Supporting e-Science Applications through the On-Demand Execution of Opportunistic and Customizable Virtual Clusters

Harold Castro, Mario Villamizar, and Eduardo Rosales

Abstract**—** This paper deals with the design and implementation of a virtual opportunistic grid infrastructure that allows taking advantage of the idle processing capabilities currently available in the computer labs of a university campus, ensuring local users to have priority in accessing the computational resources, while simultaneously, a virtual cluster takes the resources unused by them. A virtualization strategy is proposed to allow the deployment of opportunistic virtual clusters which integration provides a scalable grid solution capable of supplying the high performance computing (HPC) needs required for the development of e-Science projects. The proposed solution was implemented and tested through the execution of opportunistic virtual clusters with customized application environments for projects of different scientific disciplines, evidencing high efficiency in result generation.

Keywords**—** eScience, desktop grid, grid computing, unagrid.

I. Introduction

GRID computing surged as a vanguard technology for supporting the development of different scientific projects at a global scale [1]. Grid infrastructures may be classified as Service Grids, or Desktop Grids and Volunteer Computing Systems (DGVCSs). The first have been developed to meet the needs of applications and specific projects within large-scale environments designed so that a set of organizations can share a certain amount of dedicated and federated resources through the use of different standards and middleware grids (Torque [2], PBS [3], Sun Grid Engine (SGE) [4], LoadLeveler [5], Globus [6], gLite [7]). These infrastructures provide large computing capabilities; nonetheless, their implementation requires large financial investments due to the high costs of, not only hardware, but also those associated with physical space, temperature-controlled environment, installation processes, management, configuration, and maintenance, making this option unviable in organizations with low financial resources.

Also known as volunteer computing [8], public resources computing [9], or opportunistic grids [10], DGVCSs have emerged as an alternative for obtaining computational resources at low cost, focused on taking advantage of the capabilities of existing commodity computing resources. These infrastructures are based on the benefits of conventional desktop computers, as those daily used by employees or university students, and allow adding computational capabilities of thousands of computers, enabling the development of e-Science projects that require the execution of intensive processing, memory and/or storage applications. These computers are available through Internet or Intranet environments, have partial availability, are highly heterogeneous, and are part of independent administrative domains. DGVCs seek to maximize the efficient use of partially available computing resources; this includes the non-exclusive use of computing resources, while ensuring that interactive users of those shared resources do not perceive any deterioration in the quality of service. Such strategies are intended to provide a computing infrastructure at a large scale, without incurring into additional investments for the purchase and maintenance of hardware, physical space and controlled temperature environments of the traditional vertical growth dedicated infrastructures. These features have allowed the deployment of Internet scalable computing infrastructure, composed mainly by economic, heterogeneous, distributed and partially available computers whose added processing power has become in the order of the PetaFLOPS (Floating point Operations per Second) [11].

Taking into account the benefits of DGVCSs, we present a novel infrastructure known as UnaGrid[1], which provides the processing capabilities required by the applications of different research areas at a university. The infrastructure takes in an opportunistic way the processing capabilities unused by the end users at computer labs, avoiding the purchase of dedicated resources. The infrastructure proposed has been initially deployed on the campus of the Universidad de los Andes and, in this work, the design and details of the implementation deployed are presented along with the results obtained of the first applications executed. The proposed infrastructure,

Manuscript received May 16, 2011: Revised version received August 10, 2011. This work was supported in part by the EELA-2 and GISELA projects.

H. Castro is with the Systems and Computing Engineering Department, Universidad de los Andes, Colombia, Carrera 1 Este No 19A-40, Bogotá (e-mail: hcastro@uniandes.edu.co).

M. Villamizar is with the Systems and Computing Engineering Department, Universidad de los Andes, Colombia, Carrera 1 Este No 19A-40, Bogotá (phone: +571 332 43 24; fax: +571 332 43 25; e-mail: mj.villamizar24@uniandes.edu.co).

E. Rosales is with the Universidad de los Andes, Bogotá, Colombia and with the CMS (Compact Muon Solenoid) experiment at CERN, Geneva, Switzerland (e-mail: ee.rosales24@uniandes.edu.co).

[1] Project partially funded by ECOS-NORD action C06M02

through virtualization tools, allows the deployment of customized virtual clusters that satisfy the requirements of different research areas for the execution of specific applications by reusing physical resources on demand.

The paper is organized as follows: section 2 presents the related works to virtual and opportunistic grid infrastructures. The description of the design for the proposed infrastructure is discussed in Section 3. Section 4 details the implementation realized on the university campus and the results obtained from the first applications executed within the infrastructure. Section 5 concludes.

II. Related Work

An introduction to virtualization technologies paradigm to taking advantage of heterogeneous computing resources are shown in [12]; in [13] an exhaustive comparison and assessment of three virtualization technologies for HPC are put forth. Opportunistic models for selecting the resources used in the deployment of virtual clusters and grids are described in [14]. In [15] solutions for the deployment of virtual cluster and grid infrastructures are described.

Referring to taking advantage of idle computational resources, Worm [16] and Condor [17] were the first projects developed to take advantage of the idle processing capabilities available in LAN networks. The Worm project was proposed by Xerox Palo Alto Research Center (PARC) and was intended to develop Worm applications, which were executed, replicated and migrated in idle machines connected over a LAN during night hours, when most computational resources arranged in Xerox Palo Alto could be considered idle. Idle machine detection was implemented through a simple protocol that included the broadcast of a special package, making the target machines announcing their current status (idle or busy). Idle machines then received a request to boot from the network, loading their assigned Worm segment. A Worm program had several segments, each running on a different machine. The worm segments were able to communicate among themselves, so if a segment fails, they were able to locate another idle machine and run a backup segment there.

The Condor project, begun by the University of Wisconsin-Madison, has been focused in the development of a specialized load management system for different types of intensive computing tasks in LAN networks. This High Throughput Computing (HTC) project still remains in force and is aimed at the development, implementation, deployment, and evaluation mechanisms and policies that support HTC in large sets of distributed computers including the efficient use of idle computing resources. Like other batch queuing systems, Condor provides a mechanism to manage a work queue, policy planning, priority schemes, monitoring and resource management. This way, users can send to Condor an individual job or a set of jobs to be scheduled onto the available resources. Condor is responsible for placing the jobs in the work queue, choosing the most suitable resource to run a job based on a planning policy, resource matching, executing jobs, monitoring their progress and finally informing the user about their completion. The Condor architecture is based on the master/slave model, where the master component manages the execution of jobs onto a group of slave nodes.

For taking advantage of more idle computing resources, projects like GIMPS [18] (Great Internet Mersenne Prime Search) and SETI@home [9] have used distributed resources through Internet for solving a specific problem. GIMPS is considered to be the first volunteer computing project in the world, i.e. the first resource donation project at Internet scale. The GIMPS project is a distributed computing project dedicated to the search of Mersenne prime numbers, developed by Mersenne Research, Inc. One of the major contributions of GIMPS to the DGVCSs is the use of idle computing resources through the download and installation of a thin client that allows people to donate idle computing resources for the calculation of Mersenne primes on Windows, Linux, and Mac platforms. This client is developed in the C programming language and runs as a background process in the lowest available priority in the host operating system.

