J Grid Computing (2012) 10:237–248
DOI 10.1007/s10723-012-9203-x

Deploying LiveWN Grids in the Greek School Network

Michael N. Kalochristianakis · Fotis Georgatos · Vasilis Gkamas · Giannis Kouretis · Emmanouel Varvarigos

Received: 7 December 2010 / Accepted: 15 February 2012 / Published online: 3 March 2012

Abstract A determinant factor for the introduction of grid technologies in production domains of scale can be the design of easy, fast and, from an operational point of view, realizable deployment procedures. Remote system management technologies, typically used to monitor and manage IT environments, are designed to offer remote software installation functionality that exhibits the aforementioned characteristics; however, previous work has shown that even valuable systems can fail to perform in heterogeneous, geographically distributed environments, especially if they are maintained by organizations affiliated to the public sector. The deployment of grid technologies throughout the Greek School Network can be achieved by combining OpenRSM, a novel open source solution capable to support usable, configurable, infrastructure management use cases in heterogeneous environments and LiveWN, a grid scavenging solution that integrates live technologies with gLite grids.

Keywords Grid computing · Scavenging · Remote management · Open source

M. N. Kalochristianakis (✉) · V. Gkamas · E. Varvarigos
Computer Engineering and Informatics Department, University of Patras, Rio, Patras, Greece
e-mail: kalohr@gmail.com

E. Varvarigos
e-mail: manos@ceid.upatras.gr

F. Georgatos · E. Varvarigos
ETH Zürich, CSCS, Manno, Switzerland
e-mail: fotis@mail.cern.ch

V. Gkamas
Computer Technology Institute, Rio, Patras, Greece
e-mail: gkamas@ceid.upatras.gr

G. Kouretis
National Technical University of Athens, Athens, Greece
e-mail: kougian@gmail.com

1 Introduction

Grid computing relies on middleware technologies that implement logic on top of informational systems in order to build services that can exploit large-scale distributed computing and storage infrastructures. Grids have been the focus of intensive research in order to establish global resource management architectures capable to host novel, user-level applications. Research projects in the field of grid technologies, such as the Enabling Grids for E-sciencE (EGEE) [1] have produced middleware and infrastructures across Europe, and the European Grid Initiative (EGI) [2] has been founded to guarantee the sustainability, coordination and federation in the European Grid Infrastructure, among the individual National Grid Infrastructures (NGIs). The EGI

infrastructure currently numbers 331 resource centers world-wide, providing more than 85000 CPUs and several petabytes of storage [3]. The EGI infrastructure is based on the gLite middleware [4] that relies on a stack of lower level grid infrastructure technologies to manage distributed computational resources across organizations [5]. The Greek NGI is called HellasGrid and consists of 6 computer clusters with a total of more than 850 64-bit CPUs, and on-line storage in the form of disks and tapes of more than 100 TBs, interconnected over an end-to-end gigabit backbone. The HellasGrid is integrated with the EGI infrastructure via the National Research and Education Network (NREN) and is available to the academic community.

Extending the computational resources of the HellasGrid infrastructure by adding worker nodes running on commodity stations has been a problem, since it requires a combination of technical solutions that can implement realizable escalation procedures, besides available spare resources. The Greek School Network (GSN) [6], the organization that interlinks all school laboratories in Greece and provides telematic services to the national educational community, can be an ideal case for the deployment of grid technologies to large environments. The network infrastructure of GSN is designed and maintained by a consortium of research centers and universities serving more than 60000 teacher accounts out of a total number of 160000 teachers. GSN currently interconnects 14487 schools, which includes 5555 kindergartens, 5972 elementarily, 1685 junior high schools, 1007 highs and 268 technical education high schools. The former numbers represent 89.6% of the schools in Greece. There are 8380 school labs across GSN, 5211, 1801, 936, 432, respectively for each type of school mentioned above, giving a total of 62809 work stations. 19587 of the latter are located in elementary schools, 19880 in junior highs, 15850 highs and 7492 in technical education highs. The computational resources provided by GSN are both available and continuously updated; the main volume of workstations are located in school laboratories that are mainly used during day time, normally less than seven hours per day, five days per week [7]. The establishment of non-binding grid technologies such as Desktop Grids in the school laboratories can increase the size of the HellasGrid NGI by a significant factor [8] and produce a valuable testbed for research in the field of distributed technologies. An installation throughout GSN would need to cause minimum interference to client systems, so that the educational procedure is not disturbed and also keep costs low, through the use of open source systems.

Previous work commented on the possibility of using LiveWN [9] scavenging in the Greek School laboratories. The proposed solution addressed the issue from a technical, structural and administrative point of view. Considering such a prospect, it is obvious that efficient management solutions need to be employed. The current paper proposes the integration of LiveWN with OpenRSM [10], an open source tool for systems and network management, in order for the former to be transferred, installed, and massively run on demand throughout GSN labs. OpenRSM is light-weight, usable and scalable and can remotely manage systems and network. It builds on the experience of GSN in integrated remote management services. To the best of our knowledge, there is no reliable automated method to remotely and massively install a tool such as LiveWN and ensure that it is timely and appropriately used, unless a physical medium is employed. But the on-site manual installation at every PC cannot be an option. Instead, the OpenRSM agent can be distributed via the web; it is small in size, as opposed to LiveWN which is a full featured Linux image, it is easy to install with a single click and most importantly, it conveys management functionality to the OpenRSM server. The rest of the current paper presents the procedures and solutions designed for a viable scaled installation of LiveWN in the Greek School Network labs. The paper is organized as follows: Section 2 discusses related work in CPU scavenging and remote management technologies, Section 3 describes the installation of LiveWN in GSN and Section 4 presents our conclusions.

