








Towards Balancing Energy Savings and Performance for Volunteer Computing through Virtualized Approach

Fábio Rossi¹, Tiago Ferreto², Marcelo Conterato², Paulo Souza²,
Wagner Marques², Rodrigo Calheiros³ and Guilherme Rodrigues⁴

¹Federal Institute of Education, Science and Technology Farroupilha, Alegrete, Brazil

²Polytechnic School, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil

³Western Sydney University, Parramatta, Australia

⁴Federal Institute of Education, Science and Technology Sul Rio-grandense, Charqueadas, Brazil

Keywords: Energy-efficient, Grid Computing, Performance-aware, Virtualization, Volunteer Computing.


Abstract: Computational grids consist of distributed environments where partner institutions offer hosts along with computational resources that can be used by all members of the grid. When an application needs to run in such environment, it allocates a portion of hosts necessary for its executions. Traditionally, the workload imposed on computational grids has a characteristic of being bag-of-tasks (BoT). It means that multiple replicas are submitted to different hosts, and when a response is processed, such replicas are either ignored or released. On resource allocation, as the grid is distributed among different participants, only idle resources can be leased by a new application. However, due to the behavior of BoTs, many allocated resources do not use their resources in their entirety. Another important fact is that only fully idle hosts can be added to the grid pool, and used only at these times. From the above, this paper proposes an approach that uses underutilized resource slice of grid hosts through virtualization, adjusting the use of grid applications to the leftover resources from daily hosts usage. It allows grid applications to run, even when hosts are in use, as long as there is an idle slice of resources, and their use does not interfere with the host's current running. To evaluate this approach, we performed an empirical evaluation of a virtualized server running applications concurrently with a virtualized grid application. The results have showed that our scheme could accelerate the performance of grid applications without impacting on higher energy consumption.


1 INTRODUCTION


Usually, the volunteer computing (Nov et al., 2010) is a distributed computing system often associated with significant scientific projects, which uses the power of idle machines around the world to process substantial amounts of information. The workload is divided and sent to different hosts, and them can handle such workload simultaneously, reducing costs and the time


spent on studies and research as well as providing a way to enhance the use of resources. Such hosts use client software that connects it to a distributed and heterogeneous computational grid, creating a highly-scalable environment. With the increased processing power of servers, computational resources are getting underutilized. The most of the user applications are not parallel enough to take advantage of these resources, such as sequential applications on multicore processors.


Due to this fact, although there are applications that make intensive usage of resources (Hwang et al., 2003), this is not the reality of the most of today's grid computing environment. It means that although the hosts are running an application, there is a significant slice of idle resources available for use. It seems to be a waste of both resources and power, and it would


^a  <https://orcid.org/0000-0002-2450-1024>


^b  <https://orcid.org/0000-0001-8485-529X>

^c  <https://orcid.org/0000-0001-7782-812X>

^d  <https://orcid.org/0000-0003-4945-3329>

^e  <https://orcid.org/0000-0003-3304-5611>

^f  <https://orcid.org/0000-0001-7435-2445>

^g  <https://orcid.org/0000-0002-0389-2900>

be interesting to use for volunteer computing that part unused of the idle user, concomitantly. An important issue is that such a technology that uses idle resources does not interfere with all other user applications.

One of the current technologies that enable to use that slice of idle processing, or even allocate slices of available resources, increasing or decreasing depending on the demand in each slice, is the virtualization (Uhlig et al., 2005). Virtualization enables to run independent operating systems or applications, concurrently, on the same resources. The advantage of this approach compared with the traditional architecture of volunteer computing is that there is no interruption in grid service. Although the slices of available resources used by the virtual machine vary over the time, this new proposal could increase the execution time of grid applications.

However, a significant resource usage also increases the power consumption. Currently, there are several papers studying trade-offs between performance and power consumption such as (Siam et al., 2010). It is a growing concern, given that the physical limits of silicon have been achieved (Bohr, 1998). This paper combines some technologies that can reduce such trade-off. The big question answered by this paper, in addition to measuring the performance of a voluntary virtualized computing environment, is to check how much energy savings this proposal promotes. Moreover, this paper shows that the balance between most magnificent performance (execution time of the grid applications) and energy-saving, exposing this trade-off.