The SETI@home [9] project represented the following success in allowing the scalable participation of millions of computational resources with the objective of resolving a unique problem: SETI (Search for Extraterrestrial Intelligence). The project began at the University of California, Berkeley and was credited as the project receiving the greatest computational processing time in history (University of California). SETI@home focuses on shortwave radio signals processed from space. A radio signal analysis process needs an enormous amount of computing capabilities to cover a broad spectrum with great sensitivity. In addition, signal analysis can be parallelized and does not require communication between clients, by which SETI@home efficiently uses the public resource computational model on the Internet. Like GIMPS, SETI@home is based on a lightweight agent developed in the C++ programming language and supports nearly all existing operating systems. SETI@home may run as a screensaver which includes statistical information associated with the opportunistic processing. One of the major contributions of SETI@home to DGVCSs was the organization of a strategy-award; rewarding the contribution of distinguished participants. This viral marketing strategy is a categorization of participants by the amount of computational processing contributed to SETI@home, allowing multiple users grouping to enable the competition. This categorization became ranked worldwide, and can be found on the SETI@home official website. It is backed by a set of incentives that includes personalized acknowledgement emails and public recognition on the project's official website.

Others DGVCSs projects have been developed for supporting general-purpose distributed computing project at global scale, Internet. Distributed.Net [19] was the first general-purpose distributed computing project founded by a non-profit organization under the GNU FPL (Freeware Public

License). Its greatest contribution to the DGVCSs was proposing the implementation of a general purpose distributed computing system in the world. The project has been oriented to break encryption algorithms and to search for Optimal Golomb Rulers (OGR)[2], that are especially useful for encoding and combinatorial theories, as well as for the sensor placement for x-ray crystallography and for the study of radio astronomy techniques. Both tasks are characterized by the intensive use of huge processing capabilities and by their natural distribution into non-dependent work units to generate results. Distributed.net is also based on the opportunistic execution of a thin client that was developed in the C++ programming. The system's architecture is based on a three tier client/server model (pyramid architecture) that allows the system to be highly scalable at the Internet level. The Distributed.net client communicates directly with a proxy server which is responsible for assigning the work units obtained from a centralized server. Once a client has processed a work unit, it delivers its results to its proxy server, which in turn sends them to the main centralized server. The architecture provides a basic fault tolerance mechanism to allow customers to use round-robin DNS to locate proxy servers, whenever the server originally assigned is no longer available.

BOINC (Berkeley Open Infrastructure for Network Computing) [20] represents an evolvement of SETI@home since this proposes a system for creating and operating public-resource computing projects. BOINC aims to use computing resources for the development of multipurpose scientific projects, hiding the complexities associated with the creation, operation and maintenance of public-resource computing projects by providing a set of tools for building a secure infrastructure with autonomous domain administrative servers and high scalability on the Internet. BOINC architecture is based on a client/server model, which gives clients the responsibility for requesting work units for scientific applications and for delivering results to the principal server. BOINC supports the execution of parallel applications allowing a slight communication and synchronization between clients; implements basic fault tolerance mechanisms through checkpoints while offering the possibility of integrating multi-purpose projects among which some related to Grid Computing. BOINC depends on a robust central infrastructure capable of managing the multiple operations carried out by clients and lacks general support for parallel applications. The project includes all viral marketing strategies originally implemented in SETI@home, but extends them in a layout for the participation of multiple projects. Additionally, it implements a better security mechanism to protect files containing credits, granted to users for their voluntary contribution of computing resources. BOINC has led the scalable DGVCSs Internet multipurpose approach, allowing the development of multidisciplinary projects, including research on climate prediction, astronomy and high energy physics as well as grid computing projects.

New DGVCS projects have been specialized in grid computing projects of variable scalability which require the deployment of grid middleware for work unit processing that requires large computational capacities, particularly large processing capabilities beyond those offered by a single administrative domain [21] [22]. Bayanihan Computing .NET [23] is a generic framework for grid computing based on Microsoft .NET. Bayanihan implements volunteer computing by providing a PoolService Web service associated with computers that act as clients and providers (volunteers) of computing resources. The main Web service allows computation clients to create sets of tasks that are sent to volunteers for its execution and subsequent return of results. The files are downloaded by volunteers, implementing basic security mechanisms that are provided by the Microsoft .NET platform. Bayanihan allows the execution of general purpose applications. One of the major contributions of Bayanihan is the use of Web services as a platform, representing an alternative to the middleware used by all of its predecessor projects; however, Bayanihan has a coupled architecture which limits its scalability to Internet environments.

The Condor project also has been focused in the development of a framework to share and lever computing resources among different administrative domains, which has been called Condor-G [24]. Condor-G allows taking full advantage of Condor characteristics, particularly those related to the use of idle resources in an administrative domain, to the availability of tools and mechanisms for resource management and discovery, as well as to the security measures in multi-domain environments provided by Globus, the standard grid middleware. Condor-G combines Globus Toolkit's multi-domain resource management protocols and Condor´s intra-domain resource management, allowing users to take advantage of idle computing resources from different administrative domains as if all of these belong to a single domain. Condor-G can handle thousands of works to be executed on multiple distributed sites, providing features such as monitoring and task management, resource selection, notices, policies, security credentials management, fault tolerance and management of complex dependencies between tasks. Condor-G can be used directly by end users from high level interfaces (brokers) or Web portals. Condor-G implements a new mechanism called GlideIn, with which it is possible to execute jobs by starting a Condor daemon on a remote computer without requiring Condor installation binaries to be in such computers. This allows remote computers to be part of an existing Condor pool because the Condor-G scheduler is informed of these resources' existence.

InteGrade [25] is a GNU LGPL (Lesser General Public License) grid middleware infrastructure, based on the opportunistic use of idle computing resources. The main contribution of InteGrade to the DGVCSs is the implementation of a computing-resource usage-pattern

[2] In mathematics, the term "Golomb Ruler" refers to a set of non-negative integers such that no two distinct pairs of numbers from the set have the same difference.

analysis-component, capable of collecting statistical data and of probabilistically determining the availability of a machine [26], as well as the convenience of assigning a specific job to such a machine. This execution evolves in time, because of the permanent data collection, able to determine new prevailing patterns. Based on this component, InteGrade supports sequential, parametric (Parameter Sweep Applications) and parallel (Bulk Synchronous Parallel) applications with communication and synchronization requirements between nodes.

OurGrid [10] in an Open Source resource sharing system based on a P2P network that makes it easy to share resources equitably to form grid computing infrastructures. In OurGrid each peer represents a whole site in a different administrative domain. OurGrid offers a complete solution for the construction of opportunistic grids without human intervention for new administrative domains to make part of a grid infrastructure used for running bag-of-tasks (BoT) applications [27] [28]. The process of sharing resources is based on the principle of donating idle computing cycles in order to have access to a greater amount of computing power provided in cooperation with other participants of the grid. The OurGrid community participants should consider two fundamental assumptions; the first of these is the effective contribution of computing resources to the system by at least two peers (so one can always get resources from a different provider within the community). The second relates to the lack of QoS guarantees offered to applications deployed on OurGrid. This latter characteristic reduces the complexities associated with traditional grid economy models [29], [30], prevents negotiations between resource providers and promotes a culture of equitable resources sharing, following a best-effort strategy. The main contribution of OurGrid was the incorporation of the concept of Network of Favors [31], a partial solution to the problem of non-reciprocal participants in resources sharing system (free-riding peers) [32], to do this, each OurGrid Peer use mechanisms for locally saving information about participants that have made previous donations, allowing prioritizing requests from participants with greater credit history data. OurGrid uses an approach based on virtual machines for solving potential security problems, so it restricts an execution environment with limited hardware and software resources of the physical machine (those assigned by a type II hypervisor [33]). OurGrid also provides robust security mechanisms based on private and public keys to certify the authenticity of messages sent using OurGrid protocols. These mechanisms are seeking to prevent denial of service attacks caused by malicious participants.