2 Related Work

There is an abundance of CPU scavenging and desktop grid technologies. SETI@home [11] is

among the most well-known ones, involving approximately four million computers throughout the internet in scientific experiments that pursue the search for extraterrestrial intelligence (SETI). Condor [12] and BOINC [13] are also known, tested and viable solutions for desktop resource scavenging. Condor has developed mechanisms and policies that can support high throughput computing through effective workload management, that is, job queuing, scheduling and policing, priorities, resource monitoring and management. BOINC originated as part of SETI@home and is a well-known grid system that supports running scientific experiments at remote computers. BOINC also provides a usable framework for building volunteering computer projects. The SZTAKI Desktop Grid [14] is a BOINC-clone project initiated in Hungary, run by the Computer and Automation Research Institute (SZTAKI) of the Hungarian Academy of Sciences. SZTAKI boosted the use of grids in Hungary and currently supports on more than 80000 nodes. ClusterGrid [15] is another project created by researchers in Hungary, aiming in integrating x86 computer stations into a countrywide set of interconnected clusters. The testbed infrastructure is provided by participating high schools, universities, public libraries, and infrastructure and coordination is provided by NIIF/HUNGARNET. XtremWeb [16] is a research project aiming to serve as a substrate for global computing experiments. It supports the centralized set-up of servers and workstations as worker nodes. In addition, it can also be used to build peer-to-peer systems where any worker node can become a client capable to submit jobs. OurGrid [17] is another valuable project for desktop grids. EDGES [18] is a grid project that aims to implement integrated grid infrastructures based on a variety of desktop grids that interface with former EGEE grid services. EDGES is nowadays replaced by EDGI. DEGISCO is also one of the followers of the EDGEeS project. There are also a number of commercial desktop grid solutions for enterprises; the most well-known ones are Entropia Inc [19] and Univa United Devices [20]. Commercial platforms typically support enterprise desktops, clusters and database servers and offer elaborate characteristics. For instance, Entropia can seclude the execution of desktop grid applications from other processes running thus isolating grid applications data access on client machines. There have also been efforts for grid technologies to inter-operate and also to integrate with other types of systems. In [21] the authors elaborate on bridging desktop grids with service grids for service resources utilization, and they present and compare three approaches for solving interoperability problems between desktop and service grids. In [22] an interesting interface between the Globus Toolkit and BOINC is presented. An interesting integration would be that of grids with infrastructure management systems; grid technologies could complement management services or resource management engines and infrastructure management systems could implement methods that enable the deployment and operation of grids in their infrastructure.

Commercial remote management systems cover the needs of the enterprise; high end systems such as Tivoli [23] by IBM, Unicenter [24] by CA, BMC management tools [25] and HP management platforms fall among the most well-known Enterprise Management Systems (EMSs). EMSs are integrated platforms that promise to monitor and control network and systems infrastructure by offering high level use cases in the areas of assets, network, software and remote desktop management. Their ultimate goal is to manage the total cost of ownership (TOC) of an organization via elaborate event registration, high level analysis, work flows organization and accounting techniques. EMSs are commercial systems and therefore come costs that often exceed licensing since both their elaboration and complexity can require dedicated infrastructure, optimization, specialized operators or consulting [26]. Open source solutions in the field of infrastructure management traditionally cover specialized cases of remote management such as network management and monitoring, inventory and assets management, configuration management, software distribution, remote desktop connection, workstation tools and utilities. Open systems can be easy to use, flexible and usable in dynamic environments; there are several valuable and known open source systems but none can be considered to reach the versatility of commercial EMS. Even so, the field of open source remote management

systems is progressing and during the last years a significant number of projects have extended their functionality to include more than one EMS subsystems. High level systems in fields such as enterprise architecture management [26] have been released, such as ITERAPLAN [27] that offers mapping, monitoring, analysis and presentation of the software and systems that are deployed across the architecture of an enterprise, aiming to optimize their use and minimize costs.

