# Performance Evaluation of Virtual Machines in a Service-oriented Grid Testbed

Carlos R. Senna, Luiz F. Bittencourt, and Edmundo R. M. Madeira
*Institute of Computing,*
*University of Campinas - UNICAMP,*
*P.O. Box 6196, Campinas, São Paulo, Brazil*
{*crsenna, bit, edmundo*}*@ic.unicamp.br*

## ABSTRACT

*In this paper we analyze the performance of execution of service workflows in virtual machines (VMs) used as resources in our service-oriented grid testbed (SGT). We chosen an application widely known by computer network researchers and developers to evaluate the testbed: the simulation of networks using the Network Simulator. We constructed workflows which execute several WiMAX network scenarios in parallel, and we executed them to evaluate the performance of virtual and physical machines in the grid. The experiments show that the adequate support to the parallelism can minimize the consequences of the overhead introduced by the VMs, allowing the simulation of several scenarios with execution time as low as one simulation alone would take. We conclude that it is important to have a VM-aware resource manager to completely explore the hardware and its VMs.*

**KEYWORDS:** Grids, Services, Workflows, Virtual Machines.

## 1. INTRODUCTION

Grids are computational networks where heterogeneous resources can be shared. The Open Grid Services Architecture (OGSA) [1] incorporates Internet protocols to achieve interoperability between heterogeneous resources, allowing the grids to use standards and paradigms from the Service Oriented Computing (SOC) [2]. The service-oriented grid allows users to establish links between services, organizing them as workflows instead of building only sequential applications. Such service compositions require adequate support to the coordination and dynamic composition of processes and services, what is not supplied by current OGSA implementations.

Virtualization [3] presents logical virtualized resources such as CPUs, physical memory, network connections, and peripherals in form of Virtual Machines (VM). With this, each VM can have its own operating system, applications, and network services. VMs as grid resources can bring benefits and help in avoiding some restrictions of this environment. The grid can involve VMs in allocation problems, where more CPUs than the available ones are needed to execute a workflow. However, the use of VMs in the grid brings new management requirements. We are going to show experiments using our SGT to analyze the performance of VMs as computational resources in the grid. The SGT models VMs as services, as well as the physical resources, and uses them in the service workflows. We constructed several service workflows to execute different simulation scenarios of WiMAX networks using the Network Simulator 2 (NS2), and we utilize them to evaluate the performance of the VMs in relation to their host machines. Considering the virtual machine characteristics, our software infrastructure with VM-aware scheduling algorithms can optimize the resources usage in service workflows execution, notably in parallel execution scenarios.

The paper is organized as follows. In Section 2 we present the Grid with our management infrastructure, which is responsible for the workflow execution in the resources. Section 3 describes the experiments with NS2. Experimental results are shown in section 4, while Section 5 describes some related works. Section 6 concludes the paper.

## 2. THE GRID INFRASTRUCTURE

The SGT is composed of computers with a grid software toolkit and our software infrastructure. To compose the grid we join physical and virtual machines (VMs). A VM

is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. The VM implementation adds a software layer to a real machine to support the desired architecture [3]. The virtualization can provide a better utilization of powerful and/or underutilized CPUs, as well as it allows several applications, operating systems, and software platforms to be used in the same hardware. This can facilitate the development of software products to be delivered to many different platforms with no need to acquire more than one hardware. We use Xen as virtual machines in our grid testbed.

The grid infrastructure used in our SGT is the Globus Toolkit (GT) [4], an OGSA implementation [1]. In the OGSA all resources are modeled as services, bringing to the environment concepts like SOA [2]. We used the GT version 4 installed on Debian GNU/Linux operating systems over a gigabit ethernet network.

The SGT uses seven computational resources. There are three physical machines, namely Cronos, $P1$, and $P2$, and four virtual machines. Over the $P1$ physical machine we build three computational resources: Temis, Helios, and Persefone. Temis is a "Debian/Linux physical machine", while Helios and Persefone are Xen virtual machines in Temis. In $P2$ we used the same schema, with Medusa being the "physical machine", and with Eros and Caos being Xen virtual machines in Medusa. Table 1 summarizes all computational resources specifications. Every physical machine is abbreviated with its name's first letter plus a $0$, while the virtual machines had their names abbreviated after their hosts. For instance, Persefone is called $T2$ because it is a virtual machine of Temis ($T0$). We use this naming scheme to facilitate the identification virtual machines and their hosts. In the testbed, this configuration can be changed by the inclusion, exclusion, and migration of VMs among physical machines. Such dynamic characteristic is tracked by the SGT's resource monitor.

