INRIA

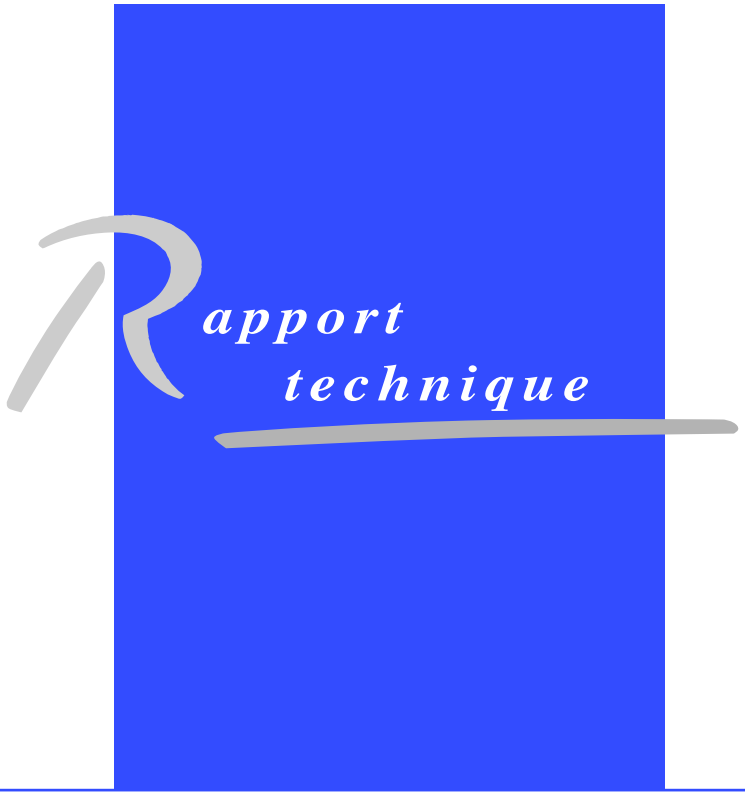
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Towards XtreemOS in the Clouds - Automatic Deployment of an XtreemOS Resource in a Nimbus Cloud

Eliana-Dina Tîrşa — Jerome Gallard — Pierre Riteau — Christine Morin — Yvon Jegou

N° ????

February 2010



ISSN 0249-0803 ISRN INRIA/RT--????--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Towards XtreemOS in the Clouds - Automatic Deployment of an XtreemOS Resource in a Nimbus Cloud

Eliana-Dina Tîrşa*, Jerome Gallard† , Pierre Riteau† , Christine Morin† , Yvon Jegou†

Thème : Cloud & Grid
Équipe-Projet XtreemOS

Rapport technique n° ???? — February 2010 — 15 pages

Abstract: Cloud is a new computing model which seamlessly provides on-demand resource lease as transparent services to the user and it is based on virtualization technologies. Grid[13] is a mature model based on he ideas of Virtual Organizations and collaborative sharing of resources between them. Nimbus is an open source cloud computing framework. XtreemOS is an open source, Linux basd, Grid operating system, with native suport for Virtual Organizations. The main purpose of this work is to use a Nimbus Cloud, in order to provide on-demand resource nodes for an XtreemOS Grid. Nimbus implements an extensible IaaS cloud based on Xen virtual machines.

Key-words: Grid, Cloud, virtualization, automatic deployment

* "Politehnica" University of Bucharest
† INRIA Rennes

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Towards XtreemOS in the Clouds - Automatic Deployment of an XtreemOS Resource in a Nimbus Cloud

Résumé : Abstract in French.

Mots-clés : Keywords in French.

Contents

1 Introduction

Cloud[11][12] is a new computing model which seamlessly provides on-demand resource lease as transparent services to the user and it is based on virtualization technologies. Grid is a mature yet not completely explored resource sharing model. It has been successfully implemented in academic and research institutions, but has not been considered as a viable business model. On the other hand, Cloud seems to have been adopted as a feasible technology for commercial use.

