18

GFOG: Green and Flexible Opportunistic Grids

Harold Castro, Mario Villamizar, German Sotelo, Cesar O. Diaz, Johnatan Pecero, Pascal Bouvry, and Samee U. Khan

18.1 INTRODUCTION

Large-scale computing platforms and current networking technologies enable sharing, selection, and aggregation of highly heterogeneous resources for solving complex real problems. Opportunistic grids are distributed platforms built out of the available resources (volunteers or donors) of an existing hardware platform that harvest the computing power of nondedicated resources when they are idle. Internet opportunistic grids are designed to take advantage of the capabilities of thousands or millions of desktop computers distributed through the Internet. On the other hand, institutional opportunistic grids involve the use of hundreds or thousands of computers available in a single organization or institution. In both opportunistic grid types, on the volunteer or donor desktops, an agent is executed to control, manage, and monitor the execution of grid tasks. Two main approaches are used to execute the agent:

- A desktop client runs an executable program to provide access to a set of resources of the desktop in a limited manner.
- A client executes a virtual machine, which is used to isolate and limit the execution environment required to execute the grid tasks.

In this last approach, isolation of both environments is critical; therefore, the use of virtualization is a promising approach.

Scalable Computing and Communications: Theory and Practice, First Edition. Samee U. Khan, Lizhe Wang, and Albert Y. Zomaya.

According to the TOP500 Supercomputer Sites [1], Latin American countries do not provide sufficient dedicated clustering infrastructures compared with the United States or Europe. Dedicated clusters are expensive. Therefore, for these countries to provide a certain level of high-performance computing (HPC), they may use existing resources available at universities or companies. For universities, computer labs are underused resources to take advantage of. In the case of Universidad de los Andes, dedicated clusters are composed of less than 200 cores. If we take into account that all of the available computer labs can provide more than 2000 cores, an opportunistic grid infrastructure could be a good approach. To implement this approach, some issues need to be studied beforehand, such as the impact of energy consumption, isolation of the owner-user (e.g., the human user currently using the hardware) and grid service (i.e., the grid task) environments, and the division of resources between these two environments, keeping in mind that owner-users have priority over any grid computing environment. Moreover, energy consumption is another important concern; it is essential nowadays to provide a mechanism that is also energy-efficient. Virtual machines are a natural solution to provide complete isolation between both environments, namely, grid computing and owner-user ones. Virtualization provides mechanisms that guarantee the properties mentioned above; however, hardware resources must be shared between these two environments in such a way that both kinds of users have sufficient resources required to perform their tasks, and the energy efficiency of using virtualization to provide HPC environments executed on commodity desktops is still in doubt.

In this work, we exploit a virtualization technique used and tested on an existing opportunistic grid infrastructure, termed UnaGrid, to share resources between desktop computers and grid computing environments. We show that using this technique leads to a reduction of energy consumption with regard to using a *dedicated* computing architecture and, furthermore, improves its scalable performance.

18.2 RELATED WORK

Over the years, desktop grid systems have been working as a particular kind of volunteer computing system, where the computing resources are provided by individuals and have become among the largest distributed systems in the world [2]. Nowadays, there are several approaches to classify desktop grid systems, moreover to visualize them in an energy-efficient opportunistic grid environment. Therefore, we must take into account three factors: scalable opportunistic grids, virtualization technologies, and energy consumption in opportunistic grids.

18.2.1 Scalable Opportunistic Grids

Scalability is one of the most important goals that must be met to make building a desktop grid worth the effort. In other words, scalability refers to the capability of the system to adapt to any change in infrastructure, location, or management. Therefore, the scalability in one system can be identified by at least three components:

1. numerically or with respect to its size; that is, the feasibility to add more users and resources to the system [3]

2. geographically scalable; that is, the distance between the farthest nodes within the system
3. administrative scalability; that is, the ability to manage even if it encompasses many independent administrative organizations [4].

Opportunistic grids or desktop grid and volunteer computing systems (DGVCSs) have allowed the aggregation of millions of partially computing resources for different scientific projects. Initial projects such as Worm [5] and Condor [6] focused on taking advantage of the idle resources available in an organization. With the massive growth of the Internet, projects such as GIMPSs |citeMers2009 and SETI @home [7] have begun to use the capabilities of volunteers around the world for executing single-purpose applications. The Distributed.net cite Distri2009 and BOINC [8] projects were designed to support the execution of multiple-purpose applications over an Internet infrastructure whose resources are managed or controlled by a central organization. When grid computing emerged as a large and scalable computing solution that allowed the integration of resources of different organizations, projects such as Bayanihan.NET [9], Condor-G [10], and InteGrade [11] began to use grid middleware to allow the aggregation of idle or dedicated resources available in different administrative domains.

All of the aforementioned projects execute grid tasks directly on the physical resources using a desktop client installed on each desktop computer, so they have some problems to support new or existing applications, such as application portability to different architectures or operating systems, isolation of environment used by the owner-user and grid user (security, intrusion, and checkpoint issues), and modification of legacy or new applications. When virtualization technologies appear, projects such as XtremWeb [12], OurGrid [13], ShareGrid [14], CernVM [15], BOINC-CernVM [16], SUCSI [17], UnaGrid [18], and Cloud@home [19] began to use virtual machines to execute the grid tasks. In these projects, a virtual machine is executed on each desktop computer and used to execute the grid tasks in a sandbox environment. The use of virtualization imposes some performance degradation issues; however, virtualization facilitates the portability of application to different architectures and operating systems, the isolation of environments, and the incorporation or legacy application without requiring recompilations or modifications. With recent virtualization advances, the performance is increasingly better. UnaGrid [18] is an opportunistic grid solution developed to allow different research groups to take advantage of the idle processing capabilities available in the computer labs at a university campus, when they need computational capabilities using an on-demand approach. Unlike OurGrid, ShareGrid, CernVM, and BOIN-CernVM, UnaGrid was designed to be a large and shared institutional opportunistic grid (not an Internet DGVCS). The SUCSI project uses a virtualization strategy similar to that used by UnaGrid; however, SUSCI changes the way researchers execute the applications because it does not allow grid users to customize and execute the application in its native environment (command-line, operating systems, grid middleware, libraries, etc.). Due to the use of customized virtual clusters (CVCs), in UnaGrid, researchers can continue executing their application in the native environment and can deploy the CVCs when they need to execute grid tasks; these features guarantee the high usability and the efficient use of the infrastructure. Cloud@home is a proposal to define a desktop cloud computing solution

that takes advantage of the benefits of the cloud computing paradigm and the opportunistic grid systems; although it is a design proposal, it is similar to UnaGrid in that it also uses an on-demand approach to request resources (or services in the cloud computing world) when they are needed. However, Cloud@home is designed to integrate different commercial and open cloud computing service providers, and heterogeneous and independent nodes as those found in DGVCSs.