This paper provides two classical problems solutions of volunteer computing:

- The first refers to the increase in energy consumption. The idle processor usually has lower power consumption than when it is active. A participant may leave the grid host connected for 24 hours, and disable the power saving features, such as suspension. Also, if adequate cooling is not in place, this constant load at host volunteer can cause overheating.
- The second problem is the decreased performance of the host. If the application tries to execute volunteer computing while the computer is in use, it can affect both host and grid application performance. It is due to the increased processor, cache, disk I/O, and network I/O contentions.

Even high-performance applications do not use all available features at all times. In these environments, there are some moments of idleness or some idle slice of resources. With this in mind, our proposal analyzes whether a virtualized grid application could be

used for these slices, whether in an HPC, cloud or fog computing environment, in order to always keep grid application running, but without interfering with the performance of other applications competing with the environment. In this way, the primary goals of this paper are:

- Verify if the virtualization layer isolates the virtualized application from the user applications;
- Test different resources usage rates for virtualized grid applications;
- Propose a new approach allowing flexibility of grid applications;
- Find the limits of the trade-off between performance and power consumption.

This paper is organized as follow: in Section 2, it describes the concepts of volunteer computation and virtualization. Section 3 presents some related works. Section 4 presents the environment used as a testbed along with its features. Section 5 shows evaluations and discussion about the outcomes and; finally, Section 6 shows the conclusions and future work.

2 BACKGROUND

This section presents general concepts of volunteer computing and virtualization, a suite of technologies studied and assessed in this paper.

2.1 Volunteer Computing

Volunteer computing (Nov et al., 2010) has gained increasing importance because of the computing capacity currently available on personal computers. For many scientific works, it is estimated that this strategy can complement (or even replace) the need for significant investments in high-performance computing infrastructures.

This approach is based on voluntary work, where computational and storage resources are ceded to research projects at no cost. This technique allows the participation of citizens in scientific research in direct and real-time, offering the time of processing of their computers for computational computation of scientific interest by distributed computing techniques.

In this way, volunteer computing aims to offer for large projects, the ability to idle servers and personal computers around the world to process significant amounts of information, thereby reducing the processing time of a given experiment. Anyone can be a volunteer. It is necessary to install client software on the computer and sign up for some project.

This software is responsible for communication with the personal computer on the Internet server of the project and the scheduling of tasks in idle cycles of personal computing.

Projects that would take thousands of years to process the information has its execution time reduced considerably using the volunteer computing through the problem division in millions of small units that can be processed in parallel and distributed architecture. The central server sends small data packets to idle computers registered in the system. Although some of these projects for common causes, requiring rugged computers of large research centers, most of them are satisfied with personal computers, either at home or work, it has to offer.

Thus, these data are processing by the personal computers, obtaining results that are returned through the Internet to a central server. Such data are received by the server, logged, analyzed, and the process starts again, until the entire workload completion. From the above, a very popular model in this type of environment consists of bag-of-tasks, i.e., a chunk of the problem to be solved is sent to several workers in the grid, who process such chunk in idle cycles and return the processing result to the grid Servers.

2.2 Virtualization

In the past few years, processing capacity in computers has considerably increased, even though there was a significant lack of its usage. There are situations where applications could be executed more efficiently, improving the processor. One of the solutions for that is the use of virtualization (Uhlig et al., 2005), which has been applied to satisfactory outcomes more often.

Considering virtualization, it can be done to dissociate the hardware to the operating system, bringing new and useful tools. Virtualization allows the user to control processor, memory, storage and other guest's operating system resources (virtual machines); thus, each guest system receives the necessary amount of resources. This control eliminates the risk of a process that uses all available memory or all processor load.

This type of flexibility changes the traditional concept of server usage and capacity planning. In virtualized environments, it is possible to deal with computational resources such as the processor, memory, and storage as a cache of resources and applications that might be quickly reallocated to receive new resources when necessary. But virtualization is not a new concept, in the 60's it was already applied to mainframes. Currently, with the processing power ad-

vance of desktop computers, different models of virtual machines have been developed, and its use has become widespread with excellent results.

Therefore, virtualization allows executing multiple operating systems on the same computer. It is possible with the use of specific programs that creates virtual machines, emulating the physical components of a host.