2.1 LiveWN

LiveWN is an open desktop grid platform compatible with LCG/EGEE grids, which exploits live media technologies in order to provide preconfigured, production-ready, grid-enabled systems. The LiveWN scavenging solution is a mixture of three technologies, Linux LiveCD, LCG/EGEE and LiveWN Server. The LiveCD technology provides ease of use and hardware independence. Users do not have to install or configure their system, while at the same time they are disposed with a fully relocatable environment. The LiveWN server incorporates the services needed by a site in order to properly deploy LiveWN easily that is, OpenVPN, OpenAFS, Rsync and PlatnetLab nodes support. OpenVPN provides tunneling techniques for a generic VPN solution under unpredictable network environments, for instance, behind firewalls. It has been employed in order to meet the public IPv4 addressing requirements imposed by the middleware, via the provision of single IP address per worker node identity. AFS is a highly available and scalable solution for file-space that allows the users to store and retrieve files, even when they work in a totally disk-less environment; this proves handy under stateless environments, such as CD/DVD-ROMs, NetBoot, etc. An OpenAFS service has been included in LiveWN for this reason. The Rsync server instance provides the development team with the ability to apply patches and updates during reconfiguration, minimizing the need for frequent software distributions on physical media or other network, or labor, intensive methods. The PlanetLab node service is optional but improves the availability of the LiveWN configuration management system when placed at strategic locations through the properties of the Coral Content Distribution Network [28]. Ideally, PlanetLab nodes should be placed near backbone network routers, for optimal reliability. There is no need for special configuration of such PlanetLab nodes, their mere existence will allow to survive wide network partitioning. Upon boot, LiveWN is pre-configured to retrieve an IP address from a DHCP server. It then configures initial network access and, once network connectivity is established, users may start the LiveWN service. Users are authenticated using a login/password in order to be assigned with a unique worker node identity, which is configured as part of a grid Computing Element (CE); it is entirely possible to auto-subscribe so-called anonymous resources, as the latter has been considered adequate in the case of GSN. Once correct credentials have been supplied, an OpenVPN tunnel is created and the system configures its host name and domain name and turns to the designated resource pool that is managed by the associated Computing Element in the grid infrastructure hierarchy. Pieces of configuration such as forward and reverse DNS, SSH keys and other useful values are retrieved from the environment of the system or can be specified by the user. After the system is initialized, it appears as just another Worker Node within a cluster of the grid infrastructure, and joins the Computing Element's queues in order to start accepting and executing jobs. LiveWN is adaptable with respect to network environments, since it uses VPN techniques. It can work behind firewalls or within private address space networks. This characteristic of LiveWN solves the problem of public IP addresses reservation and conforms with the GSN policies and design.

2.2 OpenRSM

OpenRSM is an innovative initiative in the area of open, integrated, lightweight remote management systems that builds on the experiences gained from the deployment and operation of commercial platforms for the GSN remote management service. It is one of the very few open remote management systems in the field that performs in scale and also implements enabling features such

as flexible installation modes, customization features, platform independence, security and simple management use cases. OpenRSM is lightweight so that it can be used by end users who are not specialized in the use of management or asset-reporting tools. It is also designed for fast and automatic deployment in order to cover the needs of administrators who manage very dynamic environments. The server, the agent, and the management console support automatic installations that are created using known open installer packagers. The size of the agent is about 1.5 MB, and thus it can be distributed even in remote areas connected via slow lines. After the installation, which exposes a graphical user interface, no configuration is necessary and the administrators can start using the tool. The development of the system adopts the open source model so as to exploit the dynamics of open technologies and gain value from integration, based on the assumption that even if there is no complete, integrated open source EMS, the related open technologies have matured to the point that the open source community can provide all the necessary components for one. Several open source projects were examined in order to find the most appropriate open source management tools available for the purposes of OpenRSM. Some of them are: OpenAudit for assets management and inventory, NINO and OpenNMS for network monitoring, WindowsGet and CURL for software management, TightVNC for remote desktop control, and a significant number of utilities and tools for smaller pieces of functionality ranging from zip tools and utilities, to network discovery and testing, report formatting or tree visual components. OpenRSM has been tested in four pilot installations, including a pilot installation in the Greek School network and has successfully passed extensive scale testing [10].

The major components of the system are the server, the agents and the management console. OpenRSM adopts the client-server approach; the agents of the system are installed in managed stations; agents model abstract manageable entities that convey administrative actions from the server. Administrative actions originate from the OpenRSM management console, the graphical user interface for users and administrators. Management commands that correspond to tasks are visually created at the console and are submitted to the OpenRSM server; from there they are sent to the agents. The server schedules and synchronizes the execution of jobs that represent administrative tasks. Jobs are designed with the use of the object-oriented model and play a central role in terms of usability, design efficiency and system scalability. They behave as standard abstract system tasks, for example, inventory, remote control, remote command, or as reusable user-created objects. Jobs can themselves be managed by administrators since their creation and execution stages are decoupled. The front–end of the OpenRSM logic is the server; its purpose is to schedule job execution according to user commands, prepare (wake) the agents for job execution, concentrate access to back–end services, and interact with OpenRSM proxy and database modules. The back–end of the OpenRSM server consists of web and database servers. The server wakes the agent using a custom handshaking protocol. If there are other complementary tasks that must be executed as a result of job execution, due to job dependencies, it makes sure they are also dispatched. The server is also responsible for monitoring the job execution progress. It sends event and logging information back to the management console so that all management scenarios can be handled by the management console. When jobs reach the execution stage they are served by one of the subsystems incorporated in the OpenRSM system. For instance, if the job is an inventory query the agent registers inventory information about the station it resides on and sends the information to the inventory web application, hosted by the web server of the OpenRSM system. OpenRSM was designed to support the management of environments such as the GSN access network LANs that typically take advantage of address translation technologies (NAT) in order to preserve the IP address space.