18.2.2 Virtualization Technologies

Virtualization is a mechanism used to offer a certain kind of virtual machine environment, which is possible only given the right combination of hardware and software elements [20, 21]. A virtual machine is a software implementation of a computer that performs applications and programs like a real-life computer. Virtual machines are divided into two major categories, based on their use and degree of association to any real computer: (1) A system virtual machine grants an entire system platform, which carries out the execution of a complete operating system, and (2) on the contrary, a process virtual machine is intended to support a single process execution [20]. An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the configuration of the virtual machine, meaning it cannot use more than the values it has configured as part of its virtual world. System virtual machines allow sharing the underlying physical machine resources between different environments, each one running its own operating system [22, 23]. The software layer controlling the virtualization is called a hypervisor. The main advantage of this kind of virtual machine is that several operating system platforms can coexist on the same physical machine in total isolation from each other. This way, some quality of service is provided because virtual machines configured with limits can use only the amount of resources it has set as its virtual world or has been enforced to set. There are many types of virtual machine software that provide the aforementioned characteristics, the most popular being Virtual Box, VMware, and Xen. UnaGrid uses VMware Workstation [24] to manage each virtual machine (e.g., operations such as start, resume, and power-off) and VMware Player as the system virtual machine. UnaGrid also uses the hardware-assisted virtualization (or native virtualization) features available in desktops with physical processors that support this technology, such as AMD-V [25] and Intel-VT [26]. The use of this technology improves the performance when grid applications are executed in virtual machines.

18.2.3 Energy Consumption on Opportunistic Grids

Virtualization technologies have been used in data centers for different purposes, one of which is energy consumption saving. Different analyses have shown than virtualization technologies allow reduction of the energy consumption by more than 30% [27]; depending on the energy optimization techniques, the virtualization technologies, and the application and operating systems executed on virtual machines, this percentage may vary [28]. In conventional computing clusters, the energy consumption is based mainly on the consumption of servers and cooling systems. In desktop grids, the energy consumption is based on the consumption of the desktop machines used to execute the grid tasks. Cooling consumption is not taken into account due to desktop machines being regularly available in large open spaces. Few efforts have been developed to analyze the energy consumption in opportunistic

grids. From the energy consumption point of view, most desktop grids select the resources to execute grid tasks using an algorithm that takes into account only the best physical resource to execute the jobs without having in mind the energy consumption used to execute the tasks. Regarding the Condor project, an effort [29] has been made to analyze the energy consumption of a Condor cluster and to optimize the energy consumption when desktops used to execute the grid tasks are selected. A more detailed and general project, termed DEGISCO, has been proposed [30]. This DEGISCO proposal examines several aspects of energy consumption and computational performance of different desktop grids around the world; however, at the time of this publication, the project is in its initial phase. In the current UnaGrid implementation, when a user requires the execution of virtual machines, these virtual machines are deployed on turned-on desktops using a random algorithm, without taking into account the current state of the desktop (turned off, hibernating, idle, or busy), and we consider that the energy consumption used to execute a grid task on a virtual machine depends on the state of the desktop used to execute it.

18.3 UNAGRID INFRASTRUCTURE

18.3.1 UnaGrid Architecture

UnaGrid is a virtual opportunistic grid infrastructure that takes advantage of the idle processing capabilities available in the computer labs of Universidad de los Andes. UnaGrid was designed to meet four main goals:

1. to allow the aggregation of the idle processing capabilities available in heterogeneous computer labs
2. to allow different research groups to use those capabilities when they require HPC, using an on-demand approach
3. to allow researchers to continue executing applications in the native environment and using the cluster or grid middleware they have been using
4. to operate without being intrusive to the owner-users of the desktops.

To achieve these goals, virtualization technologies are used. Virtualization allows the deployment, on-demand, of several scalable CVCs capable of providing the amount of HPC required for the development of different projects. A CVC is a set of interconnected desktops executing virtual machines through virtualization tools. A virtual cluster can be set up on a large number of desktop computers in which a virtual machine image performs as a slave of the cluster, and a dedicated machine performs as the master of the virtual cluster. Virtual machines use the idle capabilities permanently while owner-users carry out their daily activities. To operate in a nonintrusive manner, virtual machines are executed as low-priority background processes, guaranteeing that the owner-users have available all of the computational resources (if such are required), while the virtual machine consumes only the resources that the owner-users are not using (or all of these in the case of unused computers). To facilitate different research groups deploying several CVCs on the same physical infrastructure, an image of each one of the possible CVCs that may be executed is stored on each physical computer. The UnaGrid architecture is shown in Figure 18.1.