The isolation capacity of virtualization is one of the leading factors to support this work. This isolation is done through a technique called compression ring (Figure 1. Most of processors have four levels of priority for code execution, numbered 0-3. Code running on level 0 can execute any instruction on the CPU, while at level 3 (the more restricted) some instructions cannot be performed. These priority levels earned the name rings because of the way they were illustrated in the chip programming 80386 manual. It also relates to the isolation performance, meaning that a virtual machine can be isolated from the slice of resources allocated to the host operating system or other virtual machines hosted on the same physical host.

The virtual memory manager (VMM or virtual machine manager) run inside the virtual host or non-system kernel, and when a virtual machine is created, it moves the virtual kernel to run at level 1 instead of level 0. The virtual kernel thinks it is operating at level 0 but is running at level 1, and this allows the VMM to monitor the execution of the virtual machine and manages access to memory and peripherals, and eventually emulate software instructions that can only be called the real level 0.

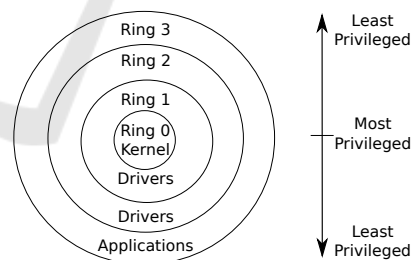


Figure 1: Virtualization Rings.

3 RELATED WORK

Virtualization provides the computing environment capabilities such as elasticity, resource isolation, better use of available resources, security and software support legacy. Such capabilities attract the interest of several distributed systems models that seek heterogeneity and scalability. According, several studies using volunteer computing along with virtualization technology, such as (Ben Belgacem et al., 2012).

Ferreira et al. (Ferreira et al., 2011) show the virtualization technology as a sandbox for security applications to BOINC. Besides, they created a middleware called libboincexec, allowing BOINC to run optimally on several virtual machine monitors. The authors claim that use BOINC in a virtualized environment can provide security against forged answers arising from grid customers. Based on portability offered at work, Theodoropoulos et al. (Theodoropoulos et al., 2016) propose to use the same platform on a cluster, with the objective of improving the performance of grid applications. Due to the high scalability of virtualized cluster environments, the results showed gains of up to 13%. The first solution focuses only on security issues due to the virtual isolation capabilities, but not taking into account energy efficiency or application performance. The second work was developed for high-performance applications, which led the focus of the work to improve the performance of applications. However, the proposal probably increased energy consumption, since it has seen the more excellent use of resources and the behavior of the applications used in such work.

Cavalcanti et al. (Cavalcanti et al., 2006) use virtual machines to meet the need for security in file sharing. For this, the study used Xen to offer the safety of a safely distributed file system to the grid environment. Brasileiro et al. (Brasileiro et al., 2007) used the previously proposed model to strengthen a scalable peer-to-peer grid environment called OurGrid. Such an environment has become one of the most widely used distributed grid platforms, including serving as a model for well-known grid simulators such as SimGrid (Brennand et al., 2016). Unlike Ferreira et al. (Ferreira et al., 2011), Cavalcanti et al. (Cavalcanti et al., 2006) used the resource allocation capability provided by the virtualization layer to tune virtual machines in a distributed, heterogeneous environment in order to increase the performance of grid applications. On top of such proposal, Brasileiro et al. (Brasileiro et al., 2007) proposed Ourgrid, one of the most used grid environments, which allows the processing of a large volume of data in a distributed way, focusing on the performance of applications.

Cunsolo et al. (Cunsolo et al., 2009) present the concept of a grid-as-a-service. This new paradigm is compared with state of the art and discussed as a viable proposal being implemented. The work addresses the heterogeneity and independence of cloud hosts as being an environment that can also be exploited by the grid paradigm, although it was not designed for this purpose. In such proposal, grid applications are placed on an already established cloud environment, and it uses the characteristics of that en-

vironment to achieve performance and energy saving metrics. In contrast, our proposal has a focus from conception to development of addressing such metrics.

Jonathan et al. (Jonathan et al., 2017) bring this concept to edge devices. The main idea is to take the application to a location closer to the consumer, but reliably and dynamically replicate data to achieve timeliness for computation and high data availability for data storage respectively. Again, the work presents a scenario that has been previously established and performs a grid environment on top of it with the intent of leveraging all of its infrastructure capabilities.