Finally, the LHC (Large Hadron Collider) project proposed the use of some DGVCSs to tackle the huge computing requirements from the different experiments (ATLAS, CMS ALICE, LHCb, TOTEM y LHCf) currently in execution. Initially the LHC developed a solution for using the underutilized computers of the CERN infrastructure; this solution is called Compact Physics Screen Saver (CPSS) [34]. The CPSS system uses a modular lightweight screensaver installed in the desktop CERN computers for taking advantage of their computing capabilities, in a non-intrusive manner. CPSS uses a centralized Web application server which contains the whole tasks to be executed and the applications used by the clients. CPSS have been deployed in hundreds of computers; however its scalability is limited to the number of desktop computers available at CERN. To take advantage of more computing capabilities, the LHC began to use the BOINC platform in a project called LHC@home [35]. This project uses the processing capabilities of millions of computers distributed through Internet provided by the BOINC project. These capabilities have been used for running different types of application with low data transfer. In the LHC@home several approaches were developed for integrating opportunistic and dedicated infrastructures.

In CPSS and LHC@home projects, each application used by CERN must be ported, in the case of CPSS project to the operating system where the screensaver is executed and in the case of LHC@home to the BOINC platform, this involves major efforts and it is not a scalable solution. A new solution, called CernVM [36] using virtualization technologies to solve the problem of portability of applications was developed. In CernVM, each volunteer downloads a virtual machine image which has a CERN customizable environment with the required application. The use of virtual machines allows that the same application to be executed on heterogeneous operating systems and hardware. Two new problems arose due to the use of virtual machines, the first associated with the size of the virtual machines (between 8-10 Gigabytes), and the second with the virtual machine and application updates. For solving these problems, a new and innovative solution was developed to manage virtual machines and applications. The volunteers initially download a lightweight virtual machine (of around 100 MB), called thin appliance, whose size is incremented the first time that is configured (until 1 o 2 Gigabytes). The virtual machines do not contain the LHC software or grid middleware; on the contrary, the software and its dependencies are accessed from one or several servers. CernVM has published several virtual machine images that can be downloaded by volunteer users with different hypervisors (VMware, Virtual Box, KVM (Kernel-based Virtual Machine), Microsoft Hyper-V, Parallels [37], Xen y Amazon EC2) and architectures (32 y 64 bits).

Taking into account the benefits of DGVCs for taking advantage of idle processing capabilities and the DGVCS projects that have already been developed, in this paper we present a novel infrastructure that allows the creation and execution of customized execution environments that do not impose any restriction on the grid or cluster middleware (described above), applications, configurations, or parallel programming technology to be used. UnaGrid also uses virtualization technologies that allow users, with basic IT knowledge, to autonomously deploy customized virtual clusters on the same physical infrastructure that is shared through an opportunistic approach. The virtual infrastructure proposed seeks to efficiently take advantage of the idle processing capabilities available in computer labs on a university campus to forming a scalable opportunistic virtual infrastructure, totally available for the development of e-Science projects with different middleware, application and configurations requirements.

III. UnaGrid Architecture

The main requirement of the UnaGrid architecture is to flexibly take advantage of the idle processing capabilities available in computer labs on a university campus. Thus, making the development of projects within different research areas possible, through the deployment of an opportunistic infrastructure that allows the execution of applications with high processing and customized execution environments on demand. In order to achieve this objective, the opportunistic infrastructure must allow the execution of a wide variety of applications with specific requirements (versions, dependencies, middleware, etc.) due to relevance of facilitating the execution of applications within its native environments, guaranteeing high usability for end users. What we call Customized Virtual Cluster (CVC) is introduced within this context as a set of commodity and interconnected desktop workstations executing coordinated virtual machines through virtualization technologies. All of these virtual machines on execution make up a processing cluster.

Virtual machines take advantage of the unused physical resources by end users while they carry out their daily activities without perceiving any loss in the quality of the service. This is achieved by executing the virtual machines as low-priority background processes, guaranteeing that the end user has available all of the computational resources (if such is required), while the virtual machine only consumes the resources that the end user is not using (or all of these in the case of unused computers). Each CVC can be deployed on hundreds or thousands of commodity desktop computers in which a virtual machine image is stored and roles as a CVC slave; using a single dedicated machine for each CVC to support the master role.

A distributed file system such as NFS or AFS for handling the data of a CVC may be used. Regarding the integration of dedicated processing resources to the opportunistic infrastructure, all what is required is the installation of a dedicated cluster or grid which communication infrastructure and configurations must be interconnected with the virtual infrastructure.

Taking advantage of the facilities provided by the virtualization technologies allows each research group to establish and configure CVCs on virtual machines and deploy them on different computer labs. The deployment of a CVC requires that the virtual machines have a specific configuration regarding applications to be used by the research group and cluster or grid middleware such as Condor, Sun Grid Engine, PBS, Globus, gLite, etc. The idea is to recreate as exact as possible the environment a researcher is used to have in a real cluster (something similar to what it is achieved in Grid5000 [38] or cloud computing approaches like Open Nebula [39], but in dedicated machines). Our strategy facilitates the different research groups to deploy several CVCs by using the same physical infrastructure. This is achieved by storing an image of each one of the possible CVCs that may be executed on each physical computer, thus facilitating the reutilization of resources and guaranteeing better quality of service to CVC users. The solution is focused on the exclusive execution of a virtual machine for each physical computer with the objective of avoiding resources competing among different virtual machines.

Our strategy solves the problems associated with the lack or sub-utilization of preexisting computer labs thus promoting the sharing of resources and the collaborative work among research groups. Nonetheless, the need exists to schedule and control the CVC deployment on demand. To solve this problem, a Web Portal called ¨GUMA – Grid Uniandes Management Application¨ was developed, which allows end users of different research groups to deploy on demand CVCs on physical infrastructures shared, for specific periods, allowing the virtual machines to execute as long as they are being used. Administration tools are also provided in this portal for analyzing and visualizing the status of the physical and virtual resources of the computer labs.

Some applications require non-conventional processing capabilities, which may exceed the capabilities, offered by an individual CVC, to solve this, the CVC addition becomes necessary, which process is viable through three alternatives: the first alternative consists in the configuration of CVC images within a computer laboratory to make up part of another existing CVC; the second alternative is focused on taking advantage of the aggregate capabilities of computer resources offered by some schedulers such as Condor and SGE; the third alternative is focused on installing a grid middleware on master nodes, which allows the execution of applications on different CVCs or a mix of a CVC and a dedicated infrastructure.

The first two alternatives are focused on taking advantage of the capabilities offered by the schedulers; the first requires less effort for its deployment. However, these alternatives require the installation of the same scheduler in all CVCs, which limits the flexibility of the solution. In order to solve this, the third alternative (which requires more effort in configuring and administrative agreements) allows the addition of the CVC processing capabilities with different schedulers. For instance, Globus allows this functionality and additionally allows the use of other tools and protocols that are very useful for creating grid solutions such as: management mechanisms for security, management, transference, replication, data management, among others. This type of solution, for sharing processing resources among different CVCs, requires the implementation of different security mechanisms. The infrastructure proposed makes use of the Globus Toolkit 4.2

Security Infrastructure. The installation of a certificate authority (CA) on the university campus became necessary for the interaction and utilization of clusters (CVCs or dedicated) between the different research groups.

In order to use the CVCs, the users may send jobs to the queue scheduler and/or to the grid middleware installed on the CVC master machine. With regard to solutions involving Globus, this middleware facilitates users to submit jobs to different clusters because they only need to learn the Globus syntax, which allows them to use the different clusters in a transparent manner. The proposed scheme is illustrated in Fig. 1.