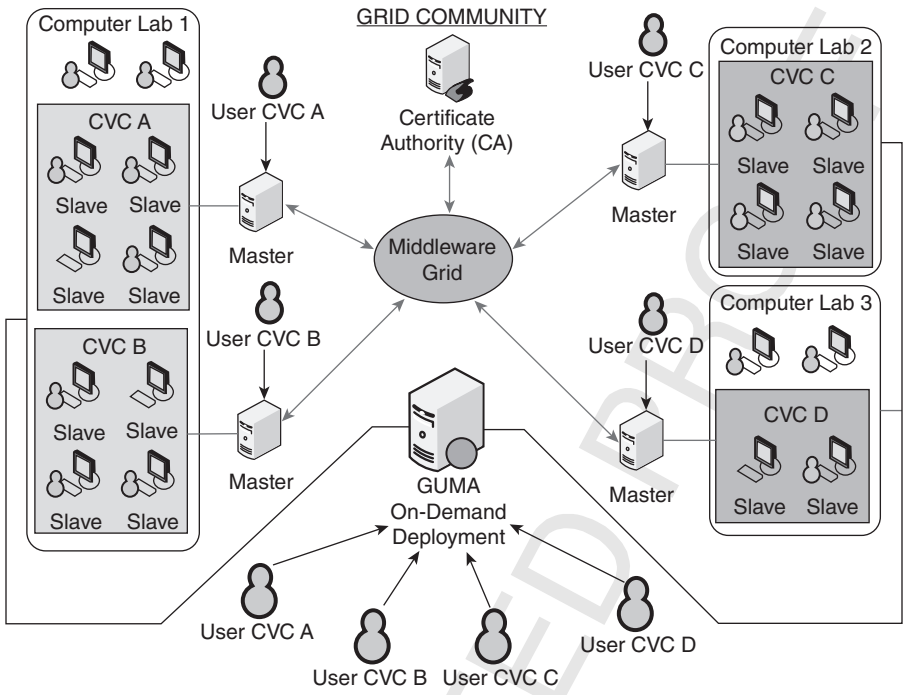

FIGURE 18.1. *UnaGrid architecture.*

Another aspect to take into account when implementing a CVC is handling the data corresponding to each application, so a distributed file system would seem to be a good approach. For UnaGrid, a network file system server is used for the complete infrastructure. The final users of the UnaGrid infrastructure can customize a virtual cluster and deploy it over different desktops. The deployment of a virtual machine requires a specific configuration concerning grid users, applications, and cluster and grid middleware, such as Condor, Globus, Sun Grid Engine, gLite, and PBS. The solution implemented is focused on the execution of a single virtual machine per physical computer in order to avoid competition of resources between virtual machines, and the collaborative work between research groups. For using the infrastructure, researchers use a web portal, termed Grid Uniandes Management Application (GUMA) [18]. GUMA allows for users from different research groups to deploy, on-demand, CVCs on desktops available in different computer labs, for specific time periods. GUMA communicates with the Java agents installed on each desktop to deploy the CVCs required by grid users. GUMA uses a client/server model with authentication, authorization, and privacy mechanisms, providing many services to manage the UnaGrid infrastructure from thin clients (only a Web browser is required), hiding the complexities associated with the location, distribution, and heterogeneity of computing resources and providing an intuitive graphical user interface (GUI). Management services include selection, shutdown, and monitoring of physical and virtual machines. Grid users can also manage their own virtual machines. GUMA is the main contribution of UnaGrid to DGVCSs because it gives high usability to an opportunistic system, using an on-demand deployment approach.

18.3.2 UnaGrid Implementation

UnaGrid has been tested executing applications from several scientific domains, all of them bag-of-tasks applications, including biology, chemical, and industrial engineering applications. In the biology domain, the HAMMER application was utilized in the analysis of genetic sequencing of the *Phytophthora infestans* genome [31]. In the chemical domain, the BSGrid application was used for the simulation of the *Bacillus thuringiensis* bacterium [32], and in the industrial engineering domain, the JG2A framework was used to solve an optimization problem in the routing of vehicles and design of routes [33]. When grid users require large processing capabilities, the addition of CVCs becomes handy. There are three options to make this feasible: (1) Configure and deploy more virtual cluster images; (2) take advantage of the aggregate capabilities of computer resources offered by some schedulers; and (3) install a grid middleware on master nodes allowing the execution of applications on different virtual clusters. To manage this situation, Globus Toolkit 4.2 is used, so the installation of a certificate authority becomes essential. The UnaGrid implementation is shown in Figure 18.2.

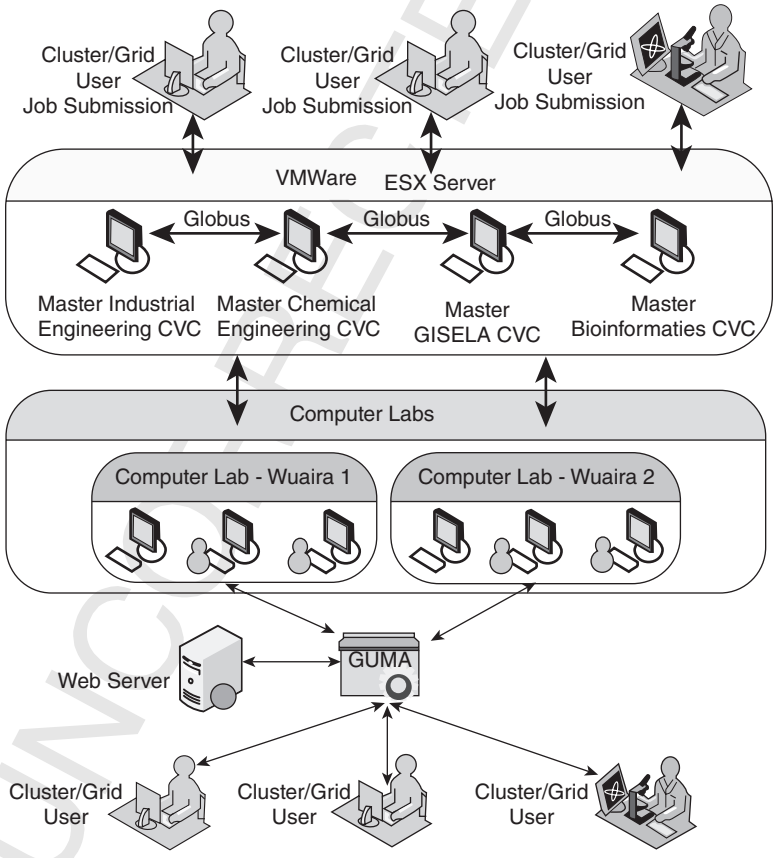

FIGURE 18.2. *UnaGrid implementation.*

18.4 ENERGY CONSUMPTION MODEL

Currently, all of the desktops that may execute as virtual machines are always turned on during business hours (6 AM–9 PM). As mentioned above, UnaGrid currently selects the physical machine to execute virtual machines using a random algorithm; however, the physical machines may be in four possible states (idle, busy, hibernating, turned off) and those states are not taken into account by UnaGrid where new virtual machines are going to be executed. Our hypothesis is focused in that UnaGrid should select physical machines where there are users doing daily tasks. Although this can be contradictory with the idea of getting the maximum resources for our CVCs, we advocate that the gain in energy savings is greater than the loss in computing power available for the scientific applications running on UnaGrid.

18.4.1 Energy Consumption for a Physical Machine

Let us suppose there is a real function f that returns the energy consumption rate of a physical computer given its CPU usage percentage. Let assume that f is crescent, so

$$\frac{\delta f}{\delta x} > 0, \quad \text{for} \quad 0 \le x \le 100. \tag{18.1}$$

When a computer is being used (busy state), its monitor is power up, so the function that describes the energy consumption rate (ECR) of a computer with a user is

$$f_{\text{user}} = f(x) + \text{MEC}, \tag{18.2}$$

where monitor energy consumption (MEC) is the energy consumed by the monitor when there is a user using a physical machine. In idle state, the monitor is turned off automatically, so in any other state, the monitor is turned off.

When there are no virtual machines in execution on a physical desktop, the energy consumption rates and CPU usage of a desktop according to the possible desktop states are as shown in Table 18.1, where P_{user} is the mean processor usage when there is a user using a physical machine (state 2) and E_{user} is the energy consumption rate of a physical machine when there is a user using it. E_{hib} is the energy consumption rate when a physical machine is hibernated (state 3).