Grid and Cloud computing are currently seen as disjoint technologies, which are being developed independently. However, it is clear that they have many common points: they both offer access to remote hardware and software resources. Grid is based on a collaborative paradigm, although some economic models for the Grid have been envisioned. The Grid model assumes the existence of multiple Virtual Organizations, often having different governing and access policies. The Cloud computing model is usually associated with a business one, the customers paying for effective resource usage. Cloud environments are more dynamic, the (virtualized) resources are highly configurable and have a shorter lifespan than in the case of Grids. Thus, it seems natural to try to combine the (long studied) advantages of the Grid with the flexibility of the Cloud.

This report presents a first attempt towards interconnecting an XtreemOS[10] Grid and a Nimbus[15] Cloud environment, focused on providing higher accessibility of Grid resources.

2 Background

2.1 XtreemOS - Overview

XtreemOS [10] is a (Linux-based) Grid operating system with native support for virtual organizations and which is capable of running on a wide range of platforms (e.g. clusters, mobiles). A Grid operating system provides to the Grid the same functionalities that an operating system provides to a machine (e.g. abstraction of the hardware, resource management, and so on). XtreemOS provides support on every layer of the Grid (OGSA) architecture [13]:

1. on the *fabric layer*, XtreemOS provides support by using custom Linux kernel modules
2. on the *connectivity layer* it provides support for VO membership for users, resources and applications
3. on the *resource layer* XtreemOS provides application execution management
4. on the *collective layer* it provides the XtreemFS file system and VO management services
5. on the *application layer*, XtreemOS provides support for SAGA [1] and POSIX interfaces

All the required system services which provide the users with all the necessary Grid capabilities are integrated into a custom Linux kernel. The main Grid capabilities provided by XtreemOS are:

1. resource discovery
2. reservation management
3. job submission
4. checkpointing
5. event management
6. monitoring
7. dynamic resource allocation
8. fault-tolerant execution
9. data management
10. file replication
11. VO lifecycle management
12. VO entity management
13. VO accounting and audit trail management
14. policy management

The main software packages of XtreemOS consist of the following:

1. extensions to Linux for VO support and checkpointing
2. Linux Single System Image (SSI) for clusters
3. Linux kernel for embedded devices
4. an infrastructure for highly available and scalable services
5. data management packages
6. VO and security management packages
7. services for mobile devices

2.2 Cloud computing - Nimbus

The Cloud model is based on the idea of transparently delivering hardware and software resources as on demand services in order to satisfy the users computing needs, without sacrificing data security and QoS. In commercial clouds[17][18], the computing model is also associated with an economic one, the customers being charged on a pay-per-use basis.

Cloud computing is based on three concepts:

- IaaS (Infrastructure as a Service) - Users can rent the hardware resources(physical or virtual) as a fully outsourced service, instead of buying real servers, investing in data centers and network equipment.
- PaaS (Platform as a Service) - PaaS deliver software platforms integrated with generic modules, as services, providing support for development, deployment, hosting and maintenance of applications
- SaaS (Software as a Service) - SaaS separates the possession and ownership of software from its use. Software is delivered as a set of configurable services.

Nimbus is a cloud computing infrastructure project developed at Argonne National Lab. Its declared mission is to serve the needs of the scientific community, but it is also suitable for commercial applications. Nimbus provides an open source toolkit(CloudKit[15]) that comprises several services that allow clients to lease remote resources by mapping environments, or "workspaces"[15] (configured VMs), onto those resources, but also to turn their cluster into an Infrastructure-as-a-Service (IaaS) cloud.

There are several configuration possibilities and uses for Nimbus clusters. The most simple and rapid way of deploying Nimbus is presented in fig.1[14].

The "cluster" in the figure 1 contains four kind of nodes: the client node, the service node, the repository node and the VMM(virtual machine manager) node.

The Workspace Service(on the service nodes) is a site VM manager that can be invoked remotely, in order to deploy and control VMs. It supports two web service front end remote protocols: one is WSRF based and the other is EC2[17] WSDL compatible.