3 Deploying LiveWN in the School Network

Experience has shown that scaled installations can be difficult, sometimes due to approaches that lead to inadequate, unrealizable procedures. An example is the case of the installation of a

commercial platform for the remote management of GSN school LANs. Whereas installing the management software in urban schools seemed feasible although inefficient, it proved difficult to accomplish throughout the rest of the network. The problems became insurmountable due to the lack of support for customization and configuration features that resulted to installation procedures that could not scale. In many cases, the distribution and installation of the agent software of the service became a crucial issue. The system could not be distributed via the internet, due to the size of the agent distribution, making its transfer via low bandwidth ADSL connections impractical. Sending the agent software in the form of CD-ROMs and providing installation support by phone or email did not solve the problem; the software was not properly filed by schools, or could not be configured by the school staff. Last but not least, during the first steps of GSN's development when broadband penetration was low, the system was not operational over narrowband connections. Distributing software on a large scale needs to be realizable from a technical and from a design point of view. Distribution or roll-out problems can be overcome by using simple procedures, based on usable solutions and well-established technologies. The goal is to support flexible installation modes and multi-platform characteristics. Until recent distributions, even well-known proprietary remote management systems did not exhibit such characteristics.

Even if effective management, decisions, procedures and approaches carefully eliminate the limiting factors, the massive deployment of software throughout a network must be feasible by the configuration of the network itself. The design of the GSN network infrastructure depends on network address translation (NAT) technologies in order to conserve public IP address space. Each school lab is assigned a four IP address subnet for the gateway router and the lab server. The laboratory server provides proxying and DHCP service that assigns private addressing. This configuration could not sustain a scalable remote installation since it would pose restrictions to direct communication. The first step towards the configuration of GSN infrastructure for scalable installations was the modification of the addressing plan so that lab subnets are assigned unique IP address ranges that can be routed within GSN.

Considering the above, trying to distribute the LiveWN CD or DVD via conventional ways would be difficult. Instead, the OpenRSM agent can be distributed via the internet . It is lightweight and simple to use, can easily be distributed via the internet, supports very simple installation procedures and can manage all types of systems. The installation can be executed by teachers without any supervision since unattended installation is supported. OpenRSM can then be used to deliver, install and execute LiveWN. The delivery of LiveWN can depend on the software management system developed by OpenRSM. Executing LiveWN can be accomplished by virtualization technologies. The VirtualBox [29] open solution for workstation virtualization can be remotely installed and configured at school labs and scripting can be employed to create virtual machines that boot upon LiveWN. This solution provides an isolated computational environment that can share a fraction of the available resources. There are virtualization solutions that exploit kernel communication for optimal resource allocation however such solutions are suitable for servers and cannot be deployed in GSN labs; thus VirtualBox can be an enabling compromise that will enable a significant amount or resources to be used. The ISO image distribution of LiveWN can be the most appropriate since virtual machines support booting ISO images. Taking into account the network traffic and client space reservation, the lightweight version (650 MB) can be used. The feasibility of the described installation plan was verified in laboratory conditions where the details were clarified and the problems were identified and solved. The verification procedure and the results are presented in the next paragraph.

The implementation of LiveWN distribution and installation via OpenRSM relies on the creation of appropriate management jobs that can be massively sent to machines. The proof of concept installation was tested within a LAN environment of 20 workstations running Windows XP, the operating system widely used in GSN school laboratories. OpenRSM was tested for scalability; in a similar environment during the stress tests, OpenRSM could dispatch up to 40000 manage-

ment jobs [10]. The agent of the systems was installed via the internet and jobs were designed to provide solutions for software transfer of the LiveWN ISO image, the remote installation Virtual Box, virtual machine configuration and virtual machine execution. The steps of the procedure are listed in Table 1. Transferring LiveWN was achieved using standard OpenRSM software delivery jobs. Appropriate packaging of the LiveWN image was used, so that it would be feasible to place the ISO file in a known, predefined location. For Linux systems this included a shell script that created the appropriate directories and moved the image. For windows, the LiveWN distribution file was archived in a self-extracting executable file (SFX) that was constructed via packaging tools and appropriately configured to execute the aforementioned functionality. The file was uploaded to the OpenRSM server, encapsulated in a software delivery package and associated with a software delivery job. The construction of the job was straightforward and the execution was successful. However, there were cases where the LiveWN ISO was also decompressed upon the execution of the delivery job at the client end. In order to avoid such situations, an OpenRSM Remote Procedure Call (RPC) job was used to configure the registry at the client in order to disable ISO image decompression. The RPC job used the REG command to remove the appropriate registry value by setting in the \HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.iso key read by the packaging tool.

