Using a Virtualization Techniques – Based Platform for Advanced Studies on Operating Systems

Gabriel Rădulescu, Nicolae Paraschiv

***Abstract*—The virtualization technology allows several (sometimes critical) applications running on a single machine, but all isolated into virtual operating system images that do not interfere with each other. Such a working manner proves to be a revolutionary tool in computer science, especially when advanced studies on operating systems behavior are performed in order to reveal the hidden interactions in heterogeneous computing environments.**

I. Introduction

ALTHOUGH the virtual machine monitors (VMMs) theoretical principles were established about 40 years ago, focusing that time some stiff computing problems running on mainframes, at present they have a fresh and promising impact on computer technology, their application ranging from high-end servers to standard/entry-level platforms endowment.

Virtualization is the technique allowing multiple operating system instances running all at once by making use of only one hardware platform. Novel architectures and applications can be designed in order to benefit from such a modern approach in computing – especially when original, safe and reliable solutions have to be developed – and the perspectives are in a continuous spotlight.

In our previous works [1] and [2] the focus is put on how to practically build-up a platform based on virtualization techniques, designed as the main tool for studies on interactions between independent operating systems instances and a homogeneous hardware environment, when heterogeneous software platforms are involved. Since then, interesting results in this area were obtained, a brief selection being here presented.

II. A Short Overview On the Virtualization-Based Platform

A. Virtualization Principles

A VMM offers an abstract representation of one or more virtual machines (VM) by de-multiplexing the resources of a real hardware platform [3]. Such a VM may run a standard operating system (OS) together with its own native applications. Fig. 1 shows the classical architecture used by modern virtualization platforms (i.e. VMware, VirtualBox and VirtualPC solutions).

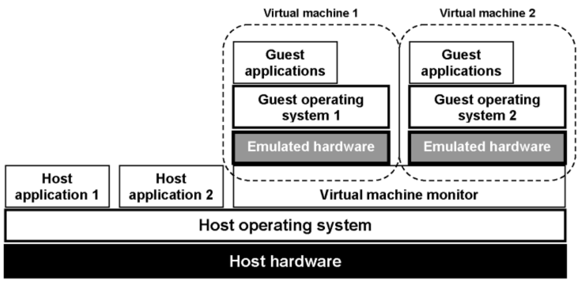

Fig. 1. The typical structure for VMM in the context of a host OS, providing the hardware abstraction layer for multiple VMs [2].

The software executed inside a VM is identified as *guest* (guest operating systems, guest applications), whereas the software running outside the virtual machine – typically the host platform operating system – represents the so-called *host software*.

The guest OS and its corresponding applications run in *user mode*, with hardly-limited control over the real hardware, but the VMM runs in the most privileged level (*kernel mode*), as the host OS in Fig. 1 is used to provide the basic access to physical devices for VMM and VMs [4]. The guest software uses the emulated hardware that is offered by VMM in the same transparent manner as it would do with real hardware. All complex interactions between guest software and the abstract hardware platform are trapped by the VMM and consequently emulated in software, allowing the guest OS to run in its standard way, also maintaining the strict control over the system at the VMM layer [1]. As previously shown in [2], by virtualization a perfect illusion of multiple, distinct virtual computers can be created, with separate operating systems and applications. For safely keeping the working environment, the VMM isolates each virtual computer and its emulated hardware through an adjustable redirection mechanism. The most easy-to-understand examples are, for instance, mapping a number of virtual disks to different zones of a physical disk or virtual memory mapping onto different pages in the real machine memory system.

The virtualization environments can be used not only for simply multiplexing the host resources, but also to add supplementary services to an existing system, including here special debuggers for new OSs, live machines migration [5],

Manuscript received May 5, 2010.

Gabriel Rădulescu is with the Petroleum-Gas University of Ploiesti, Automatic Control, Applied Informatics and Computers Department, 100680 Ploiești, Romania (e-mail: gabriel.radulescu@upg-ploiesti.ro).

Nicolae Paraschiv is also with the Petroleum-Gas University of Ploiesti, Automatic Control, Applied Informatics and Computers Department, 100680 Ploiești, Romania (e-mail: nparaschiv@upg-ploiesti.ro).

intrusion detection and prevention [6] as well as code integrity test [7]. It is also very important to mention that some of these services are implemented outside the guest machines and, as consequence, they do not affect at all the guest environment.