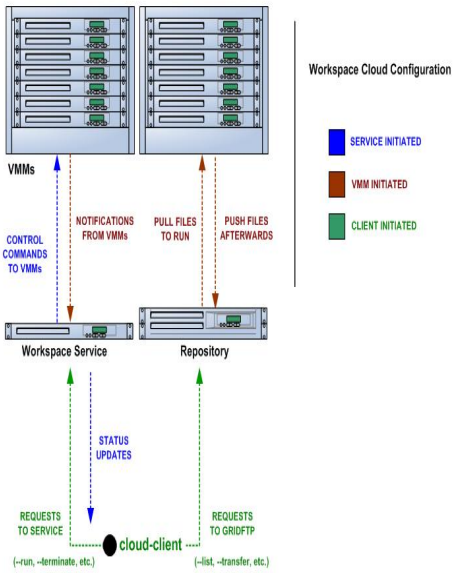


Figure 1: Nimbus deployment example.

The workspace service(WSRF) can be accessed by a reference(workspace) client, which is rather complex, but provides advanced configuration options. A lightweight version of the client(can only call a subset of the service functions), the "Cloud Client" is easy to use and configure and allows end-users to rapidly deploy "one-click" clusters.

Nimbus storage service, provides secure management of disk space where clients are each given a "repository" view of the VM images they own or they are allowed to launch. Globus GridFTP [11] must be installed on the "repository nodes"; the storage service provides support to any network file system that GridFTP can interface to.

The workspace control tools are installed on every VMM and are mainly used to start/stop/pause a VM, to reconstruct a VM image, to connect the VMs to the network, or provide contextualization information[15]. Workspace control implementation is based on Xen[16] hypervisor and KVM[20] virtualization technologies.(KVM support is not yet publicly available) .

2.3 XEN Virtualization Technology

Xen is an open source, hypervisor based virtualization solution that is designed as a standalone kernel. It supports both paravirtualization and full-virtualization.

In paravirtualization mode(as Xen is used inside Nimbus), both guest and host OS must be modified, to be aware of the virtualization layer. In other words, in order to have several guests running on the same machine, the host itself must boot on top of Xen. What used to be host OS(typically Linux), turned into privileged guest OS(Domain0), meanwhile the rest(UNIX based or Windows) of the guest OSs are unprivileged(DomainU), see Figure 2 [16]. Only

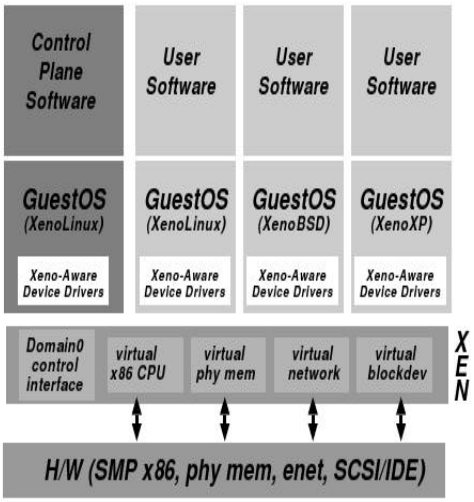


Figure 2: Machine running the Xen hypervisor, hosting different guest operating systems.

the privileged guest OS can perform management and I/O (both for its own use and on behalf of the other guests).

Xen doesn't require specialized hardware for virtualization, but it takes advantage of it, if present, supporting unmodified guest VMs if one is using either a VT capable Intel processor or an SVM capable AMD processor.

2.4 Related Work on integrating Grids and Clouds

Combining a Grid and a Cloud environment is a challenging task. One needs to overcome technical difficulties of bringing together two complex models, one based on collaborative resource sharing and the other on (on-demand) resource leasing. Some of the challenges regarding this interconnection are:

- different authorization, authentication, access and security mechanisms
- resource volatility in Cloud (Grid schedulers are designed to work in a less dynamic environment)
- the Cloud environment may be unaccustomed with the various policies of the Virtual Organizations inside Grids(such policies are sometimes hard to integrate in the first place)

The interoperability between Grids and Clouds is a new topic which has only recently started to draw the attention of researchers. Because of this, there are only a few attempts to constructing hybrid Grid-Cloud systems. A solution for application-level interoperability between Grids and Clouds based on extensions of the SAGA framework [1] has been proposed in [2]. A case for Clouds providing higher levels of abstraction to Grids has been made in [3]. A view which is closer to the one adopted in this report has been expressed in [4, 5, 8]. There, the authors consider the possibility of providing on-demand virtual