With the LiveWN image in place, the next step was to remotely install VirtualBox, using the OpenRSM delivery functionality. In order to perform the installation without user intervention, an MSI VirtualBox installation executable was used, since it supports standard flexible installation modes. Since there is no official MSI distribution for VirtualBox, the available installer and installer conversion possibilities were investigated to discover the appropriate conversion mechanisms. The installer extraction options of the VirtualBox installer package can be used to convert any version of the program that supports it, to an MSI format that can then be remotely installed using standard switches. Figure 1 presents the packaging form for the VirtualBox MSI at the OpenRSM management console. The executable was uploaded on the OpenRSM server and the execution parameters for installation and uninstallation were registered. A remote software delivery job was created and parameterized so that no interface would be exposed to the end

Table 1 Tasks and implementation approaches

High level task	Implementation
OpenRSM agent installation	Distribution via the web, single click, unattended installation
LiveWN ISO delivery	Configuration of a SFX archive
	Disable ISO extraction at the local registry database
	Configuration of a software package for LiveWN
	Configuration of a software delivery job for LiveWN
	LiveWN distribution
Driver signing bypass job	Remote procedure call job for a script that emulates user input
VirtualBox installation	Compilation of an MSI installer for unattended installations
	Software package configuration for VirtualBox
	Software delivery job creation for VirtualBox
	VirtualBox delivery
	Driver signing bypass job execution
Configure virtual machine	Registration of the LiveWN ISO
	Creation of Hard disks for the virtual machine
	Creation of the virtual machine instance
	Attachment of the LiveWN ISO to the virtual machine
	Auto-capture disable script execution
Start virtual machine	Remote procedure call to start the virtual machine

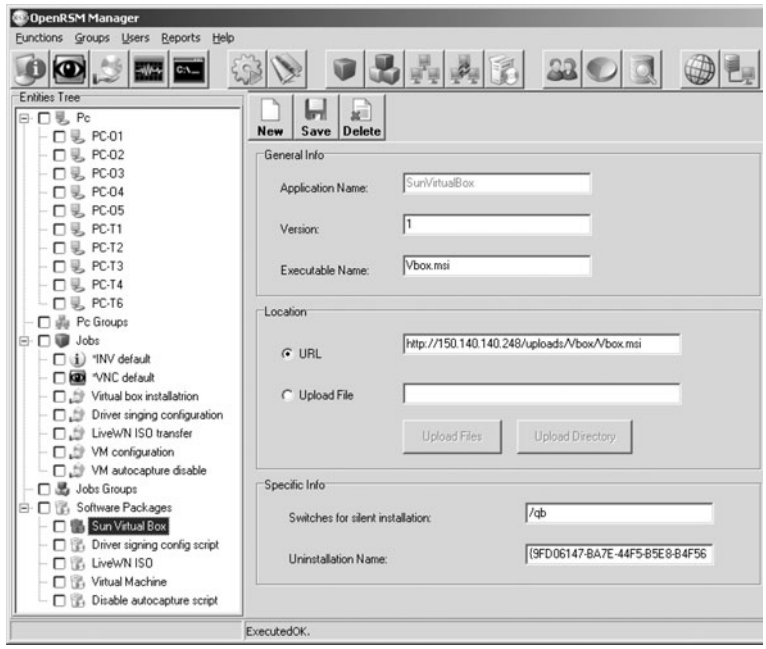

Fig. 1 The VirtualBox remote installation job

users during the installation. The delivery job is presented in Fig. 2. The installation was expected to succeed; however, since VirtualBox installs driver software and extensions that are not certified by Microsoft, even in the case of silent or unattended installation visual warnings appeared, requesting confirmation by the user. Automating the confirmation, so that no user interaction is

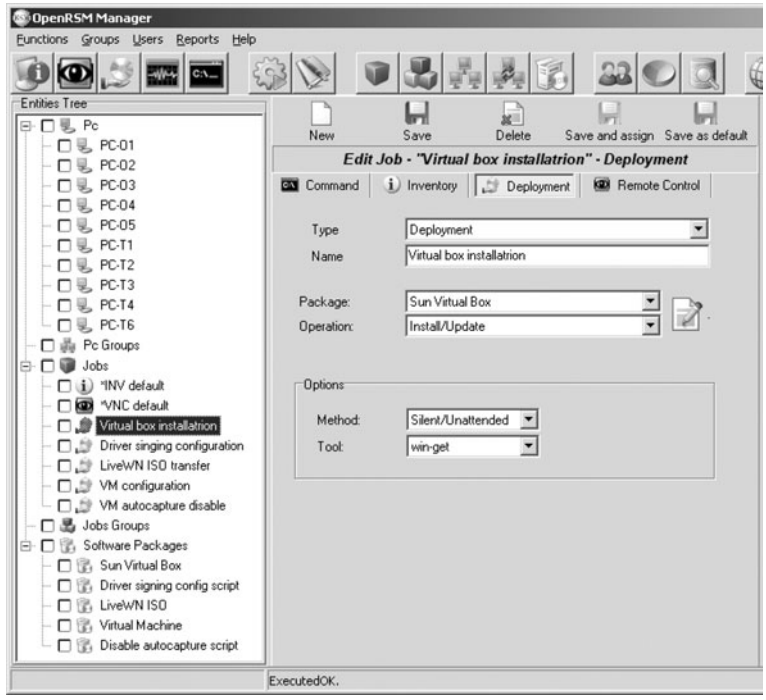

Fig. 2 VirtualBox package description

required became a crucial issue, since manual confirmation would linearize the installation procedure. To make matters worse, driver signing confirmation is not included in the Windows API, thus excluding the obvious solution of creating programs that set the default value of driver signing to ignore and sending it to the clients prior to VirtualBox delivery. For Windows server 2003 and later versions only, the executables can be signed using the windows software development kit (SDK).