When a virtual machine of the UnaGrid infrastructure is being executed on a physical machine, we assume the physical machines increase the processor usage to

TABLE 18.1. Energy Consumption Rate and Processor Usage for Desktops Computers

Computer State	Energy Consumption Rate without Virtual Machine	Mean Processor Usage
1 (idle)	$f(0)$	0
2 (busy)	E_{user}	P_{user}
3 (hibernation)	E_{hib}	0
4 (turned off)	0	0

TABLE 18.2. Energy Consumption Rate and Processor Usage for Desktops Computers Running Virtual Machines

Computer State	Energy Consumption Rate with Virtual Machine
1 (idle)	$f(100)$
2 (busy)	$f(100)$ + MEC
3 (hibernation)	$f(100)$
4 (turned off)	$f(100)$

TABLE 18.3. ECR Used by UnaGrid Intensive-Computing Task

Computer State	Execution Time	ECR for Intensive-Computing Task (ECR with VM − ECR without VM)
1 (idle)	T_i	$f(100) - f(0)$
2 (busy)	$\frac{100}{P_{\text{free}}} T_i$	$f(100)$ + MEC − E_{user}
3 (hibernation)	T_i	$f(100) - E_{\text{hib}}$
4 (turned off)	T_i	$f(100)$

VM, virtual machine.

100%. So, in Table 18.2, we present the energy consumption rate, for each state, of a desktop computer running virtual machines executing CPU-intensive tasks.

If we suppose that the execution time of a CPU-intensive task is linearly proportional to the amount of free processor used of the machine that is running it, we have the following execution time for a grid task:

$$T_i(P_{\text{free}}) = \frac{100}{P_{\text{free}}} T_i, \tag{18.3}$$

where T is the amount of time needed to run the task when the physical machine has all its free CPU, and T_i is the time required to execute a grid task in a busy desktop with a free CPU capacity of P_{free}. The amount of energy and time needed to run a CPU-intensive task is shown in Table 18.3 (obtained from Tables 18.1 and 18.2 and Equation 18.3).

To define which is the best state to execute virtual machines, we executed different experimental tests, as shown in the next section.

18.4.2 Energy Consumption for a Computer Lab

To scale the energy consumption of a computer lab, we use three consumption energy values for a desktop machine: (1) a minimum (min) when the machine is idle, (2) an average (avg) when an owner-user is making use of a physical machine, and (3) a maximum (max) when a physical machine has both environments running or just the HPC environment. To simplify the model during the calculation of a complete lab, the hibernation and turned-off states were grouped into the idle state due to the fact that in the current UnaGrid implementation, the machines are always turned on during business hours. With these assumptions, if during a time period

there exist n physical machines available, m of which are vacant for HPC tasks, u has an owner-user working, and a virtual cluster of l virtual machines needs to be initiated. A dedicated cluster has an expected energy consumption of $E\_HPC = l \times f(\max) + (m - 1) \times f(\min)$, and a computer lab environment has an expected energy consumption of $E_u = u \times f(\text{avg}) + (n - u) \times f(\min)$ during the time the HPC environment is running. In a computer lab environment, with n physical machines available, u of which have an owner-user working, and a virtual cluster of l virtual machines needs to be initiated, there are two ways to choose the physical machines where GUMA will start the virtual machines: (1) randomly or (2) where there are users working. The first approach can be modeled using hypergeometric distribution causing the expected number of machines to launch where the owner-user is $\mu = (u \times 1/n)$. Thus, the energy consumption expected during the job execution is

$$E_r = l \times f(\max) + (u - \mu) \times f(\text{avg}) + [n - u - (l - \mu)] \times f(\min). \tag{18.4}$$

On the other hand, for the second approach, as it will first choose the machines where there are users working, the energy consumption expected is

$$\begin{aligned} E_s &= l \times f(\max) + (u - l) \times f(\text{avg}) + (n - u) \times f(\min) \quad l \le u \\ E_s &= l \times f(\max) + (n - l) \times f(\min) \quad l > u. \end{aligned} \tag{18.5}$$

Let us say that the current energy consumption rate of a dedicated cluster and a computer lab while UnaGrid is not in execution can be expressed as $E_T = E\_HPC + E_u$. When UnaGrid is in execution, it is easy to see that $E_s \le E_r$. Moreover, $E_r \le E_T$, which implies that opportunistic grids are energy-savers. The percentage gained by randomly chosen physical machines is $G_r = (E_T - E_r)/E_r$ and by selectively chosen physical machines is $G_s = (E_T - E_s)/E_s$. So, the approach used by GUMA aims to minimize the energy consumption.

18.5 EXPERIMENTAL RESULTS

We execute tests to evaluate the UnaGrid intrusion level over owner-user, to calculate the energy consumption rate of a single desktop machine and a complete computer lab, and to analyze the performance of the UnaGrid infrastructure to execute grid tasks.

Two computer labs were used to deploy the UnaGrid infrastructure described here. The experimental grid task for each scenario was a multiple multiplication of double-precision matrices. The virtualization technology used was VMware Workstation, with four cores for each virtual machine, and currently, our desktop computers have the following configuration:

- Hardware configuration of physical machines: processor, Intel Core™ i5, 3.33 GHz; disk space, 250 GB; memory, 8 GB; operating system, Windows 7 Professional; network interface card, 1.
- Hardware configuration of virtual machines: processors, four virtual cores; disk space, 20 GB; memory, 2 GB; operating system, Debian 5.0; virtual network interface card, 1.

TABLE 18.4. Results for Disk Usage

Scenario	File Size (GB)	Zip Completion Time
Without running VM	1	521.16
Running VM with one core	1	526.63
Running VM with two cores	1	527.06

The grid tasks used to measure the profit of the opportunistic infrastructure consists of executing 100×100 matrix (A) multiplication tasks to compute $A^{40,000}$.

18.5.1 Intrusion Level

The tests we performed aimed at measuring the impact of executing an HPC virtual machine on a desktop computer. We wanted to know how owner-users are perturbed because of the execution of jobs in the virtual machine as well as how efficient a virtual cluster as a tool for HPC is. Three different variables determine the overall performance of an application: I/O, CPU, and memory consumption. As we are proposing a virtual environment for execution of CPU-intensive applications, we will not consider the I/O requests from the computing environment. Also, as memory consumption is limited by the virtual machine configuration, we do not have to study its impact on the owner-user experience. The only impact we have to analyze is the one caused by the use of the CPU. We simulated then different kinds of owner-users sharing their physical machines with CPU-intensive grid environments.

18.5.1.1 I/O Performance To measure the impact of the HPC virtual machine on the I/O performance for the owner-user, a file was zipped in this environment and the completion time is taken, first, without running a virtual machine and then, running a virtual machine with one and two cores, respectively. The results for this test are presented in Table 18.4. The performances of the I/O operations are not greatly impacted by the addition of a running virtual machine with two cores (less than 3%). The reason for this is that the grid tasks running in the virtual machines do not require a large amount of I/O resources in order to be executed. Thus, the additional time required to complete the file zipping is not noticeable by the owner-user.