The opportunistic infrastructure proposed allows taking advantage of the idle power processing in a 24x7 scheme and, additionally, can be deployed on desktops that have Windows, Linux and Mac as their operating systems, since the entire deployment is based on the use of virtual machines.

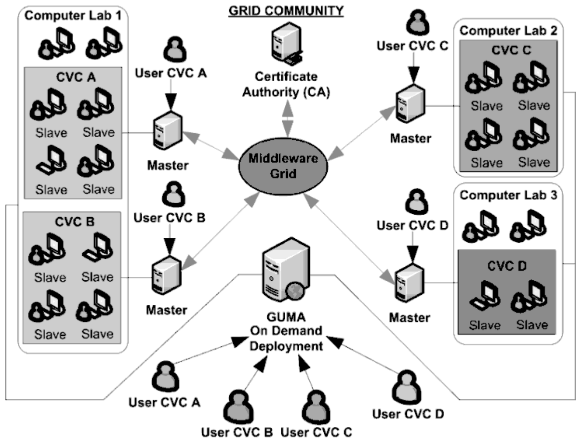


Fig. 1 architecture of the proposed opportunistic virtual grid infrastructure

IV. Implementation

The proposed infrastructure has been deployed in three computer labs (Wuaira 1, Wuaira II and Turing) at the Universidad de los Andes; those computers have Windows XP as their base operating system. This deployment has allowed providing processing capabilities to projects within different research areas (see Section 4.2). Each laboratory has 35 computers which have an Intel Core 2 Duo (1.86GHz) processor and 4GB of RAM memory. For the network interconnection, all CVC slave nodes are in a Gigabit Ethernet LAN network and the master nodes run on a dedicated server outside the computer labs (for stability reasons). For communication between the master and the slave nodes, a multilayer switch of Gigabit Ethernet links is used. Four different CVCs have been deployed on this physical infrastructure. To support the demand for superior processing capabilities to those provided by an individual computer lab, some CVCs master nodes, located on a VMware ESX dedicated server, have installed the Globus Toolkit 4.2 middleware.

The VMware virtualization software has been used for the deployment of the virtual infrastructure. Virtual machines playing the slave role of the virtual clusters have assigned both cores and 1GB of RAM memory, while the master node has assigned two cores and 2GB of RAM memory. Another virtual machine was additionally configured and it operates as the CA for some CVCs. This CA generates the digital certificates for master nodes and CVC users. NFS is used for data management within each CVC which server is located in a NFS-NAS server solution; this provides sufficient storage capacity for storing the files shared with all the slave nodes.

The process for submitting jobs to the opportunistic infrastructure consists in the entry of users to the CVC master machine through SSH (some application deployed provide Graphical User Interfaces GUIs for sending jobs to UnaGrid infrastructure), the jobs may be sent by the users to the cluster scheduler or Globus. In the first case, the jobs are executed with the machines available in the CVC, which the master node makes part, while in the second case the user may utilize the processing capabilities of the grid infrastructure built through the addition of several CVCs.

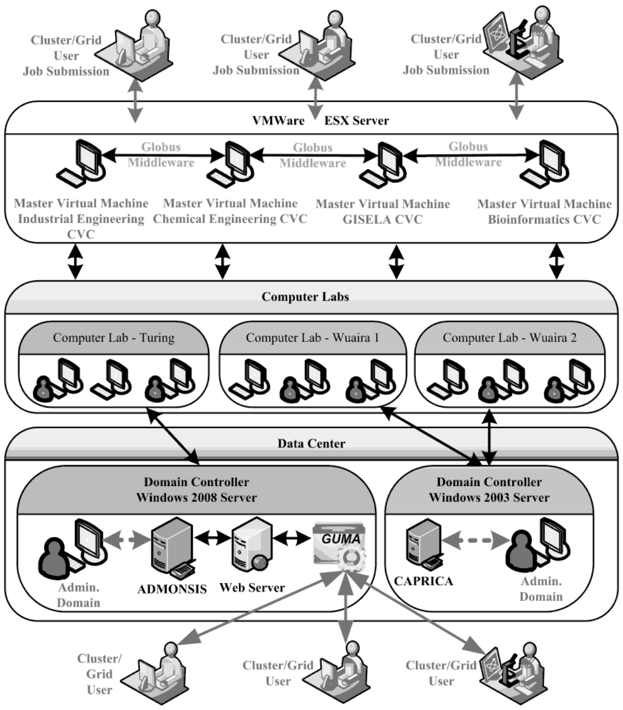
The deployment of CVCs is executed on demand through a Web portal known as GUMA, in which open technologies are used, such as Java Server Faces (JSF), Enterprise Java Beans (EJB), GlassFish, and MySQL. Regardless the administrative domain, this application executes and manages the CVCs. Multiple execution tests, supported by the active directory services of two domain controllers (Windows 2003 and 2008 Server) have evidenced high level of performance in the CVC execution, such as the launching of 35 virtual machines in less than 5 seconds and their afterward shutdown in less than 4 seconds. GUMA manages the remote execution of the instances of the virtual machines executor (VMware Workstation) in the computer resources available, allowing the CVC deployment on demand. This application utilizes the client-server scheme that, through authentication, authorization, and confidentiality mechanisms, provides multiple services for managing the grid from light clients, hiding the complexities associated to location, distribution, and heterogeneousness of the computer resources, facilitating the autonomy of the involved administrative domains and offering an intuitive graphic interface to end users. Administration services include the selection, startup, shutdown, and remote monitoring of the physical and virtual machines as well as query of all instances in execution by the different users. The scheme for the infrastructure deployed is illustrated in Fig. 2 and a GUMA GUI is shown in Fig. 3.

V. CVCs Deployed

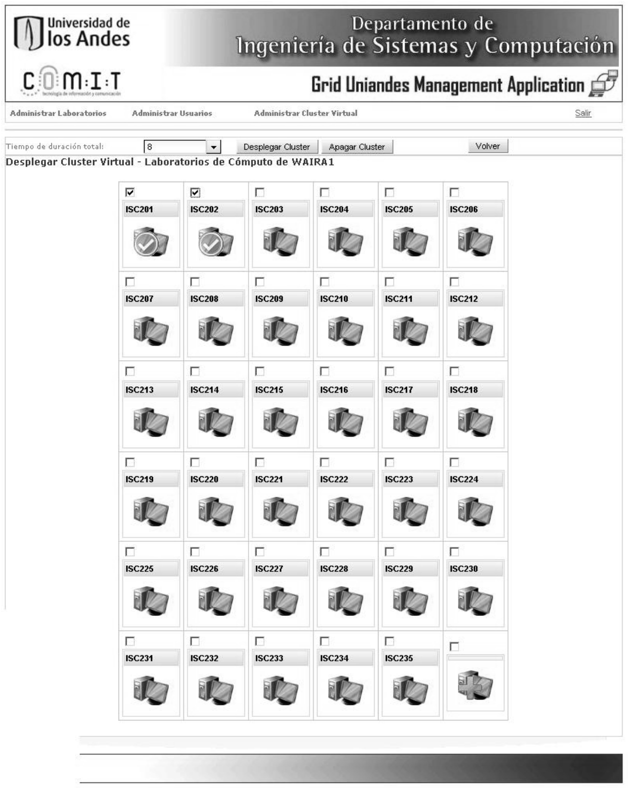
The infrastructure has been used in projects of different research areas such as chemical engineering, industrial engineering, bioinformatics and dedicated grid infrastructures; all applications executed are bag-of-tasks style.