In this context, the virtual machine introspection (VMI) is represented by all technologies used to interpret and modify the inner states and events within the guest [8], [9]. By using VMI methods, the variables and guest memory addresses are translated (after reading the guest OS and applications symbols/pages tables) into real references for the host OS. More, through hardware and software breakpoints, a specific VM service is allowed to gain full control at a specific instruction address. Finally, by this technique, such a service may invoke the guest operating system or application code in order to make use of general functions (for instance reading a file from the guest OS file system). It is also very important to emphasize that all virtual machine services can be protected by disabling external I/O procedures. In the same time, VMs assure the guest data integrity by introducing the *snapshots* (restore points), which points to a frozen guest image where the system can be rolled back (after a crash, for example).

A special VMM application which deals with concurrent OS interactions is the so-called *virtual machine based rootkit* (VMBR), which monitors the VMs activity. The target (supervised) VM practically sees no difference in its memory space, disk availability or execution (depending on the virtualization quality) [1], [2]. There is a complete isolation between the event-generating host OSs and the targeted systems, so the software in the target system cannot see or modify the interacting software from the other system [8]. Also, apart from monitoring the target states and events, the VMBR can quietly read and modify them (without being observed from inside of the running VMs), as it fully controls the virtual hardware presented to the guests [1].

B. The Platform Architecture

As previously presented in [2], the authors have designed and implemented a robust, flexible and multi-client oriented platform at the “Petroleum-Gas” University of Ploiești (Computers and Networks Laboratory) which is a valuable environment for studying the OS complex behavior. This section outlines its main hardware and software characteristics.

1) The hardware architecture

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

The central node is a HP Proliant ML310 server with RAID storage system and backup facility in order to prevent any loss of the user data. As mention, the server has proved an extraordinary robustness, with almost inexistent downtime up to now (in the case, no time required for service as no failure occurred during operating sessions for over 3 years).

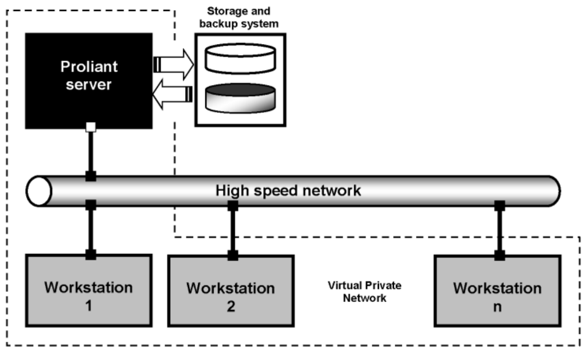

Fig. 2. A schematic representation for the system’s hardware architecture [2].

The system’s clients are located on independent workstations connected to the Proliant server through a high-speed managed Ethernet switch. While at the beginning all workstations had the same configuration, later we have used to run our experiments by targeting other hardware configurations (in order to extend the research results, if possible). The quality of service (QOS) inside the LAN was highly improved by configuring a dedicated virtual LAN (VLAN), which isolates the distributed system from the corporate network (at the University level) [2].

2) The software configuration

At first, in order to improve the compatibility between clients and server, the uniform OS installation was adopted (a very stable Linux SuSE distribution, both on server and client machines). But as the last research is focused on heterogeneous host OSs study, other Linux distributions (i.e. Fedora, Ubuntu) and Windows-class OSs are currently tested on workstations.

As suggested by Fig. 3, a uniform user account management was adopted too, by locating the users’ database and home directories on the Proliant server, all clients being authenticated via NIS (Network Information Service), whereas the storage resources are exported via NFS (Networking File System). Except the *root*, any other user has a mobile profile, with a homogeneous way of accessing the system and server resources. Apart being completely transparent for the users, this solution offers an increased data safety, as any crash at the workstations level only interrupts the communication with the server, without affecting the data last saved here over the network. The good protection against malicious code (spywares, backdoors, viruses) is easily implemented because, by locating the user files only on server side, the central management of any security/integrity solution is natively allowed [1], [2].

As virtualization platform, VMware proved to be the most flexible/reliable solution, with a high level of the hardware abstracting, as well as an almost perfect compatibility with all guest OSs we have included in the test sessions (Windows-based systems, different Linux distributions, even the native MS DOS 6.22). On this infrastructure we currently run software compatibility and endurance tests with VMware Workstation, with versions ranging from 4.5.x to 7.0.x.

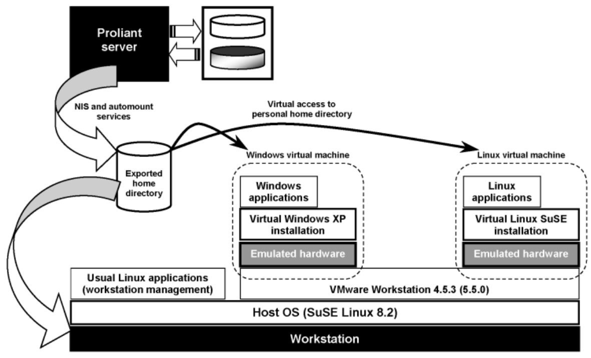
Proliant server
NIS and automount services
Virtual access to personal home directory
Exported home directory
Windows virtual machine
Linux virtual machine
Windows applications
Virtual Windows XP installation
Emulated hardware
Linux applications
Virtual Linux SuSE installation
Emulated hardware
Usual Linux applications (workstation management)
VMware Workstation 4.5.3 (5.5.0)
Host OS (SuSE Linux 8.2)
Workstation

Fig. 3. The simplified software architecture [2].

