

Towards XtreemOS in the Clouds – Automatic Deployment of XtreemOS Resources in a Nimbus Cloud

Eliana-Dina Tirsa, Pierre Riteau, Jérôme Gallard, Christine Morin, Yvon Jégou

▶ **To cite this version:**

Eliana-Dina Tirsa, Pierre Riteau, Jérôme Gallard, Christine Morin, Yvon Jégou. Towards XtreemOS in the Clouds – Automatic Deployment of XtreemOS Resources in a Nimbus Cloud. [Technical Report] RT-0395, INRIA. 2010, pp.15. inria-00524843

HAL Id: inria-00524843
https://inria.hal.science/inria-00524843

Submitted on 8 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

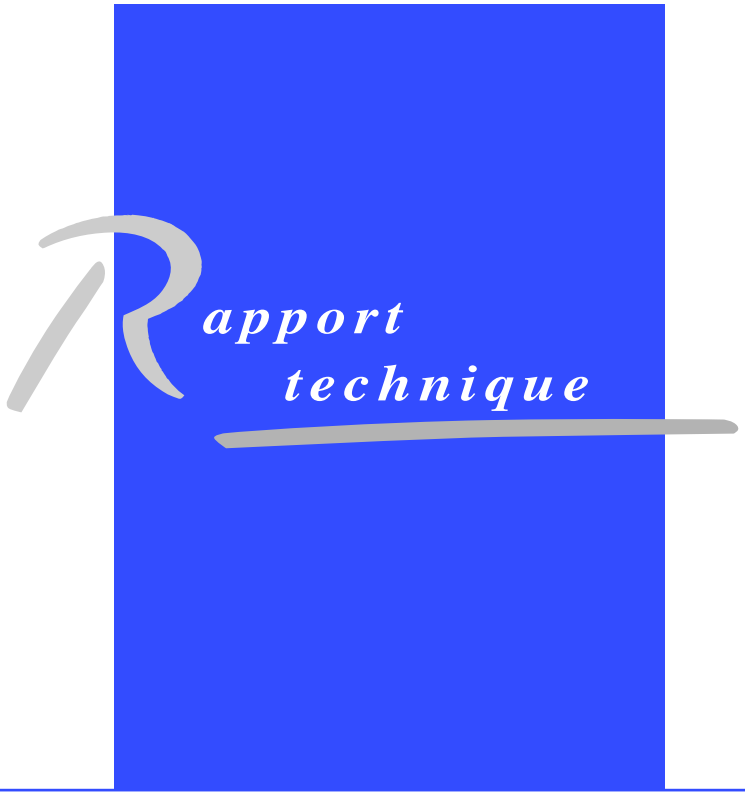
Towards XtreemOS in the Clouds – Automatic Deployment of XtreemOS Resources in a Nimbus Cloud

Eliana-Dina Tîrşa — Pierre Riteau — Jérôme Gallard — Christine Morin — Yvon Jégou

N° 0395

Octobre 2010

Distributed and High Performance Computing



ISSN 0249-0803 ISRN INRIA/RT--0395--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 XtreemOS Resources in a Nimbus Cloud

Eliana-Dina Tîrşa*, Pierre Riteau†‡ , Jérôme Gallard‡ , Christine Morin‡ , Yvon Jégou‡

Theme : Distributed and High Performance Computing
Équipe-Projet Myriads

Rapport technique n° 0395 — Octobre 2010 — 15 pages

Abstract: Grid computing is a well-known and mature model of distributed computing based on the idea of collaborative sharing of resources between multiple users. Grids are generally used for executing large and long running applications. These applications are set up and configured to work on specific kinds of hardware and software. Compared to grid computing, cloud computing is an emerging model based on virtualization technologies, which provides on-demand resource leases as transparent services to the user. Distributed applications can easily run inside clouds, in already configured hardware and software environments. Whereas grids allow users to run very large and time-consuming applications on specific and dedicated hardware, cloud computing allows users to run applications in order to get fast results with a minimum of hardware and software configuration.

Our vision is that, from a resource point of view, cloud computing is more dynamic and flexible than grid computing. We study and implement mechanisms in order to provide more dynamicity in grids by automatically extending them with cloud resources. Our work is based on XtreemOS, an open source, Linux-based grid operating system and Nimbus, an open source cloud computing framework. In this document, we present the design and implementation of a system using a Nimbus cloud in order to automatically provide on-demand resource nodes for an XtreemOS grid.