A. *Chemical Engineering CVC*

In the chemical engineering area we developed the BSGrid application [40] to simulate the Bacillus thuringiensis bacterium. Bacillus thuringiensis (B. thuringiensis) is a gram positive spore forming bacterium widely known by its capacity to synthesize δ-endotoxins which are used as biopesticides representing 90% of the total biopesticide market in the world due to its high affinity, no harm to other species (including vertebrates) and low environmental impact. For instance, in Colombia they can be used for a typical problem in the insect control of maize crops. However, these biopesticides only participate in the 5% of the total pesticide market in the world; due to industrial-scale fermentation cannot obtain a high concentration of the δ-endotoxins, so the production of biopesticides has a high cost. Several computational methods and approaches have been developed to simulate the behavior of these bacteria, which allows analyze the behavior of this organisms along the time without involve the high cost associated with experimental tests; however they require large processing capabilities.


Cluster/Grid User Job Submission
Cluster/Grid User Job Submission
Cluster/Grid User Job Submission
VMWare ESX Server
Globus Middleware
Globus Middleware
Globus Middleware
Master Virtual Machine Industrial Engineering CVC
Master Virtual Machine Chemical Engineering CVC
Master Virtual Machine GISELA CVC
Master Virtual Machine Bioinformatics CVC
Computer Labs
Computer Lab - Turing
Computer Lab - Wuaira 1
Computer Lab - Wuaira 2
Data Center
Domain Controller Windows 2008 Server
Domain Controller Windows 2003 Server
GUMA
Admin. Domain
ADMONSIS
Web Server
CAPRICA
Admin. Domain
Cluster/ Grid User
Cluster/Grid User
Cluster/Grid User
Fig. 2 opportunistic virtual grid infrastructure deployed

To support large processing capabilities a CVC has been deployed in one computing room (Wuaira 1) and several BSGrid simulations have been executed on it for determinate the optimal conditions under which the B. thuringiensis δ-endotoxins are produced (the chemical results of the simulations are described in [41] and [42]). The CVC has configured the Condor scheduler version 6.7.1 and Debian 4.0 operating system. Each simulation is executed for a bacterium population which model is defined by a user through a GUI. Once defined the model, a complete simulation is executed as a batch process in the CVC, each bacterium of the population is executed as an independent job in a CVC virtual machine, and the simulation results are stored in a relational database. Once all the population has been executed, the users can query the database results through GUIs. The architecture of the infrastructure deployed for this project is shown in the Fig. 4.



Tiempo de duración total: 8

Desplegar Cluster Virtual - Laboratorios de Cómputo de WAIRA1

Fig. 3 GUMA graphical user interface

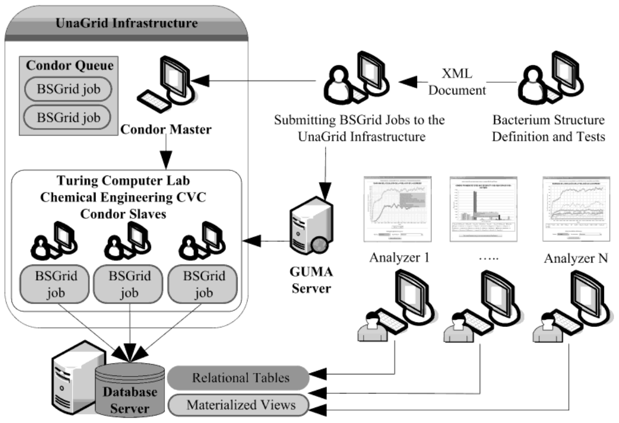


Fig. 4 chemical engineering CVC

B. *Bioinformatics CVC*

Bioinformatics research projects require the use of diverse heterogeneous types of applications and computational tools for different analyses, which demands technical expertise of commands and specific parameters of tens or even hundreds of applications, as well as the management commands of distributed infrastructures and the manual coordination for executing bioinformatics workflows. These tasks require each researcher has to spend a significant time and effort learning the technical utilization of all these computational tools. To facilitate the use applications, the transparent use of high performance computing infrastructures and the automatic management of workflows, we have been configured and adapted, the LONI Pipeline application, developed by the Laboratory of Neuro Imaging (LONI) of the University of California Los Angeles (UCLA) [43] [44]. LONI use a client-server model and a GUI that allows the easy creation, validation, execution and monitoring of workflows, allowing researchers to focus on the analysis of data generated by the workflows and not on technical issues of computer systems. Any suite of applications that can be executed through line command can be integrated to a LONI Pipeline installation; each application is integrated as a LONI module.

A CVC with several bioinformatics application, the Sun Grid Engine (SGE) 6.2u5 scheduler, and Debian 5.0 operating system has been configured. Two computer rooms (Wuaira 1 and Wuaira 2) and a dedicated cluster with 5 servers are used, in total 180 cores are available for processing, 140 from UnaGrid infrastructure and 40 from the dedicated cluster. The LONI Pipeline server and the SGE master have been installed in the same dedicated machine. From its installation, we have developed the LONI Pipeline modules for several bioinformatics application like HMMER, BLAST, Mr. Bayes and InterPro Scan, which are compute-intensive. A user authentication module were created and configured in the LONI server and now the bioinformatics researchers use the LONI Pipeline client as a tool for creating, implementing and managing workflows in bioinformatics projects, which are executed in the CVC. This application has been used in the development of several projects of genomic related to coffee and yucca, which seek genomic analysis to improve coffee, potato and yucca production affected by different biological organisms that decrease the production [45] [46]. The infrastructure deployed in the LONI Pipeline installation is shown in Fig. 5.

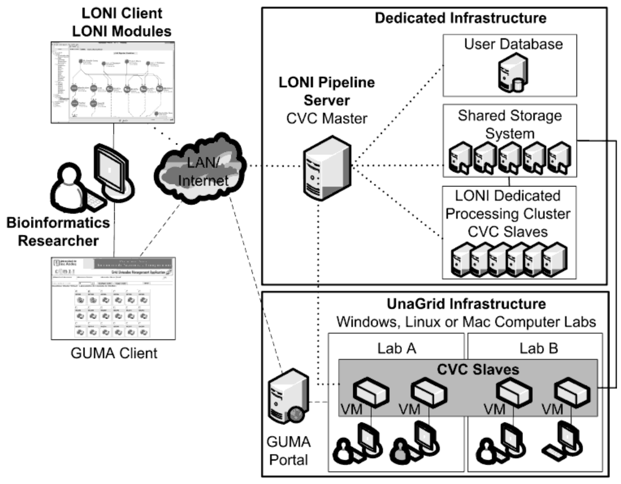


Fig. 5 bioinformatics CVC

C. Industrial Engineering CVC

In the industrial engineering area, we developed two different frameworks for solving optimization problems: JG2A [47] and pALS [48]. Java Grid-enabled Genetic Algorithm (JG2A) is a new generation of the Java Genetic Algorithm (JGA) framework for rapid prototyping of evolutionary algorithms that exploits parallelism in genetic algorithms in two ways: first, it allows the execution in parallel of a large set of instances (instances parallelization); and second, it provides parallelization of the population evaluation (population evaluation parallelization). The instances parallelization has been used in different parameter tuning experiments of vehicle routing and route design problems. The population evaluation parallelization is particularly useful for hard black-box optimization problems where the fitness function evaluation embeds a discrete-event or finite-element analysis simulation.

pALS acronym for parallel Adaptive Learning Search is a computational object oriented framework for the development of parallel and cooperative metaheuristics for solving complex optimization problems. The library exploits the parallelization allowing the deployment of mainly two models: the parallel execution of operators and the execution of separate instances or multi-start models. pALS also allows to include in the design of the problem's solution cooperation strategies such as the islands model for genetic algorithms or the parallel exploration of neighborhoods in metaheuristics derived from local searches, including a broad set of topologies associated with these models. pALS has been successfully used in different optimization problems and has proven to be a flexible, extensible and commanding library to promptly develop prototypes offering a collection of ready to use operators that encompass the nucleus of many metaheuristics including hybrid metaheuristics. Due to the large processing capabilities required for solving these optimization problems, JG2A and pALS can be deployed in a heterogeneous computational environment enabled by a grid solution based on Globus as the middleware grid, and Condor as the local resource manager.