18.5.1.2 CPU Performance The goal of this experiment is to measure the amount of CPU lost by the owner-user when his or her host is used for an HPC virtual machine. A Java application that requires a large amount of CPU resources as owner-user is executing a task and the completion time of the task was taken, first, without running a virtual machine and, second, running a virtual machine with one and two cores. The tasks consisted of a simple test using a naive algorithm. The results for this test are presented in Table 18.5. For this scenario, the CPU usage perception is not affected by the use of a running virtual machine with one and two cores (less than 1%).

One observation is that both environments (owner-user and HPC virtual machine) share the two cores of the underlying physical machine (i.e., as the owner-user process does not require the whole computing power); one core is used by this

TABLE 18.5. Results for CPU Usage

Scenario	File Size (GB)	Task Completion Time
Without running VM	524,287	98.01
Running VM with one core	524,287	98.42
Running VM with two cores	524,287	99.46

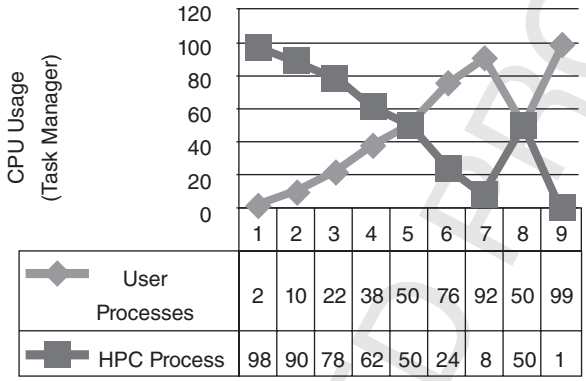

FIGURE 18.3. *CPU usage for HPC and owner-user processes.*

process and the other one is used by the HPC environment. To explore this subject, it is necessary to look at another scenario, one where the user process demands the whole computing power. We analyzed the CPU usage of the owner-user and the HPC virtual machine when both environments are in execution; the results are shown in Figure 18.3.

An HPC virtual machine and an owner-user virtual machine were initiated. Both virtual machines were configured with two cores and two CPU-intensive tasks, setting the priority for the HPC computing process to idle. In Figure 18.3, point (1) represents the situation when the HPC process is consuming all computing power because the owner-user is not processing CPU-intensive tasks. In points (2) and (3), the owner-user processes starts demanding more computing power. In points (4)–(7), the owner-user initiates a virtual machine and starts demanding computing power to complete the process. Then, when the owner-user virtual machine is completely initiated, the operating system yields 50% CPU usage for each environment (point 8). Finally, in point (9), when a CPU-intensive task is sent by the owner-user, all CPU resources are completely granted to the owner-user processes. Here, it is clear that the operating system yields the computing power to the owner-user processes because they have higher priority than the HPC virtual machine. As a result, the tasks in the HPC environment are not progressing to achieve their goal, and they will take longer to completely finish. Here, we see one of the costs of maintaining both environments in a single desktop computer and permitting the isolation of them. In the next section we explore this subject in more detail.

18.5.2 Energy Consumption Rate for a Single Desktop Machine

Several tests were made to measure the energy consumption function of a single machine. To measure the consumed energy, we used a HAMEG HM8115-2

TABLE 18.6. Energy Consumption versus CPU Usage

CPU Usage	Mean energy Consumption Rate	Standard Deviation
1	47.089	2601
5	55.175	3.773
10	67.979	1.359
15	74.579	1.078
20	72.356	0.565
26	73.637	2.230
30	77.581	2.682
35	81.247	2.691
40	86.493	1.862
45	85.597	2.095
50	85.272	0.930
55	86.866	1.735
61	88.014	1.196
65	91.668	1.106
70	91.510	1.240
75	92.881	0.384
81	95.052	0.816
85	96.108	1.120
90	96.600	0.791
100	96.752	0.115

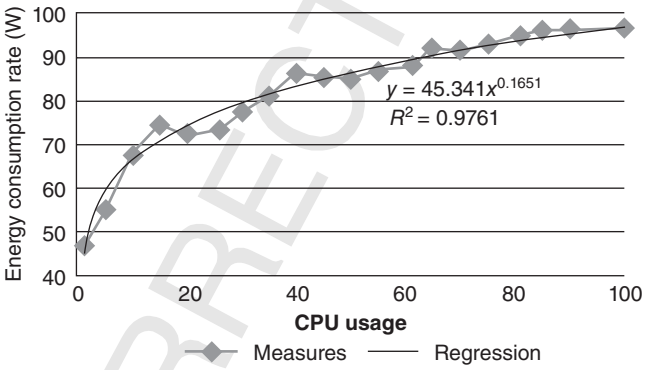

FIGURE 18.4. *Energy consumption rate versus CPU usage.*

wattmeter connected to a computer. We used a computer lab desktop machine and ran a CPU-intensive application that allows us to obtain a stable CPU usage for our tests. For each test case, we measured the energy consumption rate of the computer each second on a lapse of 47 seconds. With these data, we obtained an energy consumption rate profile for a physical machine executing CPU-intensive applications. The results are shown in Table 18.6.

To define a function $f(x)$ that returns the energy consumption rate (ECR) of a desktop given its CPU usage, we calculate a regression with the data of Table 18.6. Figure 18.4 shows the regression and data used to predict the energy consumption rate of a machine based on CPU usage.

Taking into account the above results, each computer state of a physical machine without executing a virtual machine has the energy consumption rates and processor

TABLE 18.7. Experimental Results of ECR Used by a Physical Machine without a VM

Computer State	Energy Consumption Rate (W) without VM	Mean Processor Usage (%)
1 (idle)	47	0
2 (busy)	87	10
3 (hibernation)	3	0
4 (turned off)	0	0

TABLE 18.8. Experimental Results of ECR Used by a Physical Machine Running a VM

Computer State	Mean Energy Consumption Rate with VM (W)	Mean Processor Usage (%)
1 (idle)	96	100
2 (busy)	116	100
3 (hibernation)	93	100
4 (turned off)	96	0100

