment of the project.

3.2.7 Developing the web portal

This task should be started once the software repositories are created, and it should follow an iterative prototyping model, in the sense that after new functionalities are implemented, they are tested in terms of usability, followed by the correction of whatever problems are detected and only then it will be possible to jump to the next implementation.

These usability tests are to be carried out with researchers that use FEUP's computing system.

3.3 Calendarization

Calendarization was made regarding the priority of the tasks to perform. Each phase includes a requirements elicitation phase, an implementation phase, a usability tests phase and a problem detection and correction phase.

The timeline presented in Figure 3.1 represents the calendar to follow task-wise.

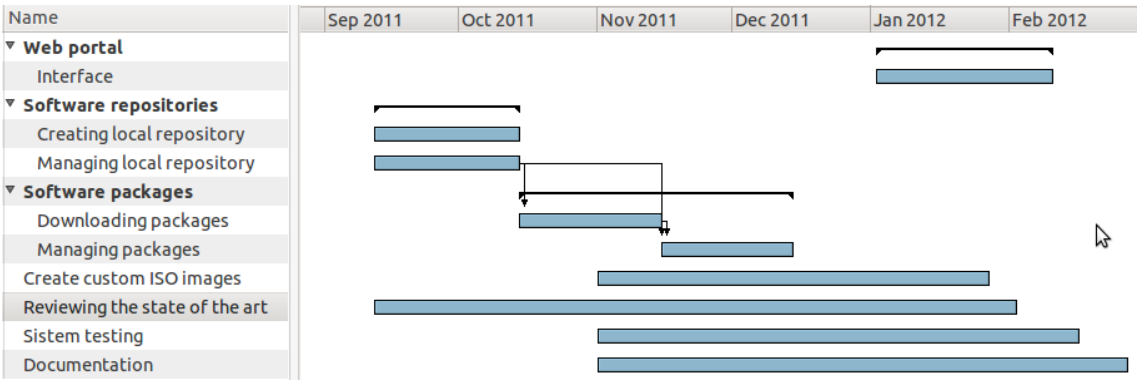

Figure 3.1: Tasks distribution for the next semester.

Estimated duration of the tasks:

- Developing the web portal 3.2.7 - 7 weeks;
- Software repositories 3.2.1 - 4 weeks:
  - Creating a local repository;
  - Accessing and managing the local repository.
- Software Packages 3.2.2 - 10 weeks:
  - Downloading packages;
  - Managing Packages.
- Creating custom ISO images 3.2.3 - 12 weeks;
- Reviewing the State of the Art 3.2.4 - Periodical review which should be made every week or every fortnight as mentioned earlier.;
- System testing 3.2.5 - 22 weeks;
- Writing the documentation 3.2.6 - 17 weeks.

Chapter 4

Conclusions

After writing this document, the bases to initiate the development phase are established.

All the acquired knowledge about the current state of the art will permit the identification of techniques which will be of great assistance through the development of the project.

The technologies chosen (namely Python, Django and most likely, VMBuilder or GHOST), should not be of any problems when it comes to learning how to use them, since there is extensive documentation available and in the case of GHOST, specialized techniques working in CICA can provide technical assistance, should it come to that.

The calendarization should work as a strong general guideline for the tasks to perform and no great mishaps are foreseen to happen. Nonetheless, extreme caution must be had regarding the usability tests, as they may change the development process in case something does not turn out to be as it was supposed to.

Finally, all the objectives regarding the study of the problem and the state of the art have been completed, as well as the workplan for the development and dissertation writing. The only thing missing are some details pertaining some technologies, since there is still a need to discuss some aspects with expert technicians at CICA, but that issue should be resolved in the near future.

References

[Ama] Amazon. Amazon EC2 FAQ - what is a "EC2 Compute Unit" and why did you introduce it? http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it. Last accessed 16 July 2011.

[BÓ8] Marc-Elian Bégin. An EGEE comparative study: Grids and clouds - evolution or revolution. Technical report, CERN - Engeneering and Equipment Data Management Service, June 2008.

[BAFB05] M. Baker, A. Apon, C. Ferner, and J. Brown. Emerging grid standards. *Computer*, 38(4):43 – 50, april 2005.

[BK07] M. Bachle and P. Kirchberg. Ruby on rails. *Software, IEEE*, 24(6):105 –108, nov.-dec. 2007.

[BLDGS04] Miguel Bote-Lorenzo, Yannis Dimitriadis, and Eduardo Gómez-Sánchez. Grid characteristics and uses: A grid definition. In Francisco Fernández Rivera, Marian Bubak, Andrés Gómez Tato, and Ramón Doallo, editors, *Grid Computing*, volume 2970 of *Lecture Notes in Computer Science*, pages 291–298. Springer Berlin / Heidelberg, 2004.

[BV08] Jaijit Bhattacharya and Sushant Vashistha. Utility computing-based framework for e-governance. In *Proceedings of the 2nd international conference on Theory and practice of electronic governance*, ICEGOV '08, pages 303–309, New York, NY, USA, 2008. ACM.