For running JG2A/pALS simulations a CVC have been deployed in a computing room (Turing). A dedicated machine have been used for the CVC master, this machine has installed Globus Toolkit 4.0 and Condor 7.0. All machines have Debian 4.0 operating system. The tasks are sent by users through a JG2A/pALS interface to GRAM (Grid Resource Allocation Manager) and then GRAM send the tasks to Condor scheduler which distribute them among the Condor slaves. The infrastructure deployed for running JG2A/pALS tasks is shown in Fig. 6.

D. GISELA CVC

Universidad de los Andes participates in the EELA-2 (E-science grid facility for Europe and Latin America) project and we have an EELA-2 certified site (VO Uniandes) [49]. The last CVC deployed has been used for the integration of the UnaGrid infrastructure with a service grid infrastructure, as EELA-2. The EELA-2 sites use the gLite middleware for their operation, and they use several gLite components installed and configured in dedicated servers. As the number of such servers is limited, a project has been developed to integrate the UnaGrid processing capabilities to the EELA-2 local infrastructure on demand. The EELA-2 site is managed by the

DTI (University Technology Office). This site is currently a Resource Centre (RC) certified for using the gLite middleware. This RC currently offers 5 services guaranteeing the correct operation of the entire system: Information and monitoring service (BDII – RGMA), Security Service (CA-VOMS), Data Service (SE – LFC), Job Management Service (CE – WMS – UI) and Registration Service (RA).

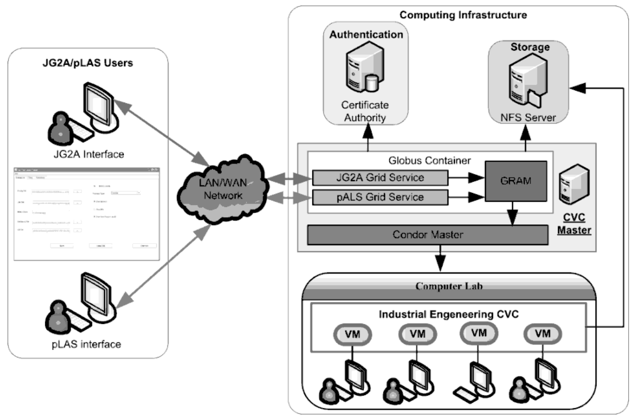


Fig. 6 industrial engineering CVC

The integration has been achieved by creating a CVC with the gLite middleware installed and configured [50]. This scheme has the basic components of the gLite infrastructure for job execution which are the Computer Element (CE) which is the point of entry to the local queue system (PBS, LSF, condor), and the Worker Nodes (WNs) which are in charge of job execution. The CE is installed within an ESXi server; and the WNs (35 total) are installed in a computer room with the PBS scheduler. The CE and the WNs are virtualized and they are executed on demand, by DTI staff, through GUMA. The CE is connected to the WMS (Workload Management System) belonging to the VO Uniandes from where jobs may be sent to be executed. In Fig. 7, the integration architecture of the opportunist virtual cluster to the VO Uniandes service infrastructure is shown.

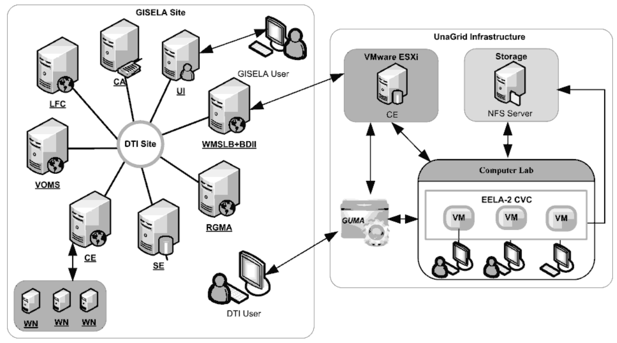


Fig. 7 GISELA CVC

VI. Performance Evaluation

Table I summarizes the execution time benefits of using Unagrid. Although these results are important as they show the gain in time that can be easily obtained by researchers, we were more interested on measuring the impact of sharing a physical machine between an end user and a CVC's node. To do this, we executed four types of tests: i) measuring the impact on end users when a CVC executes intensive processing applications (Fig. 8.a - 8.c). ii) Measuring the impact on the end users when they execute storage intensive applications (I/O) (Fig. 8.d - 8.f). iii) monitoring the processor usage from both the end user processes and the virtual machine executed in background, while this last had two cores assigned, and iv) comparing the job execution time on a CVC with the execution time on a dedicated cluster with the same hardware resources.

Table I results obtained in the use of the UnaGrid infrastructure

Application Name	Infrastructure Used	CPU Number	Job Number	Time by job (sec)	Exec. Time (days)
JG2A	PC	2	2880	3000	50,10
	Ind. Eng. CVC	70	2880	3120	1,50
BSGrid Model A	PC	2	150000	35	30,38
	Chem. Eng. CVC	70	150000	85	2,11
BSGrid Model B	PC	2	150000	63	54,69
	Chem. Eng. CVC	70	150000	111	2,75
HMMER	PC	2	4200	11700	284,40
	Bioinform. CVC	140	4200	12900	4,50

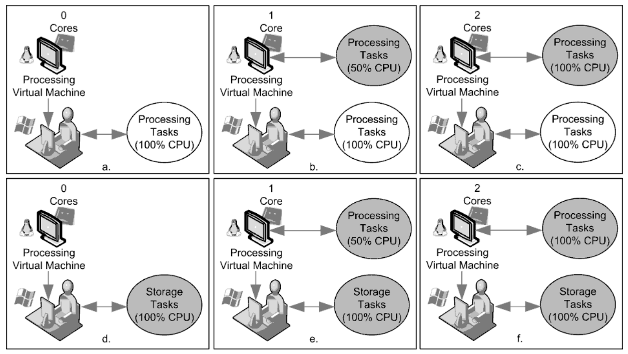

0 Cores
Processing Virtual Machine
Processing Tasks (100% CPU)
a.
1 Core
Processing Tasks (50% CPU)
Processing Virtual Machine
Processing Tasks (100% CPU)
b.
2 Cores
Processing Tasks (100% CPU)
Processing Virtual Machine
Processing Tasks (100% CPU)
c.
0 Cores
Processing Virtual Machine
Storage Tasks (100% CPU)
d.
1 Core
Processing Tasks (50% CPU)
Processing Virtual Machine
Storage Tasks (100% CPU)
e.
2 Cores
Processing Tasks (100% CPU)
Processing Virtual Machine
Storage Tasks (100% CPU)
f.

Fig. 8 performance evaluation tests executed

Table II and Table III summarize the results of impact on end users. As it is shown, regular users do not perceive (less than 3% degradation in the worst case) the execution of a virtual machine in his/her desktop machine. In these tables test1 to test4 refer to different sizes (cpu/file load) of the executed application on the desktop computer.

