VIRTUALIZATION TECHNIQUES IN COMPUTER SCIENCE – A PRACTICAL POINT OF VIEW

Nicolae Paraschiv, Prof. PhD. Eng., Gabriel Rădulescu, Lecturer PhD. Eng.
"Petroleum-Gas" University of Ploieşti, ROMÂNIA

ABSTRACT: The virtualization technology got its start (on mainframes) some decades ago, intending to avoid wasting expensive processing power. By this technique, on a single machine several applications could be served, all isolated into virtual operating system images that do not interfere each other. This manner of work allows new and revolutionary approaches in theoretical and applied computer science, especially for studies on operating systems behavior in heterogeneous environments.

1. INTRODUCTION

Developed more than 30 years ago, addressing that time some mainframe computing problems, virtual machine monitors have a new glorious and promising present, being used on standard (even entry-level) platforms, offering original solutions to challenges in computers security, reliability and administration.

Virtualization is the technology that allows multiple operating system images running all at once by using only one piece of hardware. Interesting architectures, platforms and applications have been designed, in order to take all benefits from such a novel approach in (parallel) computing – and the perspectives are obviously in a high spotlight.

This work proposes an original point of view on the possible virtual technology implication in studies over interactions between multiple operating systems in a homogeneous hardware and heterogeneous software environments, as well as how such a valuable tool can be used in training laboratory works for students in the field of computer science.

2. AN OVERVIEW ON THE VIRTUALIZATION TECHNOLOGY

A virtual machine monitor (VMM) manages the resources of a real hardware platform, offering an abstract representation of one or more virtual machines (VM) [5]. Such a virtual machine is able to run a standard operating system (OS) together with its own designed applications. Figure 1 depicts the architecture used by modern virtualization platforms (like the well-known VMware and VirtualPC environments).

Regarding the terminology, everything running inside a virtual machine is called *guest software* (i.e., guest operating systems, guest applications), while the software running outside the virtual machine – typically another operating system – is identified as *host software*.

All guest software (including the guest OS) runs in user mode, having a limited control over the real system hardware; it is only the VMM that runs in the most privileged level (kernel mode). By this mean, the host OS in Figure 1 is uniquely used to provide the basic access (also for VM-es, through VMM) to a wide variety of physical devices [9].

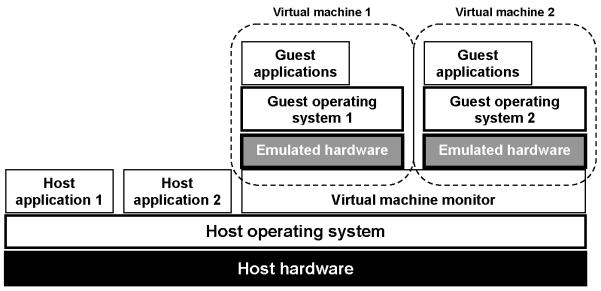

Figure 1. A standard structure for VMM, providing the abstraction of multiple VM-es.

The virtual machine monitors offer some hardware-level abstractions to the guest software, presenting them as emulated hardware. In the same manner as it would do with real hardware, the guest OS projects its actions on the virtual hardware (as input/output instructions, DMA transfers and so on). These complex interactions are trapped by the VMM and consequently emulated in software, allowing the guest OS to run in its standard way (as designed by factory, without any modification for virtualization purpose), maintaining in the same time a strict control over the system at the VMM layer.

A typical virtualization platform supports multiple virtual instances of various operating systems on a single computer by multiplexing the real (physical) hardware. Depending on the VMM performances, a perfect illusion of multiple, distinct virtual computers can be created when running separate operating systems and their applications. In order to provide a safe environment, the VMM isolates each virtual computer

and its emulated hardware through a fine adjustable redirection mechanism. For instance, the VMM can map a number of virtual disks to different zones of a shared physical disk. In the same way, the physical memory space of each virtual machine is mapped to different pages in the real machine memory system.