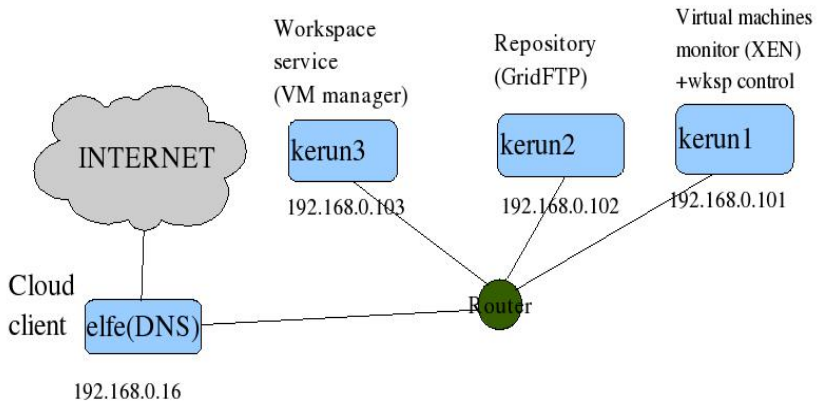


Figure 3: Nimbus local setup.

clusters (i.e. clusters of virtual machines) for any Grid system. Although the authors of these papers do not use an existing Cloud infrastructure for deploying the virtual machines (they develop their own mechanisms), it seems clear that they are heading in this direction [6]. PerfCloud [7] is a cloud environment built on top of a Grid system whose purpose is the same as in [4-6], i.e. to instantiate virtual clusters dynamically which will be part of the Grid.

In [9], a scenario in which XtreemOS [10] is used as the infrastructure for providing Cloud services is presented. This, however, differs from the perspective considered in this report, where a Cloud environment is used for dynamically deploying resource nodes in an XtreemOS Grid.

3 Experiment setup and initial configuration

3.1 Initial laboratory setup

We installed a minimal Nimbus cluster(fig2), similar to the generic one presented in fig.1.

We configured a local LAN. The VMM(with Xen hypervisor suport) was on kerun1, the image repository on kerun2 and the workspace service on kerun3. The Cloud Client was installed on a machine(elfe) that was connected to the Internet and also acted as a DNS server to the keruns machines.

3.2 XtreemOS on virtual machines inside the Nimbus cloud

The next step was to have an XOS core and an XOS resource inside the cloud[see fig. 3], as Xen VM's. There are online repositories containing pre-built Xen images for many Linux distributions, but they didn't include XtreemOS; thus, we had to create one. Two colleagues in the team had built KVM and Xbox VMs for XtreemOS core and resource. At the time, there was no automatic script for configuring XtreemOS, so we decided to base our Xen machines on a preconfigured image (the alternative being starting from an ISO or a freshly installed physical partition). We decided on using the KVM VMs.

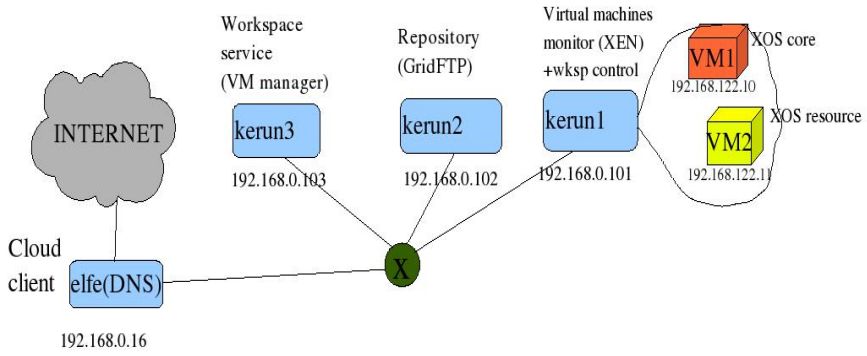


Figure 4: XOS core and XOS resource as VMs in the cloud.

After creating the XOS images, we changed the preconfigured(static) IPs of the VMs, in order to match the DHCP server settings on the VMM machine(kerun1). The XtreemOS configuration script was still work in progress, so we had to do it manually(there are many files that require modification).