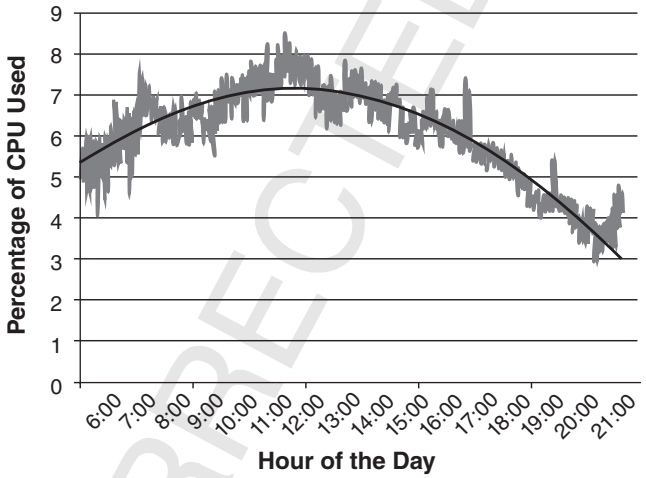

FIGURE 18.5. *CPU usage per hour.*

usage presented in Table 18.7. The mean percentage usage when there is a user using a desktop machine (P_{user}) was calculated analyzing the CPU consumption of a computer lab (70 computers) during 1 month; Figure 18.5 shows the average CPU usage during daylight working hours where the computer laboratories are open to students. On the average, CPU utilization does not exceed 10%; this value is consistent with different works on this same subject [34, 35].

In Table 18.8, we present the ECR of a desktop machine running an HPC virtual machine for each state.

Taking into account the results of Tables 18.3, 18.7, and 18.8, the amount of energy and the time needed to run a CPU-intensive task (which takes a time T to be executed in a dedicated computer) in a UnaGrid virtual machine executed on a desktop computer that can be in four possible states are shown in Table 18.9.

TABLE 18.9. Experimental Results of ECR Used by a Physical Machine Running a VM

Computer State	Execution Time with VM (W)	Mean ECR (W)	Energy Consumption
1 (idle)	T	49	49 W $\times$ T
2 (busy)	$\frac{10}{9}T$	29	32 W $\times$ T
3 (hibernation)	T	93	93 W $\times$ T
4 (turned off)	T	96	96 W $\times$ T

TABLE 18.10. Energy Consumption Rate When Several Virtual Machines (VMs) Are Executed per Physical Machine

VM	CPUs per VM	Total Running Jobs	CPU Usage	Mean ECR (W)
1	4	0	1	48.93
1	4	2	50	84.03
1	4	4	100	90.01
2	2	0	1	49.38
2	2	2	50	84.68
2	2	4	100	90.86
4	1	0	1	49.62
4	1	2	50	85.07
4	1	4	100	89.67

Results of Table 18.9 show that from the energy consumption point of view, the best desktop machines to deploy UnaGrid virtual machines are those being used (state 2), followed by the machines that are in idle state (state 1), hibernation (state 3), and turned off (state 4).

Because the desktop machines in the future will have more processing cores, we also executed tests to measure the energy consumption rate for a physical machine running one, two, and four virtual machines executing grid tasks. In Table 18.10, we can see, for each test, the number of running virtual machines, the number of cores assigned per virtual machine, and the number of total running jobs.

Results in Table 18.10 show that if multiple virtual machines are executed on a desktop computer, the ECR used by the grid tasks is very close to that used when a single virtual machine is executed.

18.5.3 Energy Consumption Tests for a Complete Computer Lab

These tests were executed to evaluate the energy consumption of a complete computer lab used to execute grid tasks. As a first step, we calculated the energy consumption rate of a computer lab on a typical day and compared it with the energy consumption rate when executing HPC virtual machines. Figure 18.6 shows the ECR of a computer lab with 70 physical machines in each case. As can be seen, the ECR is increased when executing HPC tasks (as expected). The baseline under the 5 kW corresponds to the actual consumption of the computer lab with no students or virtual machines using it. A global energy saving, only for using UnaGrid to support HPC tasks (independent of the desktop computers state used to execute virtual machines), is close to 55% considering that, currently, this ECR

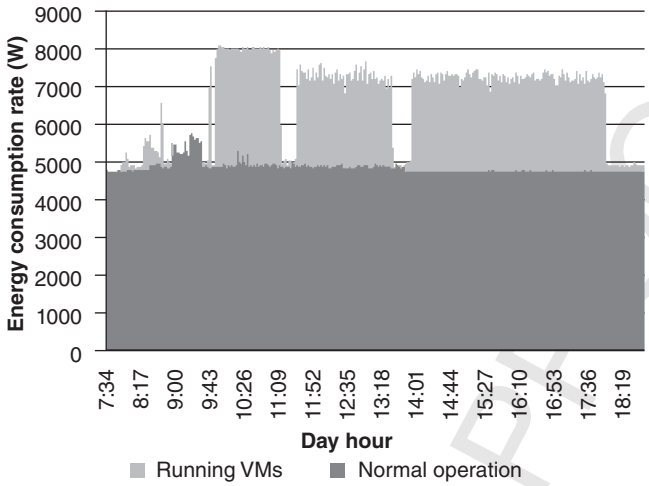

FIGURE 18.6. *Computer lab energy consumption.*

TABLE 18.11. Energy Consumption per Environment

Cluster Size (Cores)	Dedicated Cluster (kW)	User Lab (kW)	Opportunistic Grid	
			Energy	Gain
20	0.55	0.33	0.54	61
40	1.1	0.61	1.08	62
60	1.65	1.0	1.63	64
80	2.2	1.35	2.20	61
100	2.75	1.68	2.77	60
200	5.5	3.39	5.47	62
268	7.37	4.55	7.29	62

is used permanently during business hours. As we mentioned before, the grid tasks used to measure the profit of the opportunistic infrastructure consists in executing 100×100 matrix (A) multiplication tasks to compute $A_{40,000}$.

To measure all costs generated by the maintenance of an opportunistic grid, another scenario was established. The intention is to measure and compare the number of finished jobs and the amount of energy consumed to complete grid jobs. The experimental task for the scenario was the same multiple multiplication of double-precision matrices.

For this scenario, we used a computer lab with up to 268 cores. For the whole lab, we measured the average ECR when exclusively owner-users use the computers (user lab), when there are HPC virtual machines exclusively executing (dedicated cluster), and when both environments are in execution (opportunistic grid). Table 18.11 shows the detailed results. To avoid the need to reserve all computers to make dedicated tests, we reserved 10 computers and then we extrapolated the data for each cluster size. We show measures to different cluster sizes.

Based on the observations of the UnaGrid infrastructure, at least 60% of the physical machines are being used at any moment. As Table 18.11 shows, the average gain by using machines on an opportunistic model is about 61% for each test, a very significant value in terms of energy saving.