[BYV+09] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[Car09] Nicholas Carr. *The Big Switch: Rewiring the World, from Edison to Google*. W.W. Norton & Company, 2009.

[Car11] Nuno Cardoso. Virtual clusters sustained by cloud computing infrastructures. Master's thesis, FEUP - Faculdade de Engenharia da Universidade do Porto, 2011.

[CCa] Rackspace Cloud Computing. Open Stack - Open Source Cloud Computing Software. http://stackoverflow.com/questions/209470/can-i-use-python-as-a-bash-replacement. Last accessed 16 July 2011.

[CCb] Inc. Cunningham & Cunningham. Don't Repeat Yourself. http://c2.com/cgi/wiki?DontRepeatYourself. Last accessed 16 July 2011.

[CER] CERN. gLite - Lightweight Middleware for Grid Computing. http://glite.cern.ch/. Last accessed 16 July 2011.

[CGH09] Damien Cerbelaud, Shishir Garg, and Jeremy Huylebroeck. Opening the clouds: qualitative overview of the state-of-the-art open source vm-based cloud management platforms. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 22:1–22:8, New York, NY, USA, 2009. Springer-Verlag New York, Inc.

[Con] Linux Config. Easy way to create a debian package and local package repository. http://linuxconfig.org/easy-way-to-create-a-debian-package-and-local-package-repository. Last accessed 16 July 2011.

[Deba] Debian. Debian - How to set up a Debian repository. http://wiki.debian.org/HowToSetupADebianRepository. Last accessed 16 July 2011.

[Debb] Debian. The Debian GNU/Linux FAQ - Basics of the Debian package management system. http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html. Last accessed 16 July 2011.

[Debc] Debian. VMBuilder - Debian Wiki. http://wiki.debian.org/VMBuilder. Last accessed 16 July 2011.

[FFK+06] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual clusters for grid communities. *Cluster Computing and the Grid, IEEE International Symposium on*, pages 513–520, 2006.

[Fos02] Ian Foster. What is the grid? a three point checklist. *Grid Today*, 1(6):22–25, 2002.

[GKC+] Jeremy Geelan, Markus Klems, Reuven Cohen, Jeff Kaplan, Douglas Gourlay, Praising Gaw, Damon Edwards, Brian de Haaff, Ben Kepes, Kirill Sheynkman, Omar Sultan, Kevin Hartig, Jan Pritzker, Trevor Doerksen, Thorsten von Eicken, Paul Wallis, Michael Sheehan, Don Dodge, Aaron Ricadela, Bill Martin, Ben Kepes, and Irving W. Berger. Twenty-One Experts Define Cloud Computing. http://virtualization.sys-con.com/node/612375. Last accessed in 16 July 2011.

[GWQF10] Thilina Gunarathne, Tak-Lon Wu, Judy Qiu, and Geoffrey Fox. Cloud computing paradigms for pleasingly parallel biomedical applications. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 460–469, New York, NY, USA, 2010. ACM.

[Hay08] Brian Hayes. Cloud computing. *Commun. ACM*, 51:9–11, July 2008.

[Haz08] Scott Hazelhurst. Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, pages 94–103, New York, NY, USA, 2008. ACM.

[HH] David Heinemeier Hansson. Ruby on Rails. http://rubyonrails.org/. Last accessed 16 July 2011.

[HP10] Joel Hollingsworth and David J. Powell. Teaching web programming using the google cloud. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 76:1–76:5, New York, NY, USA, 2010. ACM.

[IDE+06] Alexandru Iosup, Catalin Dumitrescu, Dick Epema, Hui Li, and Lex Wolters. How are real grids used? the analysis of four grid traces and its implications. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, GRID '06, pages 262–269, Washington, DC, USA, 2006. IEEE Computer Society.

[KF98] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[KF08] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 301 –308, dec. 2008.

[KG09] Eric Knorr and Galen Gruman. What cloud computing really means. http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031, 2009. Last accessed 16 July 2011.

[Lif] Lifehacker. Use SUSE Studio to Build a Linux OS From Scratch. http://lifehacker.com/5370209/use-suse-studio-to-build-a-linux-os-from-scratch. Last accessed 16 July 2011.

[McF] Paul McFedries. Ieee spectrum - Inside Technology. http://spectrum.ieee.org/computing/hardware/the-cloud-is-the-computer. Last accessed 16 July 2011.

[Mic] Microsoft. Windows Azure | Microsoft Paas | Cloud Services | Application Hosting. http://www.microsoft.com/windowsazure/. Last accessed 16 July 2011.

[NMM07] Hideo Nishimura, Naoya Maruyama, and Satoshi Matsuoka. Virtual clusters on the fly - fast, scalable, and flexible installation. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:549–556, 2007.

[Nov] Inc. Novell. SUSE Studio. http://susestudio.com/help. Last accessed 16 July 2011.