Key-words: Cloud Computing, Virtualization, Grid Extension, Automatic Deployment

The INRIA team carries out this research work in the framework of the XtreemOS project partially funded by the European Commission under contract #FP6-033576.

* "Politehnica" University of Bucharest – eliana.tirsa@cs.pub.ro
† Université de Rennes 1, IRISA, Rennes, France – Pierre.Riteau@irisa.fr
‡ INRIA Rennes – Bretagne Atlantique, Rennes, France – firstname.lastname@inria.fr

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

Vers XtreemOS sur les *clouds* – Déploiement automatique de ressources XtreemOS dans un *cloud* Nimbus

Résumé : La grille informatique est un modèle d'informatique distribuée répandu et mature, fondé sur le partage collectif des ressources entre multiples utilisateurs. Les grilles sont généralement utilisées pour exécuter des applications de taille importante sur une longue durée. Ces applications sont installées et configurées afin de fonctionner sur une couche matérielle et logicielle précise.

Comparé à la grille informatique, le *cloud computing* est un modèle émergeant fondé sur les technologies de virtualisation qui offre aux utilisateurs la possibilité de réserver des ressources, à la demande, à travers des services transparents. Les applications distribuées peuvent facilement s'exécuter dans les *clouds*, en s'appuyant sur des environnements matériels et logiciels préconfigurés. Alors que les grilles informatiques permettent aux utilisateurs d'exécuter de larges applications sur une longue durée en utilisant du matériel spécifique et dédié, le *cloud computing* permet aux utilisateurs d'exécuter des applications pour obtenir des résultats rapidement avec un minimum de configuration logicielle et matérielle.

Notre vision est que, d'un point de vue des ressources, le *cloud computing* est plus dynamique et flexible que la grille informatique. Nous étudions et mettons en œuvre des mécanismes afin d'offrir plus de dynamicité aux grilles, en les étendant automatiquement avec des ressources des *clouds*. Notre travail se centre sur XtreemOS, un système d'exploitation libre pour grille fondé sur Linux, et Nimbus, un système libre de *cloud computing*. Dans ce document, nous décrivons l'architecture et la mise en œuvre d'un système utilisant un *cloud* Nimbus de façon automatique afin d'offrir des ressources à la demande à une grille XtreemOS.

Mots-clés : *Cloud computing*, Virtualisation, Extension de grilles, Déploiement automatique

Contents

1 Introduction **3**

2 Background **4**
2.1 XtreemOS – Overview 4
2.2 Cloud computing – Nimbus 5
2.3 Xen Virtualization Technology 6
2.4 Related Work on Integrating Grids and Clouds 7

3 Automatic Deployment of XtreemOS Resources in Clouds **8**
3.1 Creation of virtualized XtreemOS resources 8
3.2 Addition of virtualized resources to an XtreemOS grid 8
3.3 Usage of virtualized XtreemOS resources 9

4 Implementation with a Nimbus Cloud **10**
4.1 Creation of new XtreemOS resources in Nimbus 10
4.2 Addition of new resources to our XtreemOS grid 11
4.2.1 Security issues 11
4.3 Usage of our new XtreemOS resources 12

5 Validation Testing **12**

6 Conclusions and Future Work **12**

1 Introduction

Cloud computing [18, 7] is an emerging computing model which seamlessly provides on-demand resource leases as transparent services to the user and 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 computing 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 grids have been envisioned. The grid model assumes the existence of multiple Virtual Organizations (VOs), 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 provided virtual resources are highly configurable and have a shorter lifespan than in the case of grids. An interoperability solution between grids and clouds at the resource management level can improve both grid and cloud usage from an administrator and a user point of view [16]: grids can take advantage of resource elasticity from clouds and clouds can leverage existing collaboration services from grids.

This report presents a first attempt towards interconnecting XtreemOS [15], a Linux-based grid operating system, and Nimbus [10], an open-source cloud environment, focused on providing higher accessibility of grid resources. Section 2 makes a general background related to the XtreemOS grid operating system and the Nimbus cloud system. Section 3 presents and discusses our model of automatic deployment of XtreemOS resources in clouds. An implementation of our model, presented in Section 4, is made using a Nimbus cloud. We present a validation of our prototype in Section 5 followed by a conclusion and perspectives of work in Section 6.