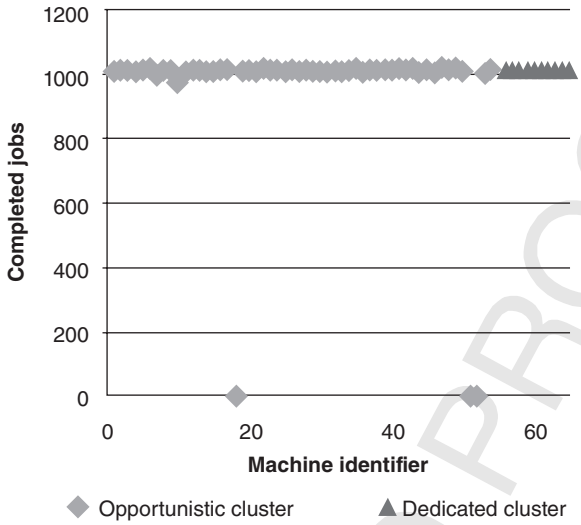

FIGURE 18.7. *Completed jobs per machine.*

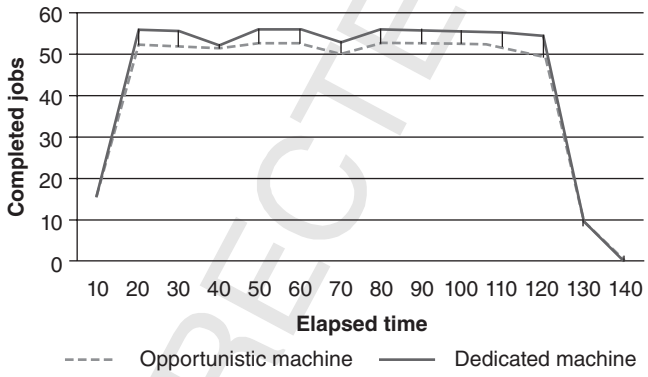

FIGURE 18.8. *Average completed jobs by cluster type.*

18.5.4 Performance Degradation Perceived by Grid Users

To test the performance degradation when grid users executed tasks on a CVC, a comparison of computational power between both environments was made by starting 40,000 jobs on 67 computers and varying the number of machines needed to perform HPC tasks during 6.2 hours. Ten of these computers had been reserved so no owner-users could use them. Figure 18.7 shows the number of completed jobs for each machine after 140 minutes of execution. As can be seen, the behavior of machines is approximately the same when they are reserved compared with when they are not. This shows that computer labs' potential is being underutilized by owner-users.

As the behavior of machines in both opportunistic and dedicated clusters is fairly equal, we calculated the average behavior of each environment using a smaller scale. Figure 18.8 shows in a detailed way the behavior of each environment under the test. The figure shows the number of completed jobs on lapses of 10 minutes. From

this figure, we see that a dedicated machine can complete on average 6% more jobs than an opportunistic one, validating our original hypothesis.

18.6 CONCLUSIONS AND FUTURE WORK

Opportunistic grids are a good alternative to develop HPC. In this work, we presented an opportunistic grid infrastructure that uses virtualization technologies to provide complete isolation between both environments: grid computing and owner-user ones. The results showed that giving low priority to the CVCs does not disturb the owner-users, who were targeted to reap the profits of the technology investments originally. Another important aspect showed in this work is that virtualization allows not only an efficient computational environment but also an energy-efficient and scalable opportunistic environment [36]. Using existing resources, industries can reduce technology investments, and the isolation of both environments allows reduction of the energy costs. When a CVC needs to be initiated, two approaches are possible: (1) Choose the idle physical machines and (2) choose physical machines where there are users working [37]. No matter which strategy is used, experimental results show that opportunistic environments offer better gains in terms of energy consumption. Comparing the two approaches presented, the gain percentage of energy saving using busy desktop computers was always greater than the other approaches. It is important to show some observations regarding the number of people using the computer labs on a daily basis. Almost 60% of the physical machines are being used at any moment, which gives a considerable saving percentage in energy consumption. Owner-users use a maximum of 10% of the CPU, so the UnaGrid infrastructure can take advantage permanently of most of the CPU capabilities available in computer labs.

REFERENCES

[1] TOP500, "TOP500 Supercomputer Series," Available at http://www.top500.org/list/2009/06/100. [Online] June 2009.

[2] F. Gilles, *Recent Advances and Research Challenges in Desktop Grid and Volunteer Computing.* [book auth.] Core GRID. "Grids, P2p and Services Computing." London: Springer, 2010.

[3] A.S. Tanenbaum and M.V. Steen, *Distributed Systems: Principles and Paradigms*, 1st ed. Upper Saddle River, NJ: Prentice Hall PTR, 2001.

[4] B.C. Neuman, "Scale in distributed systems," in *Readings in Distributed Computing Systems* (•• ••, ed.), ••: ••, pp. 50–62, 1994.

[5] J.F. Shoch, J.A. Hupp, and Xerox Corporation Palo Alto Research Center, "The 'Worm' programs early experience with a distributed computation," *Communications of the ACM*, 25:••–••, 1982.

[6] M. Litzkow, M. Livny, and M. Mutka, "Condor—A hunter of idle workstations" in *8th IEEE International Conference on Distributed Computing Systems*, pp. 104–111, San Jose, CA, 1988.

[7] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home an experiment in public-resource computing," *Communications of the ACM*, 45:56–61, 2002.

[8] D. Anderson, "BOINC: A system for public-resource computing and storage," in *5th IEEE/ACM International Workshop on Grid*, pp. 4–10, Pittsburgh: ACM, 2004.

[9] L.F.G. Sarmenta, S.J.V. Chua, R.J.V. Chua, P. Echevarria, J.M. Mendoza, R.-R. Santos, A. de Manila, S. Tan, and R.P. Lozada, "Bayanihan computing. NET: Grid computing with XML web services," in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, p. 434, 10.1109/CCGRID.2002.1017182. Berlin: IEEE/ACM, 2002.

[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A computation management agent for multi-institutional grids," in *10th IEEE International Symposium on High Performance Distributed Computing*, pp. 55–63, San Francisco: IEEE, August 2001.

[11] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G.C. Bezerra, "InteGrade: Object-oriented grid middleware leveraging the idle computing power of desktop machines," *Concurrency and Computation: Practice and Experience*, 16:449–459, 2004.

[12] C. Germain, V. Nri, G. Fedak, and F. Cappello, "XtremWeb: Building an experimental platform for global computing," *Lecture Notes in Computer Science*, 1971/2000:107–129, 2000.

[13] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg, "OurGrid: An approach to easily assemble grids with equitable resource sharing," *Lecture Notes in Computer Science*, 2862/2003:61–86, 2003.

[14] C. Anglano, M. Canonico, and M. Guazzone, "The ShareGrid peer-to-peer desktop grid: Infrastructure, applications, and performance evaluation," *Journal of Grid Computing*, 8:543–570, 2010.