The XtreemOS daemon(xosd[RRR]) requires kernel connectors inside the kernel. Otherwise it wouldn't start. The interaction of the XtreemOS Xen DomU with the underlying hardware is mediated by the XEN hypervisor, The XtreemOS kernel already has kernel connectors compiled in it, but also requires kernel connectors inside the XEN kernel. Since the Xen kernel doesn't have kernel connectors enabled by default, we had to insert a module (cn) in the Xen kernel. This requirement was not clearly documented at the time of performing the work described in this report.

Initially, we encountered a problem, as the XtreemOS daemon(xosd[RRR]) failed to start. We assumed that it had something to do with erroneous network configuration(IPs/ports) inside the Xen VMs(the logs stated a binding error to some socket). In order to eliminate the (supposed) cause of the problem, we modified the DHCP server settings on the VMM machine and kept the preconfigured IPs(of the former KVM images) unmodified. However, this did not eliminate the problem. After further researching the issue, we discovered that The connectors were already inside XtreemOS kernel, but...[TODO: link this somehow with the way Xen Dom0/DomU work - to be described in the section about XEN] We inserted the required module(cn) in the Xen kernel (some may argue that this is not enough[RRR - xos admin guide]) and after that xosd started.

We managed to have an XOS core and an XOS resource node VMs running and communicating inside the cloud[fig 3](we were able to submit jobs from the core node to the resource node). The step forward was to have the XOS core installed on a physical machine and the XOS resource as a VM in the cloud[fig. 4]. That meant further configuring(by hand) the XOS resource VM(the IP and/or the hostname of the XOS core needed to be changed in several files), and in the case of deploying multiple XOS resource VMs, the IP/hostname of the resource needed to be updated as well. Let alone the fact that many XtreemOS services don't run at boot and need to be (re)started and checked.

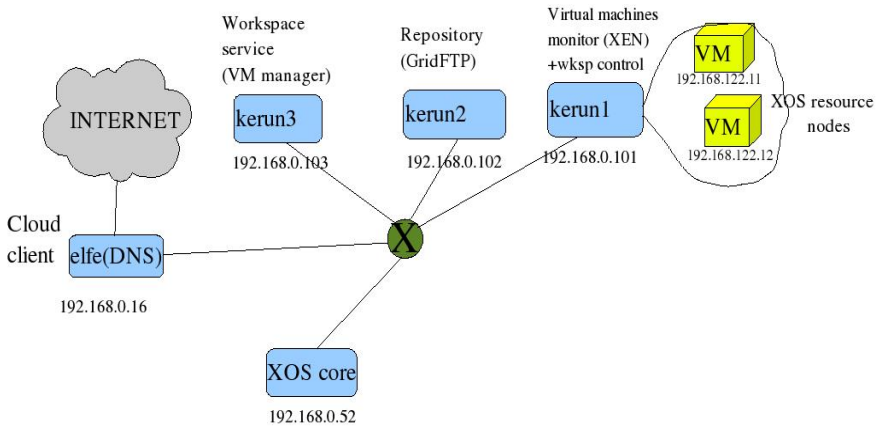


Figure 5: Virtual XOS resources inside cloud, XOS core on phisycal machine, in same LAN as VMM.

Fortunately, by the time we reached this phase, a first version of the automatic configuration script was ready.

4 Automatic scripts for deploying an XtreemOS virtualized resource inside a Nimbus cloud

4.1 Previous work - initial XtreemOS configuration tool

The initial set of scripts was developed(by another team member) in parallel with the early configurations described in this report. It was the first attempt towards an automatic configuration of an XtreemOS machine. The tool archive contains configuration scripts, variables and services definition files and template directories. The main components of the configuration tool are:

- configureXos: first part of the configuration script. The configuration of a RCA[RRR] node stops here(although some services may have to be restarted by hand afterwards) but for other nodes(an XOS resource is never an RCA node) the certification phase followed.
- finishConfig: second part of the configuration script. After the resource received the certificate, some more settings were in order.
- confirmResource(RCA node only): script to confirm a resource (loop until a resource certificate request is received). Testing/template directories: conf, root, etc
- defs: variable definitions(VO name, core/res. IP and hostmane, ports for different services)
- globalDefs: configuration of XtreemOS services on the Grid (resource, core, client)

We used this tool for configuring a freshly installed XOS core on a physical node, that we placed(for simplicity) in the same LAN as the other machines in the Nimbus cloud(the VMM, the repository, the workspace service and the client).