Table 1 summarizes the related work. As we can see and to the best of our knowledge, no previous work has used virtualization technology to fit the slices of idle resources on hosts in the same way we are (enabling voluntary computing to keep running even when a machine is in use). Besides, no other work has attempted to improve the trade-off between performance of grid applications and energy savings.

4 ENVIRONMENTAL TEST

This section shows the technologies used to this work: grid software, virtualization platform, and infrastructure.

4.1 BOINC

The grid software used in this work was the BOINC (Anderson, 2004). The motivation for creating this program was the SETI project (Search for Extraterrestrial Intelligence) abandonment by NASA in the mid-70, due to its controversial proposals is quite critical. By 1990, they were taken by an institute called SETI League, which together with the University of Berkeley launched the scientific program called SERENDIP, which applied the distributed computing via home computers with the SETI@Home.

In 1992, researchers at UC Berkeley developed the BOINC software, originally designed to ease connect the home computers to this program. The success of this project made it clear that distributed computing could be utilized for many other scientific projects, which demand high processing capabilities, and then, several new projects were created. The BOINC client software is highly customizable, and it allows to choose which projects will help and the number of resources that will be available. BOINC application does automatic updates of the applications of projects and downloads new jobs to be processed. Therefore,

Table 1: Related work comparison.

Work	Energy Efficient	Performance Aware
Ferreira et al. (Ferreira et al., 2011)	-	-
Theodoropoulos et al. (Theodoropoulos et al., 2016)	-	x
Cavalcanti et al. (Cavalcanti et al., 2006)	-	-
Brasileiro et al. (Brasileiro et al., 2007)	-	x
Cunsolo et al. (Cunsolo et al., 2009)	-	-
Jonathan et al. (Jonathan et al., 2017)	-	-
This proposal	x	x

volunteer computing uses idle processor cycles, but when the user is using such resources, these resources are not used in its entirety. BOINC allows to be configured for low usage, that is, it can remain active during the period of user usage. However, the resource utilization rate is always fixed (for example, BOINC will still use 10% of available resources). The technology that fluctuates in resource utilization rate depending on user usage is called virtualization.

4.2 VirtualBox

VirtualBox (Watson, 2008) is a virtual machine monitor developed by Oracle that aims to create environments for installation of different systems. It allows the installation and use of one operating system within another, as well as their respective software, such as two or more independent computers, but physically sharing the same hardware.

It was created by the company called Innotek, which initially offered a proprietary license, but there was also a version of the product for personal use or evaluation at no cost. In January 2007, the release of the VirtualBox OSE (Open Source Edition) with the GNU General Public License (GPL) - Version 2 was released. In February 2008, the company Innotek was acquired by Sun Microsystems. In April 2009, Oracle purchased Sun Microsystems and all of its products, including VirtualBox.

VirtualBox has an extremely modular design with well-defined internal programming interfaces and a client/server design. It makes it easy to control multiple interfaces at once. For example, the user can start a virtual machine on a typical virtual GUI machine and then control that machine from a command line, or possibly remotely. A host operating system maintains the communication between the virtual machines monitor and hardware.

Such virtual machine monitor also offers a complete software development kit. Although it is open source, the user does not need to rewrite new access interfaces. The configuration settings for virtual machines are stored in XML and are entirely independent of local hosts. Therefore, the settings can easily be transferred to other computers.

Besides, VirtualBox has special software that can be installed on Windows and Linux virtual machines

to improve performance and make integration much more seamless. Like many other virtualization solutions, to facilitate the exchange of data between hosts and guests, VirtualBox allows the declaration of the directories of specific hosts as shared folders which can be accessed from within virtual machines.

The characteristics of modularity and flexibility, as well as the management of the slices of resources provided by VirtualBox, boosted its choice for the development of this work.

4.3 Testbed

A traditional volunteer computing architecture is presented in Figure 2. In a few words, there are four personal computers connected to a server, forming a computational grid. Furthermore, the Machine 1 is with a stream running in user space (the ray represents an execution flow) which means that this machine is not available at this time for grid applications. Machines 2, 3, 4 does not support any flow in user space, allowing the grid to keep streams running on their resources (the execution flow within the BOINC space).