The OGSA grid allows bindings between services, organizing them as workflows instead of standalone applications. To manage workflows execution, the SGT is composed of a set of services showed in Figure 1. Its main components are the workflow manager, the scheduler, the deployment service, the resource monitor, and the repositories.

The workflow manager implements the Grid Process Orchestration (GPO) [5]. The service workflows are described in GPOL (Grid Process Orchestration Language) [5] an XML-based language. The GPOL includes some specific directives needed in a grid environment, such as: state maintenance, potentially transient services, notification, and group oriented services. Additionally, includes variables, lifecycle, fabric/instance control, flow control,

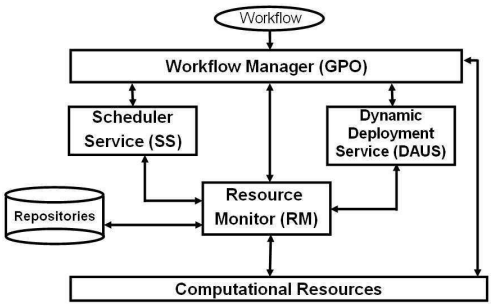



**Figure 1. SGT Software Infrastructure Architecture.**

and allows the user to start sequentially or in parallel the execution of tasks, services, and workflows. The Dynamic Deployment Service (DAUS) is responsible for the dynamic instantiation of services needed to execute the workflow. Information about grid resources and current available services are obtained by the Resource Monitor (RM) in a distributed manner. The RM performs strategic role by supplying information which allows the resource manager to identify bottlenecks and concurrency situations. Such information is stored in the repositories and is used by the scheduler in its decision making process. The Scheduler Service (SS) can use simple algorithms, as distributing the services according to the number of cores on each resource, or it can use more complex algorithms [6], which use dynamic information provided by grid monitors.

In service-oriented grids with VMs, the scheduler (SS) has to consider new information, which includes: the overhead of the virtual machine and consequent impact in the performance when executing workflows; information about which grid resource hosts which virtual machine(s); performance of hosts when the virtual machine(s) hosted by them are executing workflows; performance of virtual machines when their hosts are executing workflows; and performance of virtual machines executing workflows at the same time and hosted in the same grid resource. These information could be added to the objective function optimized by the scheduler, which can be the minimization of the workflow execution time, for instance. In the experiments presented in this paper we quantify these overheads and show how the concurrency affects the performance. With this, in conjunction with the information about VMs provided by the RM, we would have enough information to develop a VM-aware scheduling algorithm.

## 3. THE APPLICATION

To analyze the performance of VMs in our testbed we used the Network Simulator 2 (NS2) [7], a single core application which simulates computer networks and generates

**Table 1. Computational Resources Specifications.**

| Name (abbrev.) | Processor | Cores | Clock | RAM | Disk | Is it a VM? (host) |
|---|---|---|---|---|---|---|
| Cronos ($C0$) | Intel Core 2 Quad | 4 | 2.4 GHz | 4 GB | 2x320 GB | No |
| Medusa ($M0$) | Intel Xeon E5430 | 8 | 2.66 GHz | 2 GB | 19 GB | No |
| Temis ($T0$) | Intel Xeon E5430 | 8 | 2.66 GHz | 1.3 GB | 95 GB | No |
| Helios ($T1$) | Intel Xeon E5430 | 4 | 2.66 GHz | 6.6 GB | 390 GB | Yes ($T0$) |
| Persefone ($T2$) | Intel Xeon E5430 | 4 | 2.66 GHz | 6.6 GB | 390 GB | Yes ($T0$) |
| Caos ($M1$) | Intel Xeon E5430 | 4 | 2.66 GHz | 6.6 GB | 430 GB | Yes ($M0$) |
| Eros ($M2$) | Intel Xeon E5430 | 4 | 2.66 GHz | 6.6 GB | 430 GB | Yes ($M0$) |

traces of network traffic. NS2 is a discrete event simulator targeted at networking research. NS2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. The NS2 architecture is composed of an object oriented simulator in C++ and an OTcl (Object-oriented Tool Command Language) interpreter [7]. OTcl scripts created by the user are executed by NS, that generates files with the events produced during the simulation. The oTcl script file describes the network topology used in the simulation as well as the traffic flows to be simulated, along with characteristics as bandwidth, delays, queues, protocols, and simulation time.