4.2 Automatic deployment of an XtreemOS resource in a Nimbus cloud - addressed issues

The initial configuration scripts described in the previous section were the base for the automatic deployment tool of an XtreemOS resource node as VM in the Nimbus cloud. The (achieved) goal was to have a one-click deployment (and configuration) of an XtreemOS resource in the cloud, which can be used for executing jobs immediately after the automatic configuration has finished.

The automatic deployment tool consists in a set of scripts that run on three locations: on the newly deployed XtreemOS resource VM(at boot), on the Nimbus Cloud Client machine and on the XtreemOS core (and RCA) node(that is the same, in our experiments, with the machine that initiates the deployment); the Nimbus cloud client can be easily located on the same machine as the XtreemOS core. In order to present the key features of the automatic deployment tool, in the following paragraphs we will describe the technical issues of XtreemOS automatic configuration and the way we addressed them.

VM monitor machines in the cloud were configured to assign IPs via DHCP. Our automatic deployment tool retrieves the dynamically assigned IP of the XOS resource and updates the corresponding field in the definitions file of the configuration script. Based on a preconfigured /etc/hosts file, the deployment tool sets the hostname to the one associated with provided IP. The IP(s) and hostname(s) of the machine(s) containing XOS core services must be preconfigured in the definitions file.

XtreemOS grid users are authenticated within a VO(virtual organization) using (XOS) Certificates that a trusted certificate authority(CDA) issues. Unfortunately, XOS Certificates cannot be issued automatically. A user has to apply for a certificate(using a web interface) and then wait until the VO administrator grants his request. As a result, each VM has to contain (before deployment) the certificates of the grid users that plan to use that virtual resource.

XtreemOS grid user authorization is based on a set of local policy rules and global user attribute rules. This rules specify the operations that a VO user is allowed to perform on a given node. The VO user and group must be associated with the local ones, as compulsory rules of the local policy. The problem is that sometimes, local policy is not configured, even after enforcing the mapping rules for user and group. That is because the mapping rules for the user do not come into effect immediately , and the group mapping rules cannot be enforced before the user ones. Also, when checking the policy rules, we have to be sure that the group mapping has become effective. This is not an issue when the mappings and the policy check are performed by hand, but in a script we have to provide a proper delay between the three operations(and loop until the policy check returns no error).

XtreemOS resource nodes have to be authenticated, also. A Resource Certificate Authority(RCA) issues certificates for resource nodes in an XtreemOS VO. A new resource has to apply for registration and only a registered resource

can request a certificate. Resource registration is a two-phase process. The applicant resource is saved in a "pending_list". The administrator of the RCA node has to confirm a resource in a pending list and then the resource gets registered.

In order to automatically register a virtual XtreemOS resource, the deployment of the VM has to be initiated from the RCA node(and the RCA should loop until it receives the application and then confirm it). The applying resource will loop until registered (there is a list that comprises all the registered resources in a VO) and then it will request a certificate(the certificated is issued automatically for an already registered node). In the setup we tested so far, the VMs local IPs are visible from the core(RCA) node. The local IP, assigned(dynamically) to a VM in a Nimbus Cloud, can be retrieved from the VM deployment command output. To enhance security, the RCA will use that IP in order to (automatically)register only an expected resource. This will be a problem when testing with VMs behind NAT.

A possible alternative solution for automatic registration of resources could be having a list of pre-registered resources(not implemented in XtreemOS). Yet, this may not be feasible in a Cloud environment, given the dynamicity of resources/IPs (favourite/static IPs can be assigned in some Commercial Clouds, but one has to pay for the period when that IP is not used).

Another problem regarding automatic resource registration is related to the applying command itself(rca_apply). Its output is a menu(presenting several options) and keyboard input is expected. The options may be justified by different use cases, but not in an automatic configuration setup. We modified the code behind the menu and provided a default action.