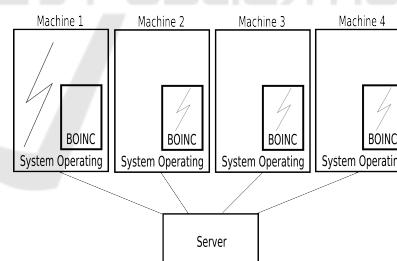


Figure 2: Traditional Grid Environment.

The primary issue discussed in this paper is that even if the user is using hosts, their resources are being underutilized because of the massive processing power of current hosts.

Therefore, it proposes a new environment that can be seen in Figure 3, which uses virtualization technology always to keep the grid execution, even if the user is using the host. Virtualization can adapt to the share of resources that other applications are not using, which means that will not overheat the user performance and its use.

Evaluations were performed on a client-server architecture when several clients are accessing the grid

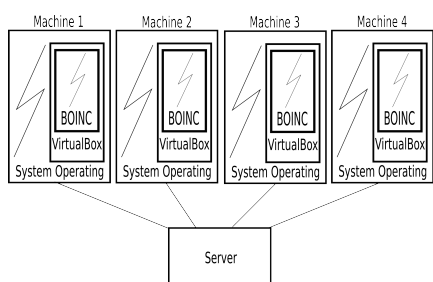


Figure 3: Virtualized Grid Environment.

virtualized application in a server host connected to a Gigabit Ethernet network. The servers used in the experiments are composed of four processors with 2 Intel Xeon E5520 (16 logical cores in total), 2.27GHz, 16 Gb RAM.

The energy consumption is acquired by using a multimeter which is connected to the power source and the machine. This device (EZ-735 digital multimeter) has a USB connection that allows external periodically reading and gives the values of power consumption in watts-per-hour.

The tests were conducted with one of the standard set of tests by BOINC leading a mean of 24 hours to perform all the processing in a real environment, in which the host was running with jobs over a period of 8 hours and idle for 16 hours. The test chosen was the Collatz Conjecture (also known as $3n+1$), a test based on mathematical Internet connections. The conjecture gives a rule stating that any natural number, as applied to this theory, in the end, will always be 1. The method applies to any natural number and tells if this number is even, divide it by 2 and if it is odd, to multiply by 3 and add 1. The test used took 40 hours to be completed.

This scenario is quite realistic, if it is compared with a production environment where the host is used during the period of a common laborer, and the application of grid uses idle time outside these 8 hours. With this test scenario using the proposed virtualized environment, it was possible to develop several examples with different settings, with the rate limitation of the virtual machine usage between 10% and 90% (regarding about the proposed slices discussed earlier).

Thus, the evaluations are from an environment with 90% of usage by user and the other 10% utilized by the virtual environment, and BOINC up to 10% of usage by user and the additional 90% used by the virtual environment, and BOINC. The virtual machine monitor used was the VirtualBox (Romero, 2010).

5 EVALUATION AND DISCUSSION

This section presents the environmental assessments of proposed metrics relating execution time and power consumption. The first evaluation shows the overhead of the virtualized environment on the grid application.

Figure 4 illustrates the difference in the throughput (the amount of data transferred from one place to another, or the amount of data processed in a given time) between the Native Linux environment and the virtualized environment when we use the BOINC within virtual machines. As the grid application receives a bag-of-tasks, processes, and returns the response to the server, it is possible to measure the throughput of both ways when packets are sent, as when the server receives packets. This difference is due to the processing network which keeps the virtualization layer. As the hypervisor interprets all communication, there is a delay of up to 25%. This overhead is not significant when compared to the other benefits that virtualization brings this new approach.

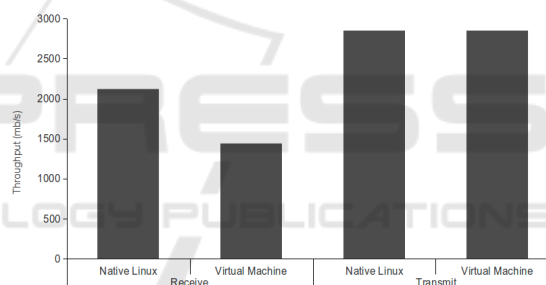


Figure 4: Throughput.