Table II results of the impact on intensive CPU end users

Environment/Test	Processing Task Completion Time (s)			
	Test 1	Test 2	Test 3	Test 4
Without VM	53,94	81,01	108,05	134,99
With a VM (1 Core)	54,16	81,42	108,39	135,58
With a VM (2 Cores)	54,21	81,46	108,58	135,60

Table III results of the impact on intensive I/O end users

Environment/Test	I/O Task Completion Time (s)			
	Test 1	Test 2	Test 3	Test 4
Without VM	104,10	259,85	521,16	1041,42
With a VM (1 Core)	105,66	262,43	526,63	1060,75
With a VM (2 Cores)	106,02	263,03	527,06	1063,07

A third set of tests were executed with the objective of monitoring the processor usage from both the end user processes and the virtual machine executed in background, while this last had two cores assigned; for such, intensive processing tasks were executed within both environments. The results are shown in Fig. 9.

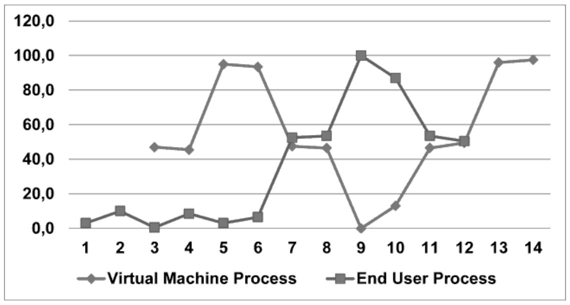


Fig. 9 CPU usage for virtual machine and end user process

In the test, after measuring CPU use with no virtual machine running, we initiate (time 3) a virtual machine using nearly 50% of the CPU, and at 5, we increase its computational requirements to something close to the 100%. We then modify the CPU need from the end user. Between 7 and 8 the user demands a 50% of the CPU and the virtual machine load automatically decreases to 50%. Between 9 and 10 the user increases the consumption to about 100% and the virtual machine automatically reduces their consumption to a minimum. Between 11 and 12 the end user goes back to a 50% demand, and after 12, the user leaves alone the physical machine. According to the results shown in Fig. 9, the virtual machine only uses idle processor cycles, guaranteeing a very low impact on the performance perceived by end users.

Finally Table IV shows an average overload of less than 17% of the time required for executing the jobs on the opportunistic infrastructure when compared to the time required on a dedicate infrastructure.

Table IV time execution comparison between an opportunistic and a dedicated infrastructure

Application	Infrastructure	Average Execution Time by Job (Minutes)	Overload (%)
HMMER	Dedicated	164,0	12,4
	Opportunistic	184,3	
BSGrid	Dedicated	10,2	16,7
	Opportunistic	11,9	
BLAST	Dedicated	302,6	9,3
	Opportunistic	330,7	

VII. Conclusions

The proposed opportunistic infrastructure has shown promising results with regard to offering new low-cost opportunities to meet the very specific processing capabilities required in the execution of applications of different research groups at Universidad de los Andes. CVCs breaks the limitations on size, accuracy and time currently experimented by our researchers. These results show the solution proposed as an opportunistic virtual infrastructure that can take advantage of the processing capabilities of the resources available in computer labs on a university campus, while users carry out their conventional activities, utilizing a key virtualization strategy in a non-intrusive manner.

New challenges will have to be faced in order to improve the processing capabilities offered: a requirement is to analyze how to guarantee quality of service, improving the best effort scheme currently in use, to define a storage system to obtain storage capacities in an opportunistic way, and to define a job-submit portal to make easier the use of the infrastructure.

Acknowledgment

This work presents software developed and tested by a team of which the authors are only a small part. The authors thank and acknowledge the support provided by the EELA-2 and GISELA projects. Both projects are co-funded by the European Commission under the Seventh EU Framework Programme.

References

[1] Ian Foster and Carl Kesselman, *The Grid 2: Blueprint for a future computing infrastructure*., 2003.
[2] TORQUE Resource Manager. [Online]. http://www.clusterresources.com/products/torque-resource-manager.php
[3] R. Henderson and D. Tweten, "Portable batch system: External reference specification," in *Technical report, NASA, Ames Research Center*, 1996.
[4] Sun Microsystems. Sun Grid Engine. [Online]. http://www.sun.com/software/sge/
[5] IBM. Tivoli Workload Scheduler LoadLeveler. [Online]. http://www-03.ibm.com/systems/clusters/software/loadleveler/index.html
[6] The Globus Alliance. Globus Toolkit Homepage. [Online]. http://www.globus.org/toolkit/
[7] EGEE. gLite. [Online]. http://glite.web.cern.ch/glite/
[8] David Toth and David Finkel, "Characterizing resource availability for volunteer computing and its impact on task distribution methods," in *Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, Wisconsin, 2007, pp. 107-114.
[9] David Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer, "SETI@home An Experiment in Public-Resource Computing," *Communications of the ACM*, vol. 45, pp. 56-61, November 2002.
[10] Francisco Brasileiro and Rodrigo Miranda, "The OurGrid Approach for Opportunistic Grid Computing," in *Proceedings of the First EELA-2 Conference*, Bogotá D.C., 2009.
[11] David P. Anderson and Gilles Fedak, "The computational and storage potential of volunteer computing," in *In CCGRID '06*, 2006, pp. 73–80.
[12] L. Kroeker, "The Evolution of Virtualization," in *Communications of the ACM*, vol. 52, New York, 2009, pp. 18-20.
[13] V. Chaudhary, J. P. Walters, S. Guercioa, and S. Gallo, "A Comparison of Virtualization Technologies for HPC," in *22nd International Conference on Advanced Information Networking and Applications*, New York, 2008, pp. 861-868.
[14] S. Yamasaki, N. Maruyama, and S. Matsuoka, "Model-Based Resource Selection for Efficient Virtual Cluster Deployment," in *3rd Int. Workshop on Virtualization Technology in Distributed Computing*, New York, 2007.
[15] I. Foster et al., "Virtual Clusters for Grid Communities," in *Sixth IEEE International Symposium on Cluster Computing and the Grid*, New York, 2006, pp. 513-520.
[16] Shoch, John F.; Hupp, Jon A., "The "Worm" Programs Early Experience with a Distributed Computation," *Communications of the ACM*, vol. 25, no. 3, marzo 1982.
[17] Dana Petcu and Marius Petcu, "Distributed Jess on a Condor pool," in *Proceedings of the 9th WSEAS International Conference on Computers*, Wisconsin, 2005, article 11, 5 pages.
[18] GIMPS: Great Internet Mersenne Prime. [Online]. http://www.mersenne.org/
[19] Distributed.Net. [Online]. http://www.distributed.net

[20] David Anderson, "BOINC: A System for Public-Resource Computing and Storage," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Washington, 2004.

[21] Shahram Amin and Mohammad Ahmadi, "Distributed resource scheduling in grid computing using fuzzy approach," in *Proceedings of the 12th WSEAS international conference on Computers*, Wisconsin, 2005, pp. 820-825.

[22] Shahram Amin and Mohammad Ahmadi, "A balanced scheduler for grid computing," in *Proceedings of the 8th WSEAS conference on Simulation, modelling and optimization*, Wisconsin, 2005, pp. 59-64.

[23] L. Sarmenta et al, "Bayanihan Computing.NET: Grid Computing with XML Web Services," in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, Berlin, 2002, pp. 434.

[24] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steve Tuecke, "Condor-G: A computation management agent for multi-institutional grids," in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, San Francisco, 2001, pp. 55-63.