Resources can be added (as we described above) and removed seamlessly from an XtreemOS Grid. But they usually are not unregistered(the reason for that being that a node can temporarily fail and then re-join the Grid). When dealling with Cloud resources(limited life, different IP's), this may be a potential problem. The RCA may certificate a resource with a known IP(that is already registered), but operated by an malicious user.

In a previous section we mentioned the XtreemOS daemon(xosd). It provides the set of XtreemOS services that run on a given node. When a component gets updated, or a service modifies its state, usually the xosd daemon need to be restarted. This operation does not complete instantly. When restarting xosd inside a (configuration) script, one has to provide a delay and make sure that the xosd services are running before initiating other commands.

XtreemFS is a distributed, replicated, object based file system that XtreemOS uses. XtreemFS server contains three services: DIR - Directory service, MRC - Metadata and Replica catalog, OSD - Object Storage Device. The DIR service is used by the other two in order to find other servers, map grid users to volumes, resolve volume names. When (re)starting XtreemFS, it is recommended that the services are started in order: DIR, MRC, OSD, because of their dependences. We had to make sure (insert a delay, loop to check) that when a service is started, the one it is depending on, is up and running. The XtreemFS server runs only on core nodes, so this is not a problem on resource nodes(XtreemFS client runs there). Each grid user has an XtreemFS volume, where it can find all the data it stored there, regardless the node. The XtreemFS volume for a user is mounted on a special directory that is not always created automatically. When dealing with automatic deployment, we have to create this special direc-

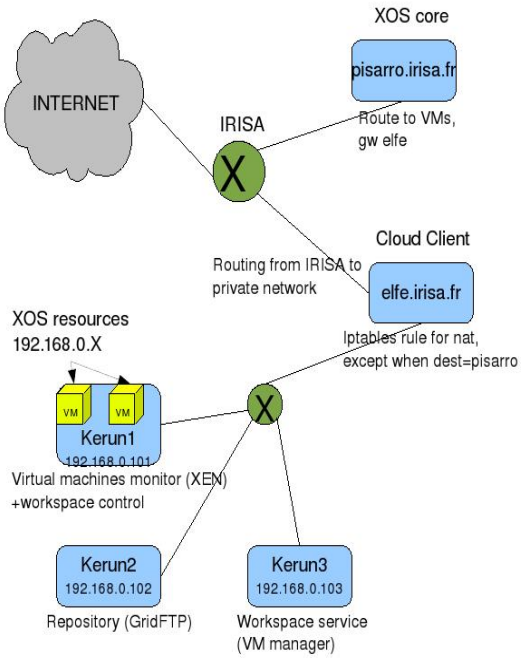


Figure 6: XOS core outside cloud, XOS resource as VM inside cloud.

tory before deployment, one for every Grid user that will use the XtreemOS virtual resource.

5 Validation Testing

The test setup[fig. 6] was somehow similar to the one presented in the fig.5. The difference is that the physical machine that contained the XOS core was on a different network than the (physical) machines in the cloud. On the other hand, even if the VMs were assigned local IPs, we configured the Cloud Client machine (that also acted as a gateway and a DNS) to route these IPs to the XOS core machine and translate(NAT) them to other destinations.

We configured a Xen VM containing an XtreemOS resource node. We installed the deployment scripts and configuration files(as described in the previous sections) and we set them to run at boot time. On the XOS core machine(that was also a RCA node) we installed the VM deployment initiation script. When this script runs, we provide the VM name and duration. The output of the deployment command also contains the IP assigned to the VM. On the same machine(core), we added another script(that runs immediately after the deployment one) that uses this IP and based on it registers a resource(after it applies for registration). The deployment script on the XOS core machine connects via ssh(using RSA keys) to the Nimbus Cloud Client machine and issues the run command of a virtual machine in the cloud, according to the name and duration provided by the user.

Note that all these steps are performed by issuing a single command on the XOS core machine. After approximately 2 minutes(without any other user intervention), the XOS Resource VM deployment(including resource registration and certification) is completed. We can use the new virtual resource for executing Grid jobs. As a relevant test, we considered submitting a simple job runs the ifconfig command. In this way, we could verify that the job was really executed on the newly deployed virtual machine(be checking the IP in the job output file). We saved the output file of the job inside the grid user XtreemFS volume. Thus, we didn't have to explicitly connect to the VM in order to retrieve the output. We just accesed the user XtreemFS volume from the core machine.