Our experiments involve simulations of WiMAX (Worldwide Interoperability for Microwave Access) [8]. In our simulations we use a WiMAX module based on the IEEE 802.16 [9, 10]. Our base scenario is the communication between uniformly distributed *subscriber stations* and a *base station*. Using this, we built several others by mixing different parameters. For all scenarios we executed simulations for 20, 40, and 60 seconds [1].

Simulations run in NS2 are characterized by the intensive use of resources, since each scenario must be simulated several times until the results converge according to the characteristics of the simulated network. Therefore, several CPU hours are usually needed in this kind of simulation. Another remarkable characteristic is the intensive disk I/O when traces are stored to be post-processed, which can be as large as several GB in the disk. In this context, the CPU availability in the grid makes this environment suitable for this kind of simulation, while VMs bring new challenges to the resource management.

# 4. EXPERIMENTAL RESULTS

To observe the VMs behavior in the grid we made simulations of computer networks with NS2. We built a service which runs NS2 simulations with different parameters, being able to run several scenarios. We aim to evaluate how hosts and their VMs would behave when the scheduler considers them as separate machines, and what it must consider to have a better overall performance.

We start our analysis running the application in a traditional manner, where each scenario is executed in a single CPU. In a second stage, we go through the use of management features provided by our grid middleware, constructing service workflows which allow the execution of more than one scenario in parallel. We built workflows in GPOL to make the parallel execution of these scenarios. The parallelism in the execution brings a significant improvement in the performance, however not all combination of resources brought the expected benefits. All results show averages over 30 executions without other user processes running in the machines, thus with the resources dedicated to the experiment.

## 4.1. Performance and Workflow Overhead

First we executed NS2 simulations in the traditional manner. Figure 2(a) shows the time spent in NS2 simulations with duration between 20s and 60s in the physical machines $T0$ and $C0$, and in the VMs $M1$, $T1$, $M2$, and $T2$. As we stated before in Section 2, $M1$ and $M2$ are hosted by the physical machine $M0$, while $T1$ and $T2$ are hosted by $T0$. In these simulations we stored only events from the simulation (trace files), and we did not store the detailed execution file (log file). The results show that the physical machines are more efficient than VMs. For instance, $T1$ and $T2$ have around $45\%$ lower performance than $T0$ (their host, thus the same hardware) for the 60s simulation.
We executed the same scenarios with service workflows to check the overhead introduced. The graph of Figure 2(b) shows that the execution of the NS2 simulation using workflows generates a minimum overhead ($4\%$ on average) when compared to executions in the traditional way shown in Figure 2(a). Also, we can observe that the execution using workflows maintains the performance difference between physical machines and VMs.

[1] Note that this simulation time is not related to the real time taken to the simulation be executed, but it is related to how many seconds of network traffic in the network is simulated and logged