2 Background

2.1 XtreemOS – Overview

XtreemOS [15] 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 [6]:

- *Fabric layer*: XtreemOS provides support by using custom Linux kernel modules.
- *Connectivity layer*: it provides support for VO membership for users, resources and applications.
- *Resource layer*: XtreemOS provides application execution management.
- *Collective layer*: it provides the XtreemFS file system and VO management services. A VO owner can define policies, stating permission and usage rules for VO resources.

  XtreemFS is the XtreemOS grid file system. It allows the federation of multiple data stores located on different administrative domains and provides secure access to stored files to VO members, whatever their locations. XtreemFS allows automatic data replication management for data access efficiency and availability.
- *Application layer*: XtreemOS provides support for SAGA [9] 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: (i) resource management, (ii) collaborative management services, (iii) distributed file system.

The resource management takes into account the resource discovery, allocation, and monitoring. In addition, node reservations can be made.

The collaborative management services is done via the VO management: VO creation, VO lifecycle, VO accounting, and VO policies.

XtreemFS is the XtreemOS distributed file system. XtreemFS provides fault-tolerant mechanisms like file replication.

Finally, a job submitted on a VO, according to the VO policies, will take advantage of XtreemFS and the resources belonging to that VO.

The main software packages of XtreemOS consist of the following:

- extensions to Linux for VO support and checkpointing,
- Linux Single System Image (SSI) for clusters,
- Linux kernel for embedded devices,
- an infrastructure for highly available and scalable services,
- data management packages,
- VO and security management packages,
- 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 [1, 2], 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 virtual hardware resources as a fully outsourced service, instead of buying real servers, investing in data centers and network equipment.
- PaaS (Platform as a Service): PaaS delivers 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 [10] that comprises several services that allow clients to lease remote resources by mapping environments, or *workspaces* [10] (configured VMs), onto those resources, but also to turn their cluster into an IaaS provider.

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

A typical Nimbus cloud contains three kind of nodes: the client node, the service node, and the VMM (virtual machine manager) nodes.

The Workspace Service, running on the service node, is a site VM manager that can be invoked remotely, in order to deploy and control VMs. It supports two web service frontends: one is WSRF based and the other is EC2 [1] WSDL compatible.

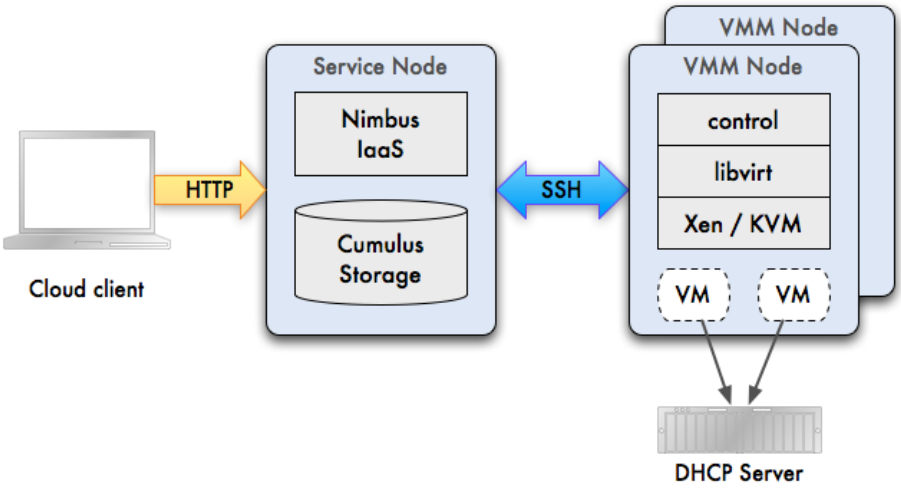


Figure 1: Nimbus deployment example.
This picture comes from the Nimbus documentation:
`http://www.nimbusproject.org/docs/2.5/admin/z2c/index.html`

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

The Nimbus storage service provides secure management of VM images, where clients get a repository view of the images they own or are allowed to launch. The storage service provides an Amazon S3-compatible interface.