[NS07] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 265–278, New York, NY, USA, 2007. ACM.

[Ope] OpenNebula. OpenNebula. http://opennebula.org/about:faq#what_is_opennebula. Last accessed 16 July 2011.

[Par] Keith Parkansky. How to install Linux software programs using Debian Linux package management tools. http://www.aboutdebian.com/packages.htm. Last accessed 16 July 2011.

[Pin10] Jorge Fernando Maciel Rodrigues Ruão Pinheiro. Interligação de infra-estruturas de computação de elevado desempenho heterogéneas recorrendo a um super escalonador. Master's thesis, FEUP - Faculdade de Engenharia da Universidade do Porto, 2010.

[Rad] Radified. Radified Guide to Norton Ghost (Symantec) - A Tutorial on How to Create, Save & Restore Hard Drive Back-up Ghost Images. http://ghost.radified.com/. Last accessed 16 July 2011.

[Rap04] M. A. Rappa. The utility business model and the future of computing services. *IBM Syst. J.*, 43:32–42, January 2004.

[Res] Cluster Resources. Cluster Resources: Moab Cluster Software Suite. http://www.clusterresources.com/products/moab-cluster-suite.php. Last accessed 16 July 2011.

[Rev] Linux Reviews. Linux Reviews - dpkg. http://linuxreviews.org/man/dpkg/. Last accessed 16 July 2011.

[RW04] J. W. Ross and G. Westerman. Preparing for utility computing: The role of it architecture and relationship management. *IBM Syst. J.*, 43:5–19, January 2004.

[sei] stack exchange inc. Can I use Python as a bash replacement? http://stackoverflow.com/questions/209470/can-i-use-python-as-a-bash-replacement. Last accessed 16 July 2011.

[Sto07] Heinz Stockinger. Defining the grid: a snapshot on the current view. *The Journal of Supercomputing*, 42:3–17, 2007.

[Sun] Karishma Sundaram. Bright Hub - The Hub for Bright Minds. http://www.brighthub.com/environment/green-computing/articles/68785.aspx. Last accessed 16 July 2011.

[Sup] About.com PC Support. ISO File Definition - What is an ISO File. http://pcsupport.about.com/od/termsi/g/isofile.htm. Last accessed 16 July 2011.

[Sym] Symantec. Ghost Tutorials. http://www.symantec.com/business/support/ghosttutorial/GhostTutorials.html. Last accessed 16 July 2011.

[Tea] Condor Team. Condor Project Homepage. http://www.cs.wisc.edu/condor/. Last accessed 16 July 2011.

[Ubua] Ubuntu. JeOSVMBuilder - Community Ubuntu Documentation. https://help.ubuntu.com/community/JeOSVMBuilder. Last accessed 16 July 2011.

[Ubub] Ubuntu. UEC/CreateYourImage - Community Ubuntu Documentation. https://help.ubuntu.com/community/UEC/CreateYourImage. Last accessed 16 July 2011.

[VRMCL08] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, December 2008.

REFERENCES

[Web] Webopedia. What is software package? - A Word Definition from the Webopedia Computer Dictionary. http://www.webopedia.com/TERM/S/software_package.html. Last accessed 16 July 2011.

[Wei07] Aaron Weiss. Computing in the clouds. *netWorker*, 11:16–25, December 2007.

[ZKFF05] Xuehai Zhang, Katarzyna Keahey, Ian Foster, and Timothy Freeman. Virtual cluster workspaces for grid applications. Technical report, 2005.

REFERENCES

Appendix A

Grids VS. Clouds

Feature	Grid	Cloud
Resource Sharing	Collaboration (VOs, fair share).	Assigned resources are not shared.
Resource Heterogeneity	Aggregation of heterogeneous resources.	Aggregation of heterogeneous resources.
Virtualization	Virtualization of data and computing resources.	Virtualization of hardware and software platforms.
Security	Security through credential delegations.	Security through isolation.
High Level Services	Plenty of high level services.	No high level services defined yet.
Architecture	Service orientated.	User chosen architecture.
Software Dependencies	Application domain-dependent software.	Application domain-independent software.
Platform Awareness	The client software must be Grid-enabled.	The SP software works on a customized environment.
Software Workflow	Applications require a predefined workflow of services.	Workflow is not essential for most applications.
Scalability	Nodes and sites scalability.	Nodes, sites, and hardware scalability.
Self-Management	Reconfigurability.	Reconfigurability, self-healing.
Centralization Degree	Decentralized control.	Centralized control (until now).
Usability	Hard to manage.	User friendliness.
Standardization	Standardization and interoperability.	Lack of standards for Clouds interoperability.
User Access	Access transparency for the end user.	Access transparency for the end user.
Payment Model	Rigid.	Flexible.
QoS Guarantees	Limited support, often best-effort only.	Limited support, focused on availability and uptime.

Figure A.1: Comparing Grids and Clouds[VRMCL08]