Another essential factor to be evaluated is the application's execution time. Figure 5 shows the results of the use of the virtual environment that keeps running BOINC and the execution time of each test case. In all cases tested there is an improvement in execution time. The running time in the worst case is quite close to the base test that took 40 hours, but still better results. This execution time has the more significant impact, the closer the values on which there is a higher use of the virtual machine, and consequently, the higher bandwidth available to the virtual machine.

The ability that makes this possible is the isolation of the virtual machines, which do not exceed the limits established of use rates, not interfere with the user applications performance.

This isolation is achieved by a technique called compression ring. The standard x86 processors have four levels of priority for code execution, numbered from 0 to 3. Code running at level 0 can execute any instruction on the CPU, while at level 3, there are in-

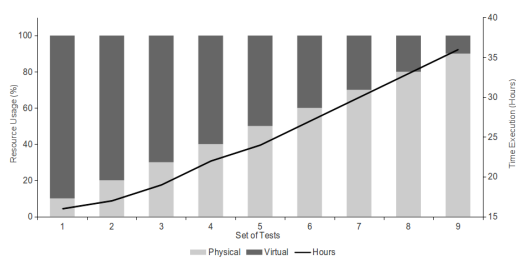


Figure 5: Time Execution.

structions that cannot be performed.

The virtual memory manager runs inside the kernel of the host system or non-virtual, and when it creates a virtual machine, it moves the virtual kernel system to run on level 1 instead of level 0. The kernel of the virtual machine believes it is operating at level 0 but is running at level 1, and this allows the virtual memory manager to monitor the execution of the virtual machine and manage access to memory and peripherals, eventually emulate in software instructions that only can be called the real level 0.

Thus, the isolation between the different levels of protection, guaranteed by the architecture of the chip, prevents virtual machine instructions compromise the host system kernel. As for the power consumption of this proposal, it is possible to see that there is energy-saving up to a certain limit. To compare the energy consumption of this proposal, we used as the base value, a test with the same load but in a native environment without virtualization, as can be seen in Table 2.

Table 2: Base value of power consumption on a real environment without virtualization. R-V: balancing the use of resources between real and virtual; ET: application execution time; PC: power consumption during the test.

R-V	ET	PC
100-0	40	7040

In the Table 3, the values are related to rates of resource use between user applications and the virtual machine with the grid application, the execution time in hours for each use threshold, and the average energy consumption per host in watts.

Table 3: Power consumption on a real-virtual environment. R-V: balancing the use of resources between real and virtual; ET: application execution time; PC: power consumption during the test.

R-V	ET	PC
10-90	16	4640
20-80	17	4930
30-70	19	5510
40-60	22	6380
50-50	24	6960
60-40	27	7830
70-30	30	8700
80-20	33	9570
90-10	36	10440

Up to a limit about 50% of use between user applications and virtualized environment, there are energy-savings. It is associated with shorter execution time of the application grid, which although having the highest power consumption during the execution, the execution time is reduced as to save energy. Above this value, although the execution time still worthwhile when compared to the tests in the native environment, the energy consumption increases significantly, which not allowed its use when there is concern about energy-saving. Perhaps it is not come to be a significant limitation, given the computers that are currently available for computational grids, such as personal computers, the use of resources of most users usually do not reach 50%.

6 CONCLUSION AND FUTURE WORK

Virtualization is the technique of running one or more virtual servers on one physical server. It allows a higher density of use of resources (hardware, storage, etc.) while allowing isolation and security are maintained. Based on these features, virtualization has become the basis for the use of other technologies that wish to use their abilities. One of the technologies that have turned their attention to this new infrastructure is the volunteer computing.

In this sense, virtualized computing enables a highly heterogeneous environment to present homogeneity to the top software layers, making grid clients less coupled and more dynamic. Focusing primarily on the virtualization characteristic regarding resource isolation, this paper proposes the use of virtualization technology along the volunteer computing, allowing this type of distributed processing executes concomitantly with other applications in the same environment.