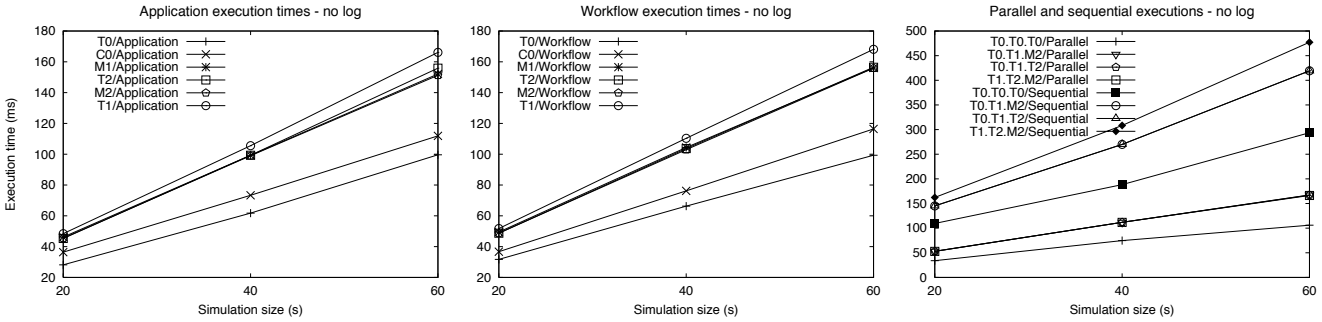



**Figure 2. NS2 Application, Workflow, and Parallel Workflow Execution Times.**

We now evaluate the performance with the workflow parallelism. Figure 2(c) shows the workflow execution times for three parallel WiMAX scenarios in $T0$ ($T0.T0.T0$); one scenario in $T0$, one in $T1$, and one in $M2$ ($T0.T1.M2$); one in $T0$, one in $T1$, and one in $T2$ ($T0.T1.T2$); one in $T1$, one in $T2$, and one in $M2$ ($T1.T2.M2$). We can observe that it is possible to execute three scenarios with nearly the same execution time of a single scenario. The overhead of the parallelism is around $8\%$ in the worst case ($T0.T0.T0$ versus $T0$ in Figure 2(b)), but it can be as fast as the execution of a single scenario in the VMs ($T1.T2.M2$ versus $M2$ in Figure 2(b)). If we compare the sequential execution of the same scenarios using traditional applications ("Sequential" suffix in Figure 2(c)), workflow executions are $2.75$ times faster on average.

In Figure 2(c) we can also observe that using $T0$ to execute three scenarios in parallel ($T0.T0.T0$/Parallel) is better than using the VMs ($T0.T1.M2$/Parallel, $T0.T1.T2$/Parallel, $T1.T2.M2$/Parallel), even with these VMs having the same processor and more RAM. On the other hand, the use of parallel workflows is one manner of taking advantage of available VMs as computational resources in the grid to speed up the execution of simulations. The parallel execution in $T0.T1.T2/Parallel$ ($T0$ and its VMs) is $1.84$ times faster than the sequential executions running in $T0$ alone ($T0.T0.T0/Sequential$).

## 4.2. Log Recording Analysis

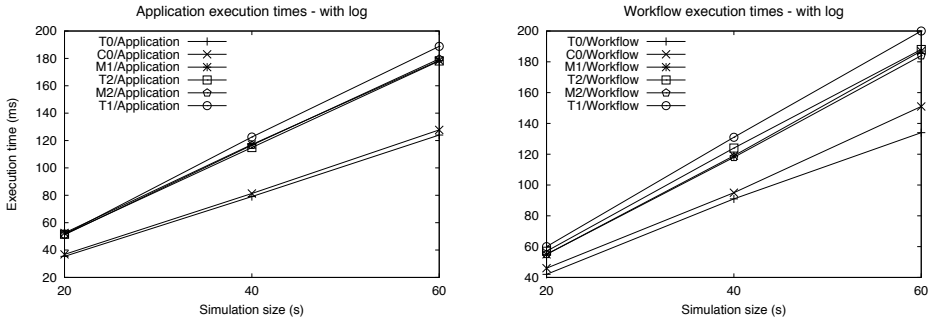
In the next experiments we added intensive disk I/O with simulations storing the detailed log file. Each simulated scenario generates a log file with 740 MB (20s simulation) to 2.6GB (60s simulation). Figure 3(a) shows the results for the execution of traditional applications with log storage. While executions in $T0$ are $25\%$ slower than the ones without log storage (Figure 2(a)), executions in $C0$ and in the VMs ($T1$, $T2$, $M1$, and $M2$) lost around $15\%$ in performance. This can be explained by the amount of RAM available in $T0$ (1.3 GB). We observed during the execution of simulations that, when recording log files, the amount of needed memory is higher than the available RAM in $T0$. Therefore, when installing and configuring VMs in a grid, one must consider that the lack of RAM left for a fast machine may compromise its performance.