The workspace control tools are installed on every VMM and are mainly used to start/stop/pause a VM, to fetch VM images, to connect VMs to the network, and provide contextualization information [10]. The workspace control implementation is based on the Xen [3] hypervisor and KVM [11] virtualization technologies. In this document, we only consider the Xen hypervisor since this is the technology we used in our prototype system. However, our approach is not limited to Xen and would also be valid with KVM and other hypervisors able to run XtreemOS resource nodes.

2.3 Xen Virtualization Technology

Xen [3] is an open source hypervisor 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.

Basically, the Xen hypervisor is executed directly on top of the physical machine. A privileged operating system (host OS, Dom0) is run on top of the Xen hypervisor. The guest OSs are executed inside the unprivileged VMs (domU).

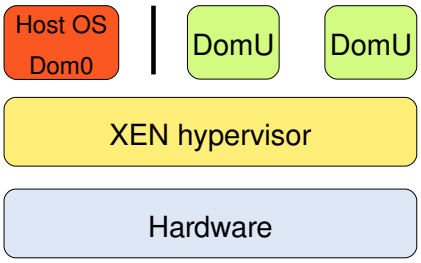


Figure 2: Machine running a Xen hypervisor, hosting different guest operating systems (DomU).

All privileged actions required by a guest OS are forwarded and executed by the host OS. Figure 2 presents the architecture of the Xen hypervisor.

Xen doesn't require specialized hardware for virtualization (a VT capable Intel processor or an AMD-V capable AMD processor), but it takes advantage of it, if present, supporting unmodified guest VMs: this is full-virtualization.

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 clouds (grid schedulers are designed to work in a less dynamic environment),
- cloud environments may be unaccustomed with the various policies of the VOs inside grids (such policies are sometimes hard to integrate in the first place).

The interoperability between grids and clouds is an emergent topic which has only recently started to draw the attention of researchers. Because of this, there are only a few attempts to construct hybrid grid-cloud systems. A solution for application-level interoperability between grids and clouds based on extensions of the SAGA framework [9] has been proposed in [14]. A case for clouds providing higher levels of abstraction to grids has been made in [8]. A view which is closer to the one adopted in this report has been expressed in [17, 4, 5]. There, the authors consider the possibility of providing on-demand virtual 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 [12]. PerfCloud [13] is a cloud environment built on top of a grid system whose purpose is the same as in [17, 4, 12], i.e. to instantiate virtual clusters dynamically which will be part of the grid.

This work is based on a previous analysis made in [16] in which the possibility of extending an XtreemOS grid operating system into a cloud system is studied.

3 Automatic Deployment of XtreemOS Resources in Clouds

An XtreemOS grid is composed of several kinds of resources: core nodes, compute resources and client resources. Core nodes run all the necessary services required by XtreemOS. Compute resources are the resources able to make the computations. Client resources are frontends from which users access the XtreemOS grid. As we want to automatically extend an XtreemOS grid with resources offered by a cloud provider, our work mainly focuses on the core and compute resources configuration, as depicted in Figure 3 (in this figure core nodes are represented by XOS core and compute resources are represented by XOS resources).

Basically, when needed (automatically or manually due to an administrator request) the core node should reserve and use external resources provided by one or multiple cloud providers. To do that, several steps have to be performed. First of all, the core service has to ask for VMs creation on the targeted cloud. Once the VMs are created, they need to be configured in order to be registered as new compute resources in the XtreemOS system and to enter into a VO. Finally, the VMs can be used, taking natively advantage of all the services provided by XtreemOS.

In the following, we consider clouds where VM images of XtreemOS are available and deployable. In addition, we consider cloud and network infrastructures as trusted. However, cloud infrastructures are shared with potentially malicious users.

3.1 Creation of virtualized XtreemOS resources

When the XtreemOS resource manager detects more resources are needed or if requested manually by an administrator, the core node makes a VMs creation request to the targeted cloud. This request should be made using a standard cloud interface, such as the Amazon EC2 WSDL interface. At the end of this operation a unique identifier and the IPs of the VMs are returned by the cloud to the core node. This identifier allows to trace the status of the VM (running, terminated, ...).

3.2 Addition of virtualized resources to an XtreemOS grid

Once the VMs are created, they have to be added as new available resources to the XtreemOS grid. XtreemOS provides native mechanisms to register resources based on a RCA (Resource Certification Authority). A RCA server is running on a core node. In the rest of this document, we will use the term XtreemOS core node and RCA interchangeably, assuming that the services running on an XtreemOS core node include a RCA server.