But the virtualization environments can be used not only for multiplexing the host computer hardware, but also to provide a powerful platform for supplementary services to an existing system. Typical targets for VMMs usage are, for example, debug sessions for new OS-es and different system configurations [7, 10], live machines migration [8], intrusions detection and prevention [4, 6, 1], code integrity test [3] and, of course, training activities for IT purposes. Some of these VM services are typically implemented outside the guest they are serving, an efficient manner to avoid perturbing the guest environment.

A problem that can occur when using VM services is represented by the difficulty in understanding the states and events inside the associated guest, as they run at a different level of abstraction from guest software. For the applications running outside of a virtual machine, the *virtual low-level state* is seen as disk blocks, network data packets and memory locations, while the software inside the VM sees this state as high-level abstractions such as files/directories, TCP data connections and variables. This dichotomy between the data/events significance for VMM and guest software is known as *the semantic gap* [2].

Virtual-machine introspection (VMI) [4, 6] is the technology covering virtual machine services that are able to interpret and modify states/events within the guest. VMI translates variables and guest memory addresses after reading the guest OS and applications symbols/pages tables. Hardware and/or software breakpoints are used in order to enable a VM service to gain control at a specific instruction address. Finally, the VMI technique makes possible a VM service to invoke guest operating system or application code to carry out general-purpose functions (for example, reading a guest file from the file cache/disk system). Regarding the data safety, the virtual machine services can protect themselves from guest code by disabling external input/output operations. In the same time, they can protect the guest data from perturbation by inserting the so-called *restore (reference) points*, which contain the "frozen image" of the guest environment, where it can be rolled back after an unsuccessful procedure.

A virtual machine monitor is a robust platform for studies on operating systems interaction (splitting them in *event-generating OS-es* and *target OS-es*). Such a platform, called *virtual machine based rootkit* (VMBR) installs the involved systems into independent virtual machines and then concurrently runs them. The target system practically sees no difference in its memory space, disk availability or execution (depending on the virtualization quality). The VMM also completely isolates the event-generating OS state/events from those of the target system, so software in the target system cannot see or modify the interacting software from the other system. At the same time, the virtual machine monitor supervises all state/events in the target system (keystrokes, network packets, disk state, memory allocation). Also, VMBR can quietly read and modify these states and events (without being observed inside the running VM-es), because it has the full control over the virtual hardware presented to the guests.

3. A COMPLEX PLATFORM FOR STUDIES ON VIRTUALIZATION TECHNOLOGY: PRACTICAL IMPLEMENTATION

In order to use in a valuable way the principles of such a robust and flexible environment, the authors have designed and implemented a complex multi-client platform at the "Petroleum-Gas" University of Ploieşti (Computers and Networks Laboratory), allowing the study of interactions between heterogeneous operating systems on virtual machines. This section of the paper intends to give an overview on how our system is build-up, both from the hardware and software perspective.

3.1. The hardware architecture

As shown in figure 2, the system is distributed over a Local Area Network (LAN) and consists in one main server and the associated workstation-clients.

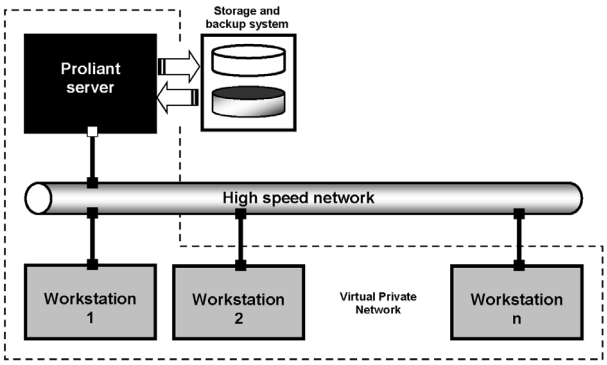

Figure 2. The schematic hardware structure

The central node is a HP Proliant ML310 server, on Intel platform (Pentium 4 CPU at 3.2GHz – Hyper-Threading Technology, 1GB EC DDR memory) with RAID storage system and backup facility in order to prevent any user data loss. By running under the "Protected by Proliant Global Pre-failure Warranty", the server is by itself a robust machine, providing an almost inexistent downtime (in our case, no time was required for maintenance and service as no failure occurred during operating sessions). Together with its high computing power, the use of a Gigabit network adapter – avoiding any possible communication bottleneck – also contributes to the server high throughput.