Results for simulations with workflows storing logs are shown in Figure 3(b). We can observe the same pattern, with $T0$ losing more performance than the others. With log, in general, $T0$ was $35\%$ slower on average due to lack of memory, while resources with more RAM were $19\%$ slower ($T1$ versus $T1$ in Figure 2(a), for instance).
Figure 4 compares execution of workflows with and without log recording (Figures 4(a) and 4(b), respectively). Three scenarios executed in parallel ($T0.T0.T0$) for 20s of simulation was $57\%$ slower with log recording, while the combination with three VMs ($T1.T2.M2$) was $17\%$ slower. Even with this performance difference, with log recording $T0$ is still faster than its VMs ($T1.T2.M2$) for $20s$ simulations. For $40s$ simulations and on, the VMs jointly are more efficient than $T0$. For $20s$ simulations, $T0.T0.T0$ is $16\%$ faster, while for $40s$ and on, it becomes around $19\%$ slower. In this scenario, besides the lack of memory, $T0$ has also to deal with disk access concurrency. Thus, the resource manager must be aware of this concurrency when sending workflows to execute in parallel.

The observed behavior when executing workflows in parallel is important and must be considered by the scheduler. When receiving a submission with parallel services, the SS must consider the use of VMs and their characteristics in the execution instead of only considering that the performance of physical machines are better for a single service.

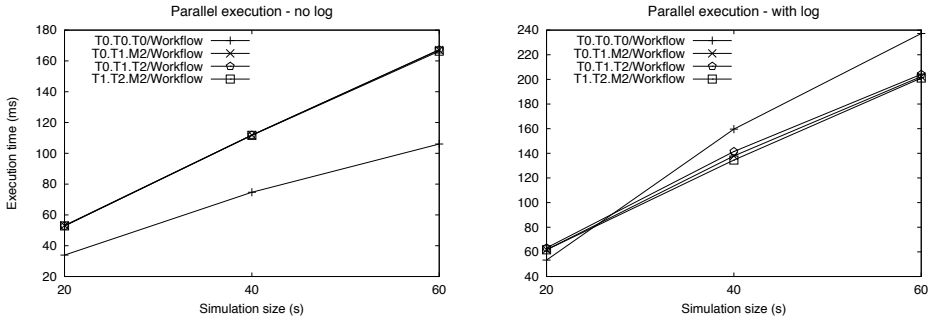
## 4.3. CPU Stress

After verifying the influence of the disk I/O, now we evaluate the behavior when the parallelism is increased and,




(a) (b)

**Figure 3. NS2 Application and Workflow Execution Times with Log File.**




(a) (b)

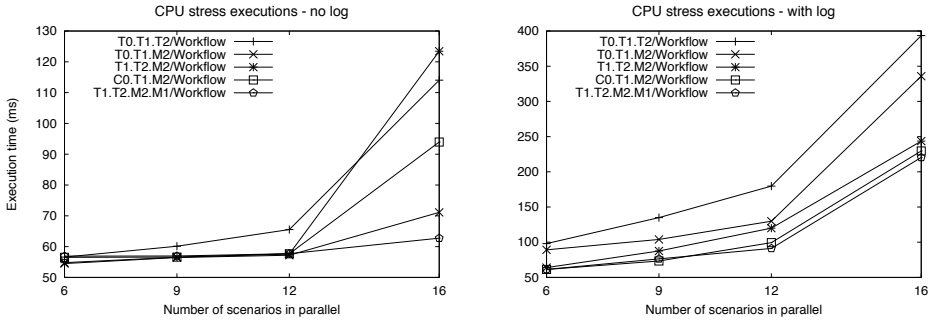
**Figure 4. Parallel Workflow Execution Without Log Versus Parallel Workflow Execution With Log.**

consequently, the CPU cores are intensively used. Figure 5(a) shows the execution of workflows that simulate from 6 to 16 scenarios in parallel. In these simulations we combined: $T0, T1$, and $T2$; $T0, T1$, and $M2$; $T1, T2$, and $M2$; $C0, T1$, and $M2$; and $T1, T2, M2$, and $M1$ (this combination uses 4 VMs but it still has only 2 physical machines). When we run 6 scenarios in 6 CPU cores the performance is similar in all combinations of machines. When the parallelism increases, thus executing more scenarios in parallel, the $T1.T2.M1.M2$ combination becomes the best option. For 16 scenarios in parallel, $T1.T2.M1.M2$ (four virtual machines) is up to $80\%$ faster than $T0.T1.T2$, for instance.