[25] Andrei Goldchleger, Fabio Kon, Alfredo Goldman, Marcelo Finger, and Germano Capistrano Bezerra, "InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 5, pp. 449 - 459, 2004.

[26] Rafael Fernandes and Francisco Da Silva, "Migration transparency in a mobile agent based computational grid," in *Proceedings of the 5th WSEAS international conference on Simulation, modelling and optimization*, Wisconsin, 2005, pp. 31-36.

[27] J. Smith and S.K. Shrivastava, "A system for fault-tolerant execution of data and compute intensive programs over a network of workstations," in *In Lecture Notes in Computer Science*, vol. 1123, 1996.

[28] Walfredo Cirne et al., "Grid computing for Bag-of-Tasks applications," in *In Proceedings of the I3E2003*, 2003.

[29] David Abramson, Rajkumar Buyya, and Jonathan Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker," in *Future Generation Computer Systems (FGCS) Journal*, 2002, pp. 1061-1074.

[30] R. Wolski, J. Plank, J. Brevik, and T. Bryan, "Analyzing market-based resource allocation strategies for the computational grid," in *International Journal of Highperformance Computing Applications*, 2001.

[31] Andrade Nazareno, Brasileiro Francisco, Cirne Walfredo, and Mowbray Miranda, "Automatic Grid Assembly by Promoting Collaboration in Peer-to-Peer Grids," in *Journal of Parallel and Distributed Computing*, vol. 68, 2007.

[32] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray, "Automatic grid assembly by promoting collaboration in peer-to-peer grids," in *Journal of Parallel and Distributed Computing*, vol. 67, 2007, pp. 957- 966.

[33] Xen. How are Hypervisors Classified? [Online]. http://www.xen.org/files/Marketing/HypervisorTypeComparison.pdf

[34] Eric McIntosh and Andreas Wagner, "CERN MODULAR PHYSICS SCREENSAVER OR USING SPARE CPU CYCLES OF CERN'S DESKTOP PCS," 2002.

[35] CERN. LHC@home. [Online]. http://lhcathome.cern.ch/

[36] CERN. CernVM Home. [Online]. http://cernvm.cern.ch/cernvm/

[37] Parallels. Parallels. [Online]. http://www.parallels.com/

[38] F. Cappello et al., "Grid'5000: a large scale and highly reconfigurable grid experimental testbed," in *The 6th IEEE/ACM International Workshop on Grid Computing*, Seattle, 2005, p. 8.

[39] OpenNebula.org. The Open Source Toolkit for Cloud Computing. [Online]. http://www.opennebula.org

[40] Mario Villamizar, Harold Castro, and Andres González, "BacteriumSimulatorGrid (BSGrid) - Tool for Simulating the Behavior of the Bacillus thuringiensis (B. thuringiensis) Bacterium in Cluster and Grid Infrastructures," in *HIBI '09. International Workshop on High Performance Computational Systems Biology*, Trento, 2009, pp. 3-12.

[41] González A., Castro H., Villamizar M., Cuervo N., Lozano G., Orduz S. and Restrepo S., "Mesoscale Modeling of the Bacillus thuringiensis Sporulation Network Based on Stochastic Kinetics and Its Application for in Silico Scale-down," in *HIBI '09. International Workshop on High Performance Computational Systems Biology*, Trento, 2009.

[42] N. Cuervo et al., "Modelamiento a mesoescala de la red de esporulación de Bacillus thuringiensis usando cinética estocástica y su aplicación para scale down in silico," in *IV Simposio sobre biofábricas*, Medellín, 2009.

[43] David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga, "The LONI Pipeline Processing Environment," *NeuroImage*, vol. 19, no. 3, pp. 1033-1048, July 2003.

[44] Ivo Dinov et al., "Efficient, distributed and interactive neuroimaging data analysis using the LONI Pipeline," in *Frontiers in Neuroinformatics*, 2009.

[45] Vargas A.M., Ocampo L.M.Q., Cespedes M.C., Carreno N., Gonzalez A., Rojas A., Zuluaga A.P., Myers K., Fry W.E. and Jimenez P., "Characterization of Phytophthora infestans Populations in Colombia: First Report of the A2 Mating Type," in *Phytopathology*, 2009, pp. 82-88.

[46] S. Restrepo et al., "Computational Biology in Colombia," *PLOS Computational Biology*, vol. 5, no. 10, 2009.

[47] Bernal A., Ramirez M.A., Castro H., Walteros J.L. and Medaglia, A.L., "JG2A: A Grid-enabled object-oriented framework for developing genetic algorithms," in *IEEE Systems and Information Engineering Design Symposium SIEDS'09*, Virginia, 2009.

[48] A. Bernal and H. Castro, "pALS: An Object-Oriented Framework for Developing ParallelCooperative Metaheuristics," in *24th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Atlanta, 2010, pp. 1-8.

[49] H. Castro et al., "EELA: una infraestructura para e-ciencia en Latinoamérica," *Revista de Ingeniera de laUniversidad de los Andes*, vol. 1, no. 23, pp. 26-32, May 2009.

[50] Artur Miller, Harold Castro, Mario Villamizar, and Eduardo Rosales, "Integrating a virtual Opportunistic Infrastructure to an EELASite," in *2nd EELA-2 Conference*, Choroní, 2009, pp. 209-216.

Harold Castro was born in Bogota, Colombia, in 1967. He graduated in computing and systems engineering at Universidad de los Andes in Bogota, Colombia in 1989, he got a D.E.A (MSc) from the Institut National Polytechnique de Grenoble (INPG), in Grenoble, France in 1991 and since 1995 he holds a Ph.D. in computer science from INPG also.

Since 2005 he is associate professor at the Computing and Systems Department at Universidad de los Andes. He is the director of the COMIT (Communications and Information Technology) research group which main research focus are distributed systems. Dr. Castro personally leads institutional and national grid initiatives, and his interest areas are: distributed systems, grid and cloud computing, and mobile computing.

Mario Villamizar was born in Cúcuta, Colombia, in 1985. He received the BS degree in Systems Engineering at the Universidad Francisco de Paula Santander (Norte de Santander, Colombia) in 2007, and the MS degree in Systems and Computing Engineering at the Universidad de los Andes (Bogotá, Colombia) in 2010.

Since 2010 he has been an Instructor Professor of the Department of Systems and Computing Engineering at the Universidad de los Andes. He has more than 15 publications (including papers, journals and book chapters) that show the research results of his participation in different national and international grid and cloud projects such as EELA, GISELA, Grid Colombia, UnaGrid, Bio-UnaGrid and UnaCloud. His research interests lie mainly in cluster/grid/cloud computing technologies, in particular: opportunistic grids/clouds, grid/cloud resource management and scheduling, load balancing, and configuration and deployment of virtual machines on-demand.

Eduardo Rosales was born in Pasto, Colombia, in 1983. He received the BS degree in Systems Engineering at the Universidad Católica de Colombia (Bogotá, Colombia) in 2007, and the MS degree in Systems and Computing Engineering at the Universidad de los Andes (Bogotá, Colombia) in 2010.

Since 2008, he has been participating in the COMIT (Communications and Information Technology) research group, specifically in the UnaGrid and UnaCloud projects. He also worked as an IT Solution Architect. Since 2011, he has been working as a DQM Offline Engineer at the CMS (Compact Muon Solenoid) experiment at CERN (Geneva, Switzerland). He has more than 10 publications including papers, journals and book chapters. His research interests include cluster, grid and cloud computing, in particular solutions based on opportunistic and volunteer computing systems.