The system clients are located on independent workstations (typical Pentium 4 systems: 2GHz CPU speed, 512MB DDR memory, ATA storage system), connected to the Proliant server through a high-speed managed Ethernet switch. The quality of service inside the LAN (during working sessions) was dramatically improved by configuring a dedicated virtual LAN

(VLAN), which isolates the distributed system from the corporate network (at the University level).

3.2. The software configuration

In order to improve the compatibility between clients and server, as well as the system maintainability, a uniform OS installation was adopted. After benchmarking the available options, we considered installing the Linux SuSE 8.2 distribution on every client machine and also on the server – as it proved to be the most stable and, above all, most "controllable" and fine adjustable modern Linux platform serving our purpose (even if it is not the last SuSE version on the market).

A uniform user account management was adopted too, by locating the users database on the server, all clients being authenticated via NIS (Network Information Service). The user profile is mobile, all workstations providing a homogeneous way of access inside the system and to the server resources. More, the user home directories are located on the Proliant machine, exported via NFS (Networking File System) and auto-mounted at boot/logon time on every workstation, as suggested in figure 3. Although completely transparent for the user, this manner of work has the big advantage of an increased data safety (because a crash on the workstation only interrupts the communication with the server, but the last data components were already written there). Improved protection against malicious code (spyware applications, viruses) has to be expected, as locating the user files only on the server makes possible a central management, including the usage of an unique antiviral solution, only on the Proliant machine.

As virtualization platform, VMware was found to be the most flexible solution, by offering a high level of the hardware abstract representation and a very good compatibility with all the guest operating systems we intend to include from now on in our test sessions (Windows-based OS-es, heterogeneous Linux distro's, even the "old" MS-DOS). Currently we run some compatibility and endurance tests with VMware Workstation versions 4.5.3 (a well recognized stable release, with a really safe behavior) and 5.5.0 (the latest on the market, with some significant improvements, but somehow in its "early age"). In order to provide a high level of independency and data safety, each workstation has a separate standard VMware installation – on the server being kept only the VMware users mobile profiles.

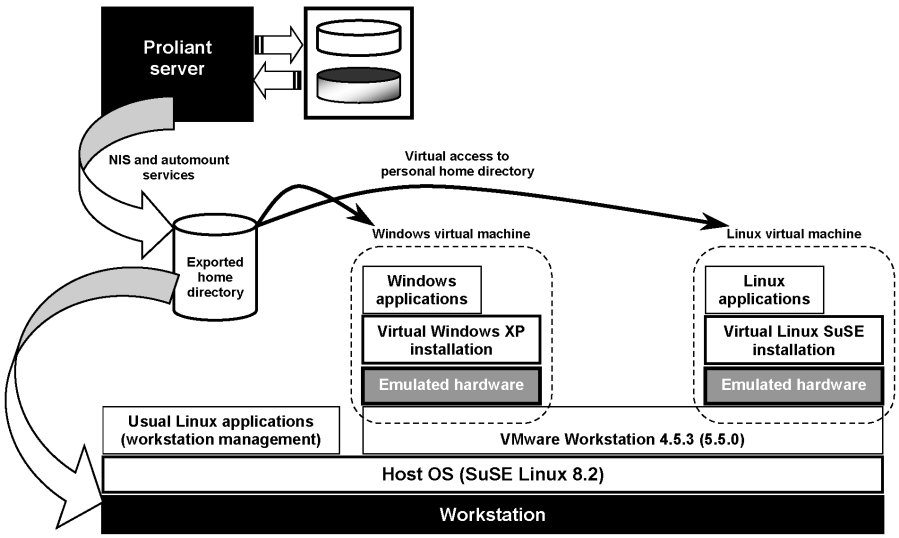

Figure 3. A representation for the system software structure.