[15] P. Buncic, C. Aguado Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato, and Y. Yao, "CernVM a virtual software appliance for LHC applications. Buncic," *Journal of Physics*, 219, 2010. DOI:10.1088/1742-6596/219/4/042003.

[16] J.G. Pedersen and C.U. Sttrup, "*Developing Distributed Computing Solutions Combining Grid Computing and Public Computing,*" *FatBat Software*. Available at http://www.fatbat.dk/?content=thesis.new. Accessed January 12, 2010.

[17] D. Wang and G. Bin, "SUCSI: A light-weight desktop grid system using virtualization for application sandboxing," in *International Conference on Network Computing and Information Security (NCIS)*, pp. 352–356, IEEE, 2011.

[18] H. Castro, E. Rosales, M. Villamizar, and A. Miller, "UnaGrid—On demand opportunistic desktop grid," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 661–666, Melbourne: IEEE, June 2010.

[19] R. Aversa, M. Avvenuti, A. Cuomo, B. Di Martino, G. Di Modica, S. Distefano, A. Puliafito, M. Rak, O. Tomarchio, A. Vecchio, S. Venticinque, U. Villano, M. Guarracino, F. Vivien, J. Trff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knpfer, and M. Alexander, "The Cloud@Home project: Towards a new enhanced computing paradigm," in *Euro-Par 2010 Parallel Processing Workshops*, Vol. 6586, pp. 555–562, Springer, 2010.

[20] J. Sahoo, S. Mohapatra, R.B. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Second International Conference on Computer and Network Technology (ICCNT)*, pp. 222–226, IEEE, 2010.

[21] D. Kliazovich, P. Bouvry, Y. Audzevich, and S.U. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," 53rd IEEE Global Communications Conference (Globecom), Miami, FL, December 2010.

[22] J.P. Walters, V. Chaudhary, M. Cha, S. Guercio, Jr., and G. Steve, "A comparison of virtualization technologies for HPC," in *22nd International Conference on Advanced Information Networking and Applications*, pp. 861–868, New York: IEEE Press, 2008.

[23] G.L. Valentini, W. Lassonde, S.U. Khan, N. Min-Allah, S.A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A.Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J.E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, ••:••–••, ••, forthcoming.

[24] VMware, Inc., *VMware Workstation*. Available at http://www.vmware.com/products/workstation/, 2011. Accessed July 31, 2011.

[25] AMD, "AMD Virtualization (AMD-V) Technology." Available at http://sites.amd.com/us/business/it-solutions/virtualization/Pages/amd-v.aspx. [Online] July 2011.

[26] R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, F.H. Leung, and L. Smith, "Intel virtualization technology," *Computer*, 38:48–56, 2005.

[27] M. Pretorius, M. Ghassemian, and C. Ierotheou, "An Investigation into energy efficiency of data centre virtualisation," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 157–163, IEEE, 2010.

[28] X. Liao, H. Liting, and H. Jin, "Energy optimization schemes in cluster with virtual machines," *Cluster Computing*, 13:116–126, 2010.

[29] University of Liverpool, "Condor High Throughput Computing." Available at http://www.liv.ac.uk/csd/escience/condor/power_save.pdf. [Online] April 2010.

[30] B. Schott and •• Emmen, "Green methodologies in desktop-grid," in *International Multiconference on Computer Science and Information Technology (IMCSIT)*, pp. 671–676, IEEE, 2010.

[31] A.M. Vargas, L.M. Quesada Ocampo, M.C. Cspedes, N. Carreo, A. Gonzlez, A. Rojas, A.P. Zuluaga, K. Myers, W.E. Fry, P. Jimnez, A.J. Bernal, and S. Restrepo, "Characterization of *Phytophthora infestans* Populations in Colombia," First Report of the A2 Mating Type." pp. 82–88. DOI:10.1094/PHYTO-99-1-0082. Bogot: s.n., 2009.

[32] A. Gonzalez, H. Castro, M. Villamizar, N. Cuervo, G. Lozano, S. Restrepo, and S. Orduz, "Mesoscale modeling of the *Bacillus thuringiensis* sporulation network based on stochastic kinetics and its application for in silico scale-down," in *International Workshop on igh Performance Computational Systems Biology*," pp. 3–12, IEEE, HIBI '09, 2009.

[33] A. Bernal, M.A. Ramirez, H. Castro, J.L. Walteros, and A.L. Medaglia, "JG2A: A grid-enabled object-oriented framework for developing genetic algorithms," in *IEEE Systems and Information Engineering Design Symposium SIEDS'09*," pp. 67–72, Virginia: IEEE, 2009.

[34] D. Kondo, G. Fedak, F. Cappello, A.A. Chien, and H. Casanova, "Characterizing resource availability in enterprise desktop grids," *Future Generation Computer Systems—FGCS*, 23:888–903, 2007.

[35] P. Domingues, P. Marques, and L. Silva, "Resource usage of Windows computer laboratories," in *International Conference Workshops on Parallel Processing (ICPP)*, pp. 469–476, IEEE, 2005.

[36] F. Pinel, J.E. Pecero, P. Bouvry, and S.U. Khan, "A two-phase heuristic for the scheduling of independent tasks on computational grids," ACM/IEEE/IFIP International Conference on High Performance Computing and Simulation (HPCS), Istanbul, Turkey, July 2011.

[37] S.U. Khan and I. Ahmad, "Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation," 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece, April 2006.

[38] Mersenne Research, Inc., "GIMPS: 'Great Internet Mersenne Prime'." Available at http://www.mersenne.org/. [Online] January 2010. Accessed June 15, 2009.

[39] Distributed.Net, "Distributed.net FAQ-O-Matic," Available at http://www.distributed.net. [Online] July 2011. Accessed June 17, 2009.

[40] J. Kolodziej, S.U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids," 6th IEEE International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing (3PGCIC), Barcelona, Spain, October 2011.

[41] S.U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, 20(3):346–360, 2009.

[42] S.U. Khan, "A goal programming approach for the joint optimization of energy consumption and response time in computational grids," in *28th IEEE International Performance Computing and Communications Conference (IPCCC)*, pp. 410–417, Phoenix, AZ, December 2009.

[43] L. Wang and S.U. Khan, "Review of performance metrics for green data centers: A taxonomy study," *Journal of Supercomputing*, ••:••–••, ••, forthcoming.

[44] S. Zeadally, S.U. Khan, and N. Chilamkurti, "Energy-efficient networking: Past, present, and future," *Journal of Supercomputing*, ••:••–••, ••, forthcoming.

[45] M.A. Aziz, S.U. Khan, T. Loukopoulos, P. Bouvry, H. Li, and J. Li, "An overview of achieving energy efficiency in on-chip networks," *International Journal of Communication Networks and Distributed Systems*, 5(4):444–458, 2010.