It is important to observe that the number of real cores is extrapolated (a CPU runs more processes than the number of cores available) when the resource manager does not consider that some machines are VMs. For instance, with 16 processes in parallel, a resource manager not aware of the VMs may select the combination $T0.T1.T2$ instead of $C0.T1.M2$, since $T0$ has 8 cores, while $T1$ and $T2$ have 4 cores each, which attends (in theory) the 16 cores demand. On the other hand, $C0$ has only 4 cores and its performance is worse than $T0$, and $C0$ combined with $T1$ and $M2$ (which has the same performance of $T2$), has only 12 cores. But $C0.T1.M2$ is a better choice than $T0.T1.T2$, since the 4 cores of $T1$ and the 4 cores of $T2$ actually are the same 8 cores from $T0$. This is only one example of misinterpretation that can arise if the scheduler is not aware of which machines are VMs and what are their hosts.

Adding log recording the differences become larger. Figure 5(b) shows that the combination $T1.T2.M1.M2$ is always more efficient, reaching a performance $97\%$ better than $T0.T1.T2$. On the other hand, $C0.T1.M2$, which uses three different physical machines, has almost the same performance of $T1.T2.M1.M2$. In this case, the disk concurrency is making the $T1.T2.M1.M2$ combination performance to decrease around 3 times for 16 processes in parallel, while the combination $C0.T1.M2$, which uses 3 different physical machines, decreases only around 2 times.

In these experiments we can see that combinations with less powerful machines ($C0.T1.M2$, for instance) are important to show that resources with less processing power ($C0$, in this case) can be interesting options when combined in the correct way by the grid resource manager. However, to do this, the resource manager must be aware of which machines are VMs and which are not, besides having in-




(a) (b)

**Figure 5. CPU Stress in Parallel Workflow Execution With and Without Log.**

formation about where each VM is hosted.

### 4.4. Discussion

The loss of performance, caused by the concurrency introduced in the grid result of the distribution of workflows unaware of VMs localization, can compromise the resources performance, making less powerful resources more efficient. We detected situations where disk intensive services are run in VMs within the same physical machine and the number of physical CPU cores are extrapolated. Both of these situations were created by wrong choices based on correct information about the resources performance.

Table 2 summarizes some situations where the characteristics of each machine influence in the resources choice, with each column sorted by performance. For example, to execute the traditional applications shown in the first column it is sufficient to choose the machine with the best hardware configuration ($T0$). In parallel scenarios, the choice is not straightforward: disk and CPU concurrency must be considered in a grid with VMs. In the third column we can see that the bottleneck generated by the concurrency in the disk I/O and lack of RAM in $T0$ eliminates its advantage, turning $T0.T0.T0$ into the worse option for this kind of execution. On the other hand, despite the fact that VMs were $50\%$ worse when running one application alone, choosing them to execute services in parallel brings benefits by distributing the disk I/O ($T1.T2.M2$). When stressing the CPU (last column), because in the $T0.T1.T2$ combination the physical CPU used is the same, its performance is degraded when many scenarios are executed in parallel, while $T1.T2.M1.M2$ is more efficient. This is because such combination is distributed in two physical machines, thus it has a higher number of real cores available.

The use of VMs in a grid can be interesting when the resource manager considers the concurrency when VMs in the same physical machine are used in parallel.

## 5. RELATED WORK

Our SGT involves some technologies like grids, SOC, execution of service workflows, virtualization, network simulation, and grid testbeds, and there are many works related with these areas. [3] presents a good view about virtual machine architectures. In [11] the authors organize solenoid applications in virtual data system workflows and execute them on VMs in a computational grid. [12] proposes a metascheduler framework that achieves coallocation using the concept of virtualization. They used Deviation Based Resource Scheduling algorithm to initiate SLA negotiation with other resources for participating in resource co-allocation, and it also supports SLA monitoring and enforcement. However they discuss only the scheduling performance. Finally, [13] focuses on green computing by scheduling virtual machines in a computer cluster to reduce power consumption.

The references presented here do not consider services in their analysis. Our work contributes in evaluating the combination of service-oriented grid with virtual machines, analyzing the new requirements and possibilities in such combination of resources. Our analysis suggest that service workflows, where there exist mechanisms to allow the creation of instances and parallelism support, is an alternative to the use of VMs in the same conditions shown in [12].