3.3. Some remarks about the system utility

During our software environmental tests, as well as on training sessions for students, we started identifying the major advantages by using such a virtualization platform (and we are still counting these benefits). In this general presentation, we want to emphasize only a few of them:

a) It is possible to pack and distribute the classroom material in virtual machines with considerable less effort (in large networks, it could be at most 10% than by using the standard working manner, with multi-boot computers and independent OS-es). With this respect, an important facility we created for our system is the automated virtual machines replication on all workstation: when a software update is needed, it has to be applied only on one computer and the new VM image is multiplied in the network, making it available on all workstations, for each authorized user.

b) The system allows students to experiment with multiple operating systems, applications and tools in secure, isolated virtual machines. Also, this manner of work reduces the hardware needed in the laboratory.

c) By configuring the virtual machines to "undo" all changes at shutdown, it is possible to assign even administrator rights to the users (when dealing with

subjects on OS management tasks), or to generate system crashes during training sessions – as time as at next boot all VM instances will start from a standard "clean" state, without being affected by the previous user actions.

d) With the proposed hardware and software configuration, the time required for technical interventions, network tuning and maintenance activities are drastically reduced.

More than this, the system we designed keeps a very important feature: it has an *open architecture* in multiple ways. First, all network clients are identical, so integrating a new workstation for virtualization purpose in the system is trivially easy. Then, at logical (software) level, each virtual machine on every workstation has the same attributes, no matter what guest OS it uses – and the local VMM sees all individual VM-es as a homogeneous environment. Last, any system upgrade is always possible (we experimented several configurations before keeping the current one), as time as the architecture and functionality are preserved.

4. CONCLUSIONS

Virtualization, in its current state, is the technology allowing multiple operating system instances to simultaneously run by using a single hardware platform, without any interference. New and revolutionary approaches in the computer science, especially for subjects related on operating systems, are now possible by using the virtual machines technique.

This work presented a point of view on how the virtualization technology can be applied in studies over interactions between multiple operating systems in a homogeneous hardware and heterogeneous software environment, as a valuable tool for training laboratory works. This point of view has a true practical relevance, as time as the presented system we have designed and implemented is currently in full service.

REFERENCES

1. Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., Keromytis, A. D. – Detecting Targeted Attacks Using Shadow Honeypots. In: Proceedings of the 2004 USENIX Security Symposium, August 2005.

2. Chen P. M., Noble, B. D. – When virtual is better than real. In: Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS), pages 133–138, May 2001.

3. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A Virtual Machine-Based Platform for Trusted Computing. In: Proceedings of the 2003 Symposium on Operating Systems Principles, October 2003.

4. Garfinkel, T., Rosenblum, M. – A Virtual Machine Introspection Based Architecture for Intrusion Detection. In: Proceedings of the 2003 Network and Distributed System Security Symposium (NDSS), February 2003.

5. Goldberg, R. P. – Survey of Virtual Machine Research. IEEE Computer, pages 34–45, June 1974.

6. Joshi, A., King, S. T., Dunlap, G. W., Chen, P. M. – Detecting past and present intrusions through vulnerability-specific predicates. In: Proceedings of the 2005 Symposium on Operating Systems Principles (SOSP), pages 91–104, October 2005.

7. King, S. T., Dunlap, G. W., Chen, P. M. – Debugging operating systems with time-traveling virtual machines. In Proceedings of the 2005 USENIX Technical Conference, pages 1–15, April 2005.

8. Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M. S., Rosenblum, M. – Optimizing the Migration of Virtual Computers. In: Proceedings of the 2002 Symposium on Operating Systems Design and Implementation, December 2002.

9. Sugerman, J., Venkitachalam, G., Lim, B.-H. – Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In: Proceedings of the 2001 USENIX Technical Conference, June 2001

10. Whitaker, A., Cox, R. S., Gribble, S. D. – Configuration Debugging as Search: Finding the Needle in the Haystack. In: Proceedings of the 2004 Symposium on Operating Systems Design and Implementation (OSDI), December 2004.