Registry configuration RPC jobs were used to set the driver signing policy to ignore warnings only to verify that driver signing configuration is not feasible by means of direct or indirect API exploitation. The only solution could come by using high level solutions, such as scripting. Windows Script Host (WSH) was used to send click signals to the clients, so that driver signing would be configured visually, just as users would do it. The script opens a file system explorer window and simulates clicks via routines that can send signals to window objects. The "System Properties", "Diver Signing Settings", "Hardware" and "My Computer" tabs which were sequentially opened by successive clicks and thus configuration took place. The configuration of the virtualization software also included the creation of the appropriate scripts needed to configure a virtual machine that would run LiveWN. The necessary steps included the registration of the LiveWN ISO image with the virtualization software, the creation of a virtual hard disk, the creation of the virtual machine and the attachment of the already delivered ISO image and the virtual hard disk to the new virtual machine. VirtualBox offers utilities capable to execute the aforementioned tasks. Using the remote command system of OpenRSM, the former functionality was implemented even if the command syntax was complex and the length of the parameter strings were big. For instance, the command:

"vboxmanage openmedium dvd D:\livewn_working_dir\dvd_images\LiveWN-2_2_4-DVD-RC2.iso"

was encapsulated to the appropriate OpenRSM job wrapper and was sent in order to register the LiveWN ISO image with the virtualisation software. Then a command such as:

"vboxmanage createhd -filename D: \livewn_working_dir\ hard_disks\ livewnHD -size 6000"

was used to remotely create the necessary virtual hard disk at the nodes. The virtual machine was created by sending a remote procedure call such as:

"vboxmanage createvm -name livewn -ostype RedHat -basefolder D:\livewn_working_dir\vms –register"

The former commands, as well as similar ones that implemented the rest of the procedures, were sent in batch mode using OpenRSM jobs grouping. After the successful configuration of the VirtualBox throughout the school lab, the startvm command for VBoxManage was used to initiate the virtual machine. The successful execution of LiveWN also required the creation of a script for the deactivation of the auto-capture feature of VirtualBox that binds the keyboard and mouse for the exclusive control of the virtual machine. If not disabled, a warning/confirmation message will interrupt the initiation of the virtual machine, requesting user confirmation. Disabling this feature included changing attribute values for parameters such as the ExtraDataItem in the VirtualBox configuration file using simple windows command line scripting and an appropriate RPC job at the OpenRSM graphical console. After that, the virtual machine input devices were not free for use by the hosting system. Since the virtual machine was configured with the LiveWN ISO image attached to the DVD virtual drive, it successfully booted from it and since the latter was pre-configured to connect to HellasGrid it immediately did so. All the workstations of the pilot installation became connected grid nodes and could execute arbitrary jobs.

The procedure described above can be applied to any system that is distributed via bootable images and it can easily be applied by experienced technicians. Once the problems regarding operating system driver signing policies, or virtual machine configuration have been identified and resolved via the corresponding RPC