The results showed that there are advantages of this approach, in which no influence on user applications, while there is an improvement of the grid applications performance, considering the best use of available resources, managed by virtualization. Furthermore, the results show that there is energy-saving up to 50% use of the virtualized resources, which validates this proposal, taking into account the use of resources by ordinary users do not exceed this limit. There are virtual machines with more performance, but with less isolation. Just as there are virtual machines with less performance, but with more isolation.

Based on discussed above, the following findings can be cited:

- Grid execution is kept even if the user using the host.
- Better utilization of the hosts.
- Upper limiting the proportion of resources used by grid execution ensuring the avoidance of overheating.
- Due to virtualization the volunteer client app (BOINC) is isolated from the user app.

On the results presented in this paper, several future works can be addressed. The elasticity provided by virtualization has enhanced cloud environments, so grid applications with the intent to better utilize available resources can take advantage of this new paradigm. Besides, the service-oriented architecture may be usable as it provides integration and interoperability to client applications.

REFERENCES

- Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 4–10, Washington, DC, USA. IEEE Computer Society.
- Ben Belgacem, M., Abdennadher, N., and Niinimäki, M. (2012). Virtual ez grid: A volunteer computing infrastructure for scientific medical applications. *Int. J. Handheld Comput. Res.*, 3(1):74–85.
- Bohr, M. (1998). Silicon trends and limits for advanced microprocessors. *Commun. ACM*, 41(3):80–87.
- Brasileiro, F., Araujo, E., Voorsluys, W., Oliveira, M., and Figueiredo, F. (2007). Bridging the high performance computing gap: the ourgrid experience. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, pages 817–822.
- Brennand, C. A. R. L., Duarte, J. M., and Silva, A. P. (2016). Simgrid: A simulator of network monitoring topologies for peer-to-peer based computational grids. In *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6.
- Cavalcanti, E., Assis, L., Gaudencio, M., Cirne, W., and Brasileiro, F. (2006). Sandboxing for a free-to-join grid with support for secure site-wide storage area. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, VTDC '06, pages 11–, Washington, DC, USA. IEEE Computer Society.
- Cunsolo, V. D., Distefano, S., Puliafito, A., and Scarpa, M. (2009). Cloud@home: bridging the gap between volunteer and cloud computing. In *Proceedings of the 5th international conference on Emerging intelligent computing technology and applications*, ICIC'09, pages 423–432, Berlin, Heidelberg. Springer-Verlag.
- Ferreira, D., Araujo, F., and Domingues, P. (2011). lib-boincexec: A generic virtualization approach for the boinc middleware. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 1903–1908, Washington, DC, USA. IEEE Computer Society.
- Hwang, S., Jeong, K., Im, E., Woo, C., Hahn, K.-S., Kim, M., and Lee, S. (2003). An analysis of idle cpu cycles at university computer labs. In *Proceedings of the 2003 international conference on Computational science and its applications: Part I*, ICCSA'03, pages 733–741, Berlin, Heidelberg. Springer-Verlag.
- Jonathan, A., Uluyol, M., Chandra, A., and Weissman, J. (2017). Ensuring reliability in geo-distributed edge cloud. In *2017 Resilience Week (RWS)*, pages 127–132.
- Nov, O., Anderson, D., and Arazy, O. (2010). Volunteer computing: a model of the factors determining contribution to community-based scientific research. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 741–750, New York, NY, USA. ACM.
- Romero, A. V. (2010). *VirtualBox 3.1: Beginner's Guide*. Packt Publishing.
- Siam, M. Z., Krunz, M., Cui, S., and Muqattash, A. (2010). Energy-efficient protocols for wireless networks with adaptive mimo capabilities. *Wirel. Netw.*, 16(1):199–212.
- Theodoropoulos, D., Chrysos, G., Koidis, I., Charitopoulos, G., Pissadakis, E., Varikos, A., Pneumatikatos, D., Smaragdos, G., Strydis, C., and Zervos, N. (2016). mcluster: A software framework for portable device-based volunteer computing. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 336–341.
- Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C. M., Anderson, A. V., Bennett, S. M., Kagi, A., Leung, F. H., and Smith, L. (2005). Intel virtualization technology. *Computer*, 38(5):48–56.
- Watson, J. (2008). Virtualbox: Bits and bytes masquerading as machines. *Linux J.*, 2008(166).