The registration services provided by XtreemOS was made for physical resources belonging to trusted environments. In that way, to add a new resource to a running XtreemOS grid, a compute resource asks the RCA for a certificate. Then, the RCA confirms the request, and the compute resource can obtain the certificate. At the end of this operation, the new resource is added to the XtreemOS grid.

We decide to use the same mechanism of authentication to register compute resources provided by clouds. In addition, thanks to the combination of unique identifiers and IPs, the system can check the status of the requested VMs.

When the new compute resources are registered into XtreemOS, they can be configured to belong to a VO. Only users belonging to this VO will be able to access to these new resources. Resources added into a VO are subject to the policies of this VO.

3.3 Usage of virtualized XtreemOS resources

Finally, VM resources are managed in the VO as regular resources: VO policies are applied both to VMs and regular resources. Jobs can be deployed on these resources and can take advantage of all the XtreemOS services. For instance, XtreemFS, the XtreemOS grid file system, allows applications to get their input and write their output on a volume accessible to all the nodes belonging to the same VO.

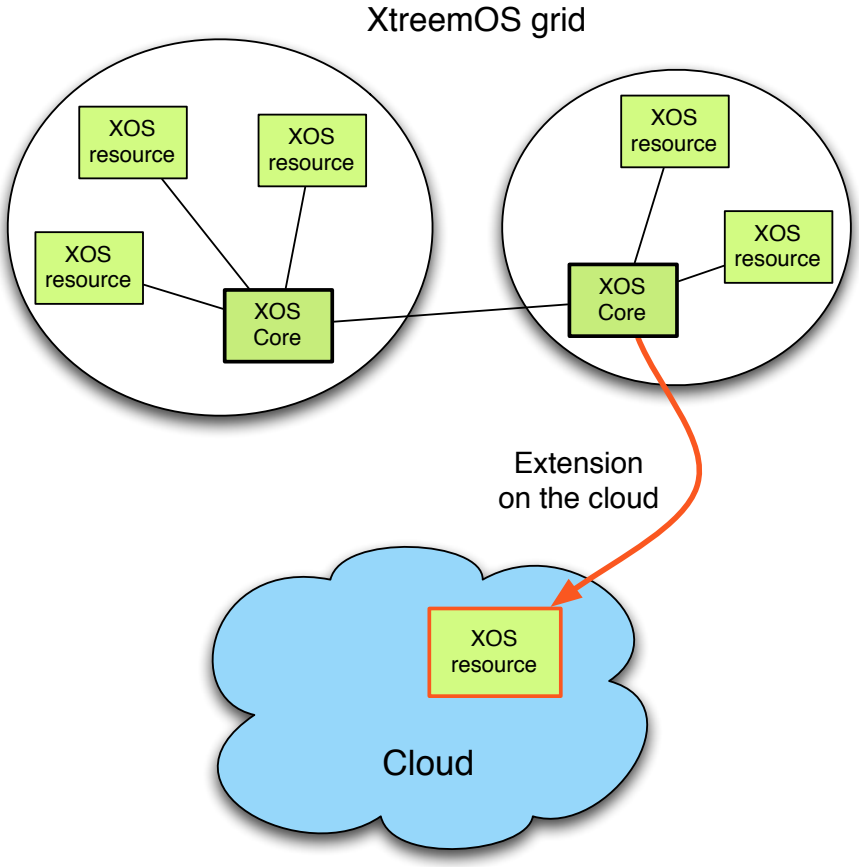


Figure 3: Extension of an XtreemOS grid (composed of two sites) on a cloud

4 Implementation with a Nimbus Cloud

In this section, we describe the design and implementation of our prototype system. This prototype provides a one-click deployment of fully-configured XtreemOS resources in a Nimbus cloud, which can be used for executing jobs immediately after their automatic configuration has finished.

This automatic deployment system consists in a set of scripts running in two different locations. On an XtreemOS core node, scripts are used to initiate VM deployment and confirm the resource when it applies for a certificate. On the newly deployed XtreemOS resource VM, a script is run at boot to automatically configure the system and register as a new resource with a core node.

We now present the implementation of our system, which is described in Figure 4.