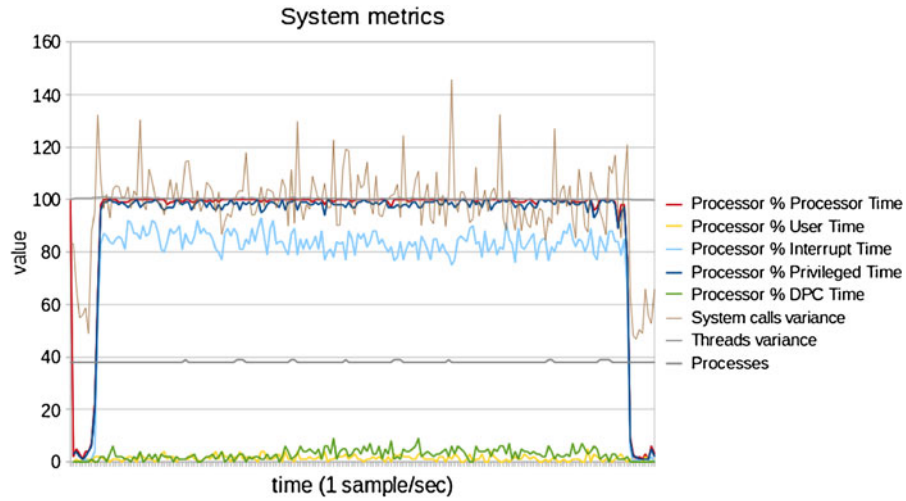

Fig. 3 Performance of the server system during lab tests

jobs, the deployment of new systems or virtual machines can be straightforward for typical OpenRSM users that is, administrators of computing infrastructures. The same jobs can be reused, or can be exploited as templates for the creation of new deployments, for example future releases of LiveWN with updated gLite stacks. It is also worth noting that the procedure can be implemented for any type of target platforms since OpenRSM supports cross-platform infrastructure management. The implementation for other types of systems, such as open source platforms would adhere to the same principles and configuration but would be much easier considering the ease of administration via command line jobs that install software and transfer files and the absence of limiting factors such as driver signing policies. The procedure was tested in lab conditions and was then applied in a school laboratory. Even if OpenRSM is known to perform in scale [10] it was never tested before against the management of very large volumes of data as part of complex scenarios. Thus, the lab tests aimed to monitor the behavior of the basic components of the system and identify potential problems, unpredicted behavior or performance bottlenecks.

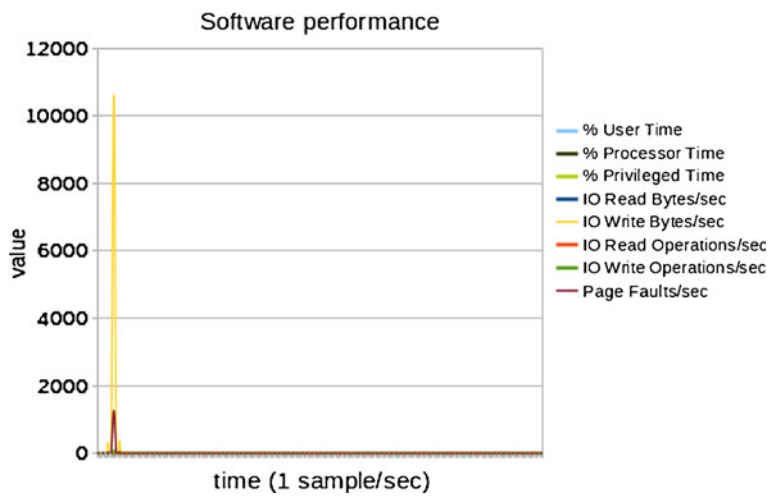

Fig. 4 Performance of the OpenRSM server software during the lab test

Figures 3 and 4 illustrate plots for characteristic metrics of the hosting system and the OpenRSM service respectively, measured during the batch disposal of LiveWN installations at the lab. When jobs are sent from the management console, the OpenRSM service loads the required data that is, the LiveWN ISO image and thus the I/O operations peek in Fig. 4. The management service is designed to integrate management tools by orchestrating their execution and it immediately passes the control to appropriate components that call system services capable to deliver data. Figure 4 shows that the load from the execution of the service is negligible; the server is primarily stressed by I/O operations since it needs to deliver LiveWN to remote stations and by page faults due to the size of the image. The former observations comply with the stress test results in [10] that show how OpenRSM scales to the limits of the underlying hardware. Figure 3 presents more detailed allocation of CPU time during the dispatch of a job that distributes LiveWN throughout the infrastructure. It shows how processor time is allocated and how the number of system calls, processes and threads vary during the remote installation of LiveWN. It can be seen that the processor is occupied by in system mode in order to call system services that handle resources such as disc space and network. All metrics that relate to operational functionality peek when the system processes the LiveWN images, while the number of threads and processes is practically constant.

4 Summary and Conclusions

We presented a solution for the creation of scaled remote desktop grid installations using the LiveWN desktop grid system and the OpenRSM remote management tool. The procedures were designed to be general so as not to limit the applicability of the solution. OpenRSM was capable to provide the infrastructure for the remote deployment of LiveWN and although the solution was tested in a LAN environment enumerating only 20 workstations, all the jobs, actions and procedures were designed to scale and thus produce results irrespectively of the size of the network.

The current work has shown that a scaled installation of LiveWN in GSN that will use OpenRSM as the remote installation infrastructure is feasible, provided that all the procedures can reach the appropriate scale. Experience in the deployment of the commercial remote management service for GSN has shown that if details are not accounted for, they can create insurmountable problems in the large-scale by linearizing procedures. All the installation aspects must be studied, including both managerial procedures and implementation. Small scale tests such as the current work are also necessary. The previous paragraphs prove that appropriate infrastructure selection, combinational engineering work and smart, scalable solutions are vital for supporting of such deployments. More specifically, the idea of automatically and remotely creating virtual machines for the hosting the grid nodes can be important since it enables the creation of grid infrastructures in GSN without disrupting the educational procedure. It also eliminates the need for physically distributing the desktop Grid system and using manpower to insert the LiveCD media in the workstations and booting them. The development of scripts or taking advantage of code can also be an enabling detail. The case of bypassing unwanted operating system features or checks such as the driver signing policies seen in the previous paragraphs is characteristic.

Future work will include steps towards the scaled installation of a pilot desktop grid in GSN. The installation will start with the deployment in a small number of school labs before going full scale. The upcoming LiveWN releases are expected to follow the evolution in the grid technologies it supports. Also, lightweight releases and form factors that facilitate scalable installations and set-up times are considered. Future releases of LiveWN will include an OpenRSM agent so that installations can support inherent management. Future work on OpenRSM will focus on integrating more functionality in the field of open integrated systems and network management. It will also include deeper integration with LiveWN by supporting default functionality for the automatic download of ISO images and updates. In that case,

a prospective scaled installation within GSN will inherently support desktop grid network creation.