6 Conclusions and Future Work

In this report we presented the automatic deployment of an XtreemOS resource, as a virtual machine in a Nimbus Cloud. We also mentioned the technical difficulties we had to overcome in order to achieve our goal. In our test setup, the XtreemOS core machine that initiated the deployment was in a different network than the VMs in the cloud, yet the VMs' IPs were routed to the XOS core (and NATed to the rest of the Internet).

As a first step in our future work, we plan to have the core communicate and automatically deploy NATed VMs. This scenario is similar to the way Amazon cloud services are used. We also envisage an XtreemOS resource deployment in an Amazon cloud.

References

[1] H. Kaiser, A. Merzky, S. Hirmer, G. Allen, "The SAGA C++ Reference Implementation - Lessons Learnt from Juggling with Seemingly Contradictory Goals", OOPSLA/LCSD 2006.

[2] A. Merzky, K. Stamou, S. Jha, "Application Level Interoperability between Clouds and Grids", Workshops of the Grid and Pervasive Computing Conference, pp. 143-150, 2009.

[3] S. Jha, A. Merzky, G. Fox, "Clouds Provide Grids With Higher Levels of Abstractions and Support for Explicit Usage Modes", Concurrency and Computation:Practice and Experience, Vol21, no 8, 1087-1108, 2009.

[4] M. Rodríguez, D. Tapiador, J. Fontan, E. Huedo, R. S. Montero, I. M. Llorente, "Dynamic Provisioning of Virtual Clusters for Grid Computing", Euro-Par Workshops, pp. 23-32, 2008.

[5] M. L. Bertogna, E. Grosclaude, M. R. Naiouf, A. De Giusti, E. Luque, "Dynamic on Demand Virtual Clusters in Grid", Euro-Par Workshops, pp. 13-22, 2008.

[6] I. M. Llorente, R. Moreno-Vozmediano, R. S. Montero, "Cloud Computing for On-Demand Grid Resource Provisioning", Advances in Parallel Computing, IOS Press, in press.

[7] E. P. Mancini, M. Rak, U. Villano, "PerfCloud: GRID Services for Performance-Oriented Development of Cloud Computing Applications", IEEE Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009.

[8] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, X. Zhang, "Virtual Clusters for Grid Communities", Proceedings of the IEEE Intl. Symposium on Cluster Computing and the Grid, pp. 513-520, 2006.

[9] C. Morin, J. Galard, Y. Jegou, P. Riteau, "Clouds: a New Playground for the XtreemOS Grid Operating System", Parallel Processing Letters, vol. 19 (3), 2009.

[10] C. Morin, "XtreemOS: a Grid Operating System Making your Computer Ready for Participating in Virtual Organizations", Proc. of the IEEE International Symposium on Object and component-oriented Real-time Distributed Computing, pp. 393-402, 2007.

[11] A. Weiss, Computing in the Clouds. netWorker, 11(4):16-25, 2007.

[12] B. Hayes. "Cloud Computing". Communications of the ACM, 51(7):9-11, 2008.

[13] I. Foster, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, pp1-4, 2001

[14] Nimbus open source IaaS Cloud computing. http://workspace.globus.org/.

[15] K. Keahey , T. Freeman, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications", Cloud Computing and Its Applications 2008 (CCA-08), Chicago, IL. October 2008.

[16] P. Barham, B. Dragovic, K. Fraser, S. Hand, Steven. T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization", Proceedings of the nineteenth ACM symposium on Operating systems principles, pp.164-177, 2003.

[17] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/.

[18] FlexiScale - Utility Computing On-Demand. http://www.flexiscale.com/.

[19] W. Allcock, "GridFTP: Protocol extensions to FTP for the Grid", Global Grid ForumGFD-R-P. 020, 2003.

[20] A Kivity, Y Kamay, D Laor, U Lublin, A Liguori, "kvm: the Linux virtual machine monitor", Proceedings of the Ottawa Linux Symposium, pp. 225-230, 2007.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803