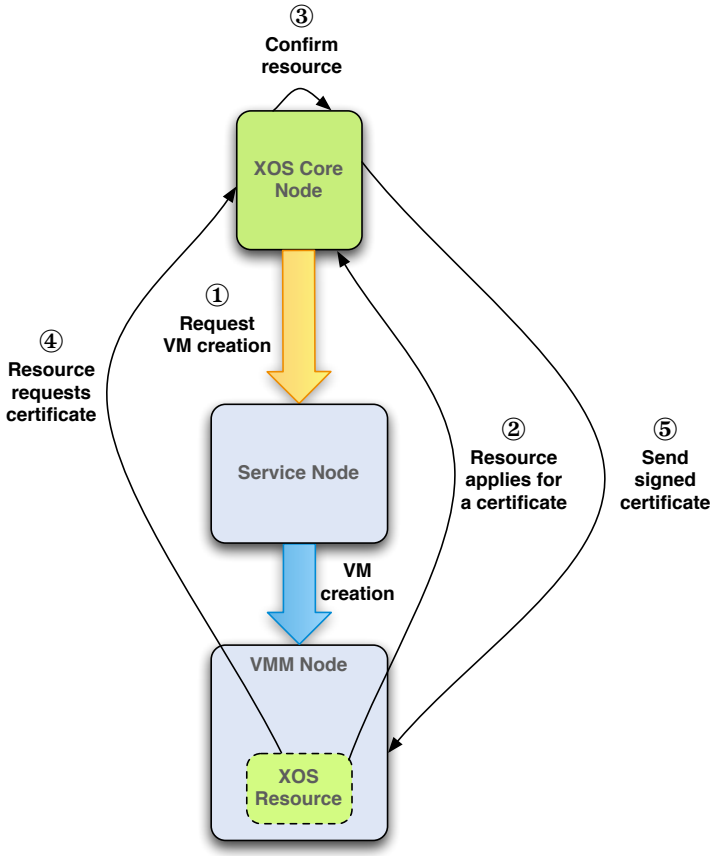


Figure 4: Architecture of our automatic deployment system

4.1 Creation of new XtreemOS resources in Nimbus

In order to create virtual machines on a Nimbus cloud, we use the Nimbus cloud client. Our script calls the cloud client with the requested reservation

time (specified in hours) and the name of the VM image to deploy. This image contains a fresh installation of an XtreemOS resource node, stored in the Nimbus storage service. The cloud client blocks until the VM is created and outputs information about the VM such as the IP that was assigned.

The cloud client relies on the WSRF frontend. Since this interface is specific to Nimbus and we do not make use of its advanced features, we plan to update our prototype to use the EC2 protocol in the future.

After the VM is created, the core node executes another script which waits for the XtreemOS resource to apply for registration in the grid.

4.2 Addition of new resources to our XtreemOS grid

Once a VM has been created, the boot process executes the XtreemOS automatic configuration tool. This tool, developed by the XtreemOS consortium, is called `xosautoconfig`. It uses three types of files: configuration scripts, variables and services definition files, and template directories. The main components of this tool are:

- `xosautoconfig`: main configuration script. This script configures the resource and applies for registration in the grid. After execution, the resource still need to be confirmed on the core node before joining the grid.
- `finishConfig`: once a resource registration has been confirmed, this script is run on the resource node to receive and install a resource certificate.
- localDefs: local variables definitions (VO name, IP and hostname of the machine)
- globalDefs: global variables definitions (IP and hostname of the core node)

Once the `xosautoconfig` script has been executed, the resource has applied for registration in the grid. The script running on the core node validates the pending registration using the `confirmResource` tool. Finally, the resource obtains the certificate using the `finishConfig` script. After this script has been executed successfully, the resource is fully integrated in the grid and is able to execute jobs.

4.2.1 Security issues

As explained above, XtreemOS resource nodes have to be authenticated before joining the grid. 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 the current implementation of the RCA server in XtreemOS, resources are uniquely identified by their IP and port.

In order to securely register a virtual XtreemOS resource, we deploy VMs from an XtreemOS core node running the RCA server. This allows the core node to learn the IP allocated to the VM. The core node then waits until it

receives a registration request from this IP. This allows to confirm registration of only known resources.

However, in a cloud environment, the pool of available IPs is shared between all users (i.e. user A can receive an IP previously allocated to user B). Considering this problem, we have to make sure that IPs are unregistered before they can be reallocated to potentially malicious users.