References

1. Gagliardi, F., Jones, B., Grey, F., Bégin, M., Heikkurinen, M.: Building an infrastructure for scientific Grid computing: status & goals of the EGEE project. Phil. Trans. R. Soc. **A363**, 1739–1742 (2005)
2. Andronico, G., et al.: e-Infrastructures for e-Science: a global view. Journal of Grid Computing **1**, 1–30 (2003)
3. Field, L., Huang, J., Tsai, M.: Gstat 2.0: Grid information system status monitoring. 17th International Conference on Computing in High Energy and Nuclear Physics (2009)
4. Laure, E. et al.: Programming the Grid with gLite. Comput. Methods Sci. Technol. **12**, 34–45 (2006)
5. Jermini, P., Thiémard, M.: Condor at the Swiss Institute of Technology of Lausanne. Condor Week 2009, University of Wisconsin, Madison, Wisconsin
6. Kalochristianakis, M., Paraskevas, M., Varvarigos, E., Xypolitos, N.: The Greek School Network: a paradigm of successful educational services based on the dynamics of open-source technology. IEEE Trans. Ed. **50**(4), 321–330 (2007)
7. The Greek School Network networks and services stats pages. http://nic.ioa.sch.gr/schstats/. Accessed 20 Feb 2012
8. Kouretis, G., Georgatos, F., Alexopoulos, T., Tsipolitis, Y.: LiveWN: Scavenging in the Grid Era. 3rd EELA Conference (2007)
9. The LiveWN CPU scavenging project pages. http://gridathome.sourceforge.net. Accessed 20 Feb 2012
10. Karalis, I., Kalochristianakis, M., Kokkinos, P., Varvarigos, E.: OpenRSM:a lightweight, integrated, open source remote management solution. Int. J. Netw. Manage. **19**(9), 237–252 (2008)
11. Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@ home: an experiment in public – resource computing. Commun. ACM **45**(11), 56–61 (2002)
12. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor – a distributed job Scheduler. Beowulf Cluster Computing with Linux, The MIT Press (2002)
13. Anderson, D.: BOINC: a system for public-resource computing and storage. 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)
14. Kacsuk, P., Kovacs, J., Farkas, Z., Marosi, A., Gombas, G., Balaton, Z.: SZTAKI Desktop Grid (SZDG): a flexible and scalable desktop grid system. Journall of Grid Computing **7**(4), 439–461
15. Máray, T., Stefán, S., Szalai1, F., Vitéz, G.: The Hungarian ClusterGrid Project: challenges of a production grid. DSPN, (2004)
16. Fedak, G., Germain, C., Neri, V., Cappello, F.: XtremWeb: a generic global computing system. 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 582–587 (2001)
17. Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., Mowbray, M.: Labs of the World, Unite!!!. Journal of Grid Computing **4**(3), 225–246 (2006)
18. Urbah, et al.: EDGeS: bridging EGEE to BOINC and XtremWeb. Journal of Grid Computing **7**(3), 335–354 (2009)
19. Chien, A., Calder, B., Elbert, S., Bhatia, K.: Entropia: architecture and performance of an enterprise desktop grid system. J. Parallel Distrib. Comput. **63**, 597–610 (2003)
20. Univa UD Inc. http://www.univaud.com/. Accessed 20 Feb 2012
21. Farkas, Z., Kacsuk, P., Rubio del Sola, M.: Utilizing the EGEE infrastructure for desktop grids. 7th International Conference on Distributed and Parallel Systems, pp. 25–27 (2008)
22. Myers, D.S., Bazinet, A.L., Cummings, M.P.: Expanding the reach of Grid computing: combining Globus- and BOINC-based systems, Grids for Bioinformatics and Computational Biology. Wiley Book Series on Parallel and Distributed Computing. Wiley, New York (2008)
23. IBM Tivoli enterprise management system. http://www-01.ibm.com/software/tivoli/. Accessed 20 Feb 2012
24. CA Infrastructure and Operations Management. http://www.cs.com/us/infrastructure-management.aspx. Accessed 20 Feb 2012
25. BMC software. http://www.bmc.com. Accessed 20 Feb 2012
26. Jonkers, H., Lankhorst, M.M., Doest, H.W.L., Arbab, F., Bosma, H., Wieringa, R.J.: Enterprise architecture: management tool and blueprint for the organization. ISF **8**(2), 63–66 (2006)
27. Symbiose swischen EAM und SOA: IT Management, pp. 48–52 (2009)
28. Freedman, M., Freudenthal, E., Mazieres, D.: Democratizing content publication with Coral. 1st Symposium on Networked Systems Design and Implementation, vol. 1, pp. 18. (2004)
29. Li, P.: Selecting and using virtualization solutions: our experiences with Vmware and VirtualBox. VirtualBox, Journal of Computing Sciences in Colleges **25**(3), (2010)