## 6. CONCLUSION

The use of VMs in the grid brings new management requirements. In this paper we evaluate the performance in the execution of service workflows in VMs as computational resources in our service-oriented grid testbed (SGT) using the Network Simulator 2. In our experiments we constructed several workflows to execute different simulation scenarios of WiMAX networks, and we executed these workflows to evaluate the performance of the VMs in rela-

**Table 2. Performance of Computational Resources in the Experiments.**

| Applications / no log | Workflows / log | Parallel executions / log | CPU stress / log |
|---|---|---|---|
| $T0$ | $T0$ | $T1.T2.M2$ | $T1.T2.M1.M2$ |
| $C0$ | $C0$ | $T0.T1.M2$ | $C0.T1.M2$ |
| $M2$ | $M2$ | $T0.T1.T2$ | $T1.T2.M2$ |
| $M1$ | $M1$ | $T0.T0.T0$ | $T0.T1.M2$ |
| $T2$ | $T2$ | | $T0.T1.T2$ |
| $T1$ | $T1$ | | |

tion to their host machines in the SGT.

Our experimental results show that, despite the VMs advantages, the grid manager must take into consideration where each VM is hosted to avoid concurrency. Unintended concurrency in disk I/O and CPU is possible when VMs are present in the environment. The results indicate a performance loss when such concurrency occurs in situations where disk intensive services are run in VMs within the same physical machine and the number of physical CPU cores are extrapolated. Therefore, the concurrency introduced without knowledge about the VMs and their hosts can deteriorate the performance of the resources. Thus, a VM-aware resource manager is fundamental to explore the hardware and its VMs in their full extent.

Future works include the development of a VM-aware scheduler service that takes into consideration the lessons learned from our experiments and uses the VMs information given by the SGT's resource monitor.

## ACKNOWLEDGEMENTS

We thank Flavio Kubota for his help with the NS2, and CAPES, FAPESP (05/59706-3), and CNPq (472810/2006-5 and 142574/2007-4) for the financial support.

## REFERENCES

[1] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," 2002. [Online]. Available: http://www.globus.org/research/papers/ogsa.pdf

[2] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The next step in web services," *Communications of ACM*, Vol. 46, No. 10, pp. 29–34, 2003.

[3] J. E. Smith and R. Nair, "The architecture of virtual machines," *IEEE Computer*, Vol. 38, No. 5, pp. 32–38, 2005.

[4] I. Foster, "Globus toolkit version 4: Software for service-oriented systems", IFIP Intl. Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, Beijing, China, pp. 2–13, 2005.

[5] C. Senna, L. Bittencourt, and E. R. M. Madeira, "Execution of service workflows in grid environment", 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom 2009), Washington, USA, pp. 1–10, 2009.

[6] L. F. Bittencourt and E. R. M. Madeira, "A performance-oriented adaptive scheduler for dependent tasks on grids," *Concurrency and Computation: Practice and Experience*, Vol. 20, No. 9, pp. 1029–1049, 2008.

[7] "Network simulator 2," 2009. [Online]. Available: http://nsnam.isi.edu/nsnam

[8] "IEEE standard 802.16-2004, part 16. air interface for fixed broadband wirelesss access system," IEEE, Tech. Rep., 2004. [Online]. Available: http://www.ieee802.org/16/pubs/80216-2004.html

[9] J. Freitag and N. L. S. da Fonseca, "Wimax module for the ns-2 simulator," IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, Athens, Greece, pp. 1–6, 2007.

[10] "WiMAX module for the ns-2 simulator", Computer Networks Laboratory, Institute of Computing, UNICAMP, 2009. [Online]. Available: http://www.lrc.ic.unicamp.br/wimax_ns2/

[11] L. Wang, M. Kunze, and J. Tao, "Performance evaluation of virtual machine-based grid workflow system," *Concurrency and Computation : Practice and Experience*, Vol. 20, No. 15, pp. 1759–1771, 2008.

[12] T. S. Somasundaram, B. R. Amarnath, B. Ponnuram, K. Rangasamy, R. Kandan, R. Rajaian, R. B. Gnanapragasam, M. Ellappan, and M. Bairappan, "Achieving co-allocation through virtualization in grid environment,", 4th International Conference on Grid and Pervasive Computing (GPC 09), Geneva, Switzerland, pp. 235–243, 2009.

[13] A. X. H. von Laszewski, G.; Lizhe Wang; Younge, "Power-aware scheduling of virtual machines in dvfs-enabled clusters,", IEEE International Conference on Cluster Computing and Workshops (CLUSTER 09), New Orleans, USA, pp. 1–10, 2009.