We propose that our system unregisters resources when the job they were executing is finished. Also, we should regularly check the status of our cloud resources and unregister those which have been terminated before the end of a job. For instance, this could happen if spot instances are used for job execution (a spot instance can be terminated at any time).

4.3 Usage of our new XtreemOS resources

Once a VM resource has been registered in the grid, jobs submitted in the grid can be scheduled on this new node. Users who submitted jobs provide input and get output results through the XtreemFS file system. The XtreemFS services run on core nodes which remains accessible after the resources have been terminated.

5 Validation Testing

We built a test setup to replicate a typical cloud environment, where the cloud has a dedicated internal network and offers access to the outside (the Internet) through a frontend. In our setup, the laboratory network replicates the Internet, and we configured a machine to act as a frontend to our internal cloud network.

Figure 5 presents our network setup. The laboratory network (10.0.0.0/8) contains the XOS core node (10.0.0.2) and a gateway to our cloud network (10.0.0.1). Our Nimbus cloud uses a separate LAN (192.168.0.0/24). This network contains the gateway (192.168.0.254), the service node (192.168.0.253), a VMM (192.168.0.252) and VMs (192.168.0.X). The gateway is configured to route traffic between both network, in order to allow the XOS core node to communicate with the Nimbus service node and the XOS resource VMs.

Our validation testing used version 2.0 of XtreemOS. We used this setup to validate our prototype. After deploying new XtreemOS resources as Nimbus VMs through our system, we verified that jobs were scheduled to these resources.

6 Conclusions and Future Work

We believe that cloud computing can bring more flexibility in resource management to grid infrastructures to improve user and administrator experience. In this report, we present the design and implementation of a prototype system allowing the XtreemOS grid operating system to be dynamically extended with resources from a Nimbus cloud. Our system uses automatic deployment in order to transparently create new resources. We validated our system by verifying that jobs were scheduled on XtreemOS resource VMs which were deployed on a Nimbus cloud and added into an existing XtreemOS grid automatically by our prototype.

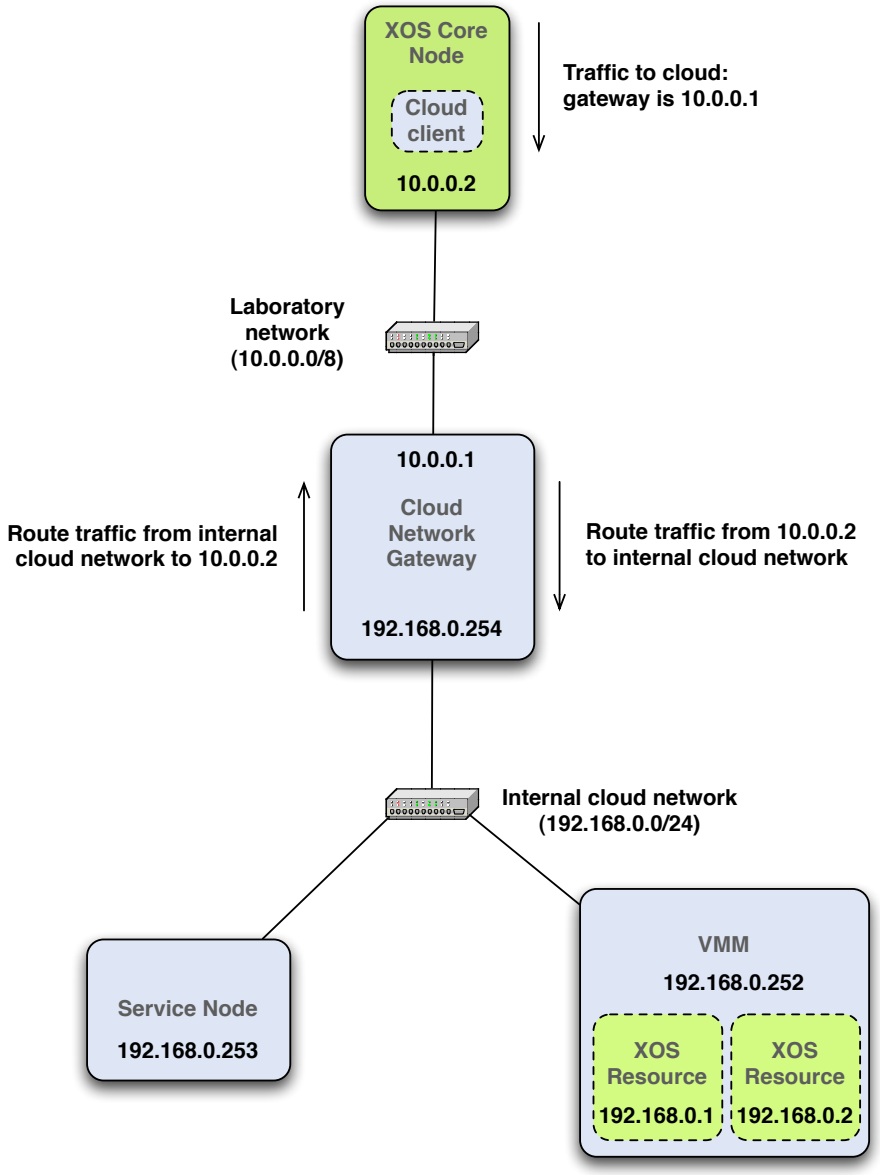


Figure 5: Architecture of our network setup: our Nimbus cloud is using a dedicated network

In future work, we plan to test our system with a larger number of nodes and evaluate its scalability. Then, we will extend the XtreemOS resource manager to automatically trigger VM creation when a job is submitted and no more grid resource is available. Finally, we will design policies in order to select when and where cloud resources should be deployed. We envisage testing this system with Amazon EC2 resources, and also plan to add support for other cloud interfaces to take advantage of multiple clouds.

References

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

[2] FlexiScale: Cloud Computing on demand. http://www.flexiscale.com/.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[4] Mario Leandro Bertogna, Eduardo Grosclaude, Marcelo Naiouf, Armando Giusti, and Emilio Luque. Dynamic on demand virtual clusters in grid. In *Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 2008)*, pages 13–22, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual Clusters for Grid Communities. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 513–520, Washington, DC, USA, 2006. IEEE Computer Society.

[6] Ian T. Foster. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 1–4, London, UK, 2001. Springer-Verlag.

[7] Brian Hayes. Cloud Computing. *Communications of the ACM*, 51(7):9–11, 2008.

[8] Shantenu Jha, Andre Merzky, and Geoffrey Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and Computation: Practice & Experience*, 21(8):1087–1108, 2009.

[9] Hartmut Kaiser, Andre Merzky, Stephan Hirmer, and Gabrielle Allen. The SAGA C++ Reference Implementation: Lessons Learnt from Juggling with Seemingly Contradictory Goals. In *Second International Workshop on Library-Centric Software Design (LCSD'06)*, 2006.

[10] Kate Keahey and Tim Freeman. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *Cloud Computing and Its Applications 2008 (CCA-08)*, 2008.

[11] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the Linux Virtual Machine Monitor. In *Proceedings of the 2007 Linux Symposium*, volume 1, pages 225–230, June 2007.

[12] I. Llorente, R. Moreno-Vozmediano, and R. Montero. Cloud Computing for on-Demand Grid Resource Provisioning. *Advances in Parallel Computing*, 18:177–191, 2009.

[13] Emilio P. Mancini, Massimiliano Rak, and Umberto Villano. PerfCloud: GRID Services for Performance-Oriented Development of Cloud Computing Applications. In *IEEE International Workshops on Enabling Technologies*, pages 201–206, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[14] Andre Merzky, Katerina Stamou, and Shantenu Jha. Application Level Interoperability between Clouds and Grids. In *Workshops at the Grid and Pervasive Computing Conference*, pages 143–150, 2009.

[15] Christine Morin. XtreemOS: A Grid Operating System Making your Computer Ready for Participating in Virtual Organizations. In *ISORC '07: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 393–402, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Christine Morin, Yvon Jégou, Jérôme Gallard, and Pierre Riteau. Clouds: a New Playground for the XtreemOS Grid Operating System. *Parallel Processing Letters (PPL)*, 19(3):435–449, September 2009.

[17] Manuel Rodríguez, Daniel Tapiador, Javier Fontán, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. Dynamic Provisioning of Virtual Clusters for Grid Computing. In *Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 2008)*, pages 23–32, Berlin, Heidelberg, 2008. Springer-Verlag.

[18] Aaron Weiss. Computing in the Clouds. *netWorker*, 11(4):16–25, 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