3) Advantages

The major advantages when using such a multi-client virtualization platform were extensively presented in [2], so in this general outline only a few of them are emphasized:

--The possibility to pack and distribute software pieces in virtual machines with considerable less effort.

--The facility of virtual machines replication on all workstations (“instantaneous” software update).

--The "undo" at shutdown feature, for all changes inside virtual machine instances.

--Reduced time for technical interventions, network tuning and maintenance activities.

But the most important advantage is the open system architecture, from many different points of view. For instance, all network clients can be considered as identical, so adding a new workstation in the system is trivially easy. On the other side, at logical level, all VMs on every workstation have the same attributes, no matter the installed guest OS, being seen by the local VMM as part of a homogeneous environment. At last, any system upgrade may be performed, as time as the architecture and its functionality are preserved [2].

III. Case-Study: OS Regular Activity Monitoring

In order to have a complete image over the platform utility, here are presented some significant results when monitoring different OSs running a usual processor-test (a hard-encoded MPEG video file playback) This short case-study is taken from a more complex research (which includes, for instance, a performance improving analysis), but this will be subject of a future work..

A. The Test Configuration

Such a procedure may not give interesting results when the hardware resources are generous (because the processing power is enough to handle without problems the software decoding process), this being the reason why the authors performed all tests on simulated limited-power architectures.

As host machine the authors used a medium-performance platform based on Intel Celeron D352 processor, with 1GB RAM and SATA2 HDD, with dual boot feature (SuSE Linux and Windows XP Pro), integrated as client for the previously described network.

The software solution adopted for VMM was VMware Workstation 6.0 (running both under Linux and Windows host machines) coupled with Veeam Monitor analyzing tool. This configuration allows a perfect integration with the VMware virtualization layer, so the platform can read and present a wide range of operating parameters related to VMs – kernel errors/traps, CPU, memory, disk, network and pagefile usage statistics. In this hosting context, three identical virtual machines were created (each emulating an Intel Celeron platform with 256MB RAM and IDE HDD storage) and configured with typical installations of SuSE Linux 8.2, Ubuntu 7.2 and Windows XP Professional. In order to have a complete functional environment, all VMs include VMware Tools platform. All OS images were stored on the Proliant server and NFS exported over the network.

B. The Testing Scenario and Results

In the context of normal computer use, dealing with multimedia applications is also a regular task. But when the file to be played is hard-encoded (i.e. high bitrates/video resolutions), the CPU usage may become problematic if the processor performances are below a specific limit, so it is of high interest to know how an OS acts in order to keep the entire system in acceptable working state.

The tests performed by authors consisted in playing a local MPEG4 video file encoded with high bitrate (approx. 10 Mbps) by using the same version (1.0) of the famous MPlayer installed under SuSE Linux, Ubuntu and Windows XP VMs. There were two main scenarios, involving both separate tests (consecutively performed) and simultaneous (on all VMs in the same time), the results being presented and briefly commented here.

1) The 3 minutes testing session results (separate run)

Fig. 4 shows a comparison between CPU usage percent for the above mentioned OSs. By analyzing these graphs it is to mention that, although at the beginning in SuSE OS the CPU is used not more than 50%, after a few seconds the percent raises to a maximum of 85%. The Ubuntu machine acts totally different, with an initial peak of 92% processor usage but followed by a moderate behavior on long term (with lots of inverse peaks to 50%). The best response is obtained when using Windows XP, which acts between maximum 82% and minimum 31% CPU usage. In fact, this behavior is in total agreement with expectations, as time as high activity means lots of system calls (and for Linux OSs the rendering modules were compiled and integrated in kernel, unlike in Windows case).

The HDD usage during the playback process is depicted in Fig. 5. The SuSE Linux VM shows a uniform-time disc access, with long periods of inactivity (approx. 30 s) and high peaks between 8 and 22 Mbytes/s (read/write access). The same uniform behavior can be observed for Ubuntu, at shorter intervals (about 6 s), through moderate peaks (810 to 9216 Kbytes/s). Windows XP is non-predictable, having an almost continuous HDD access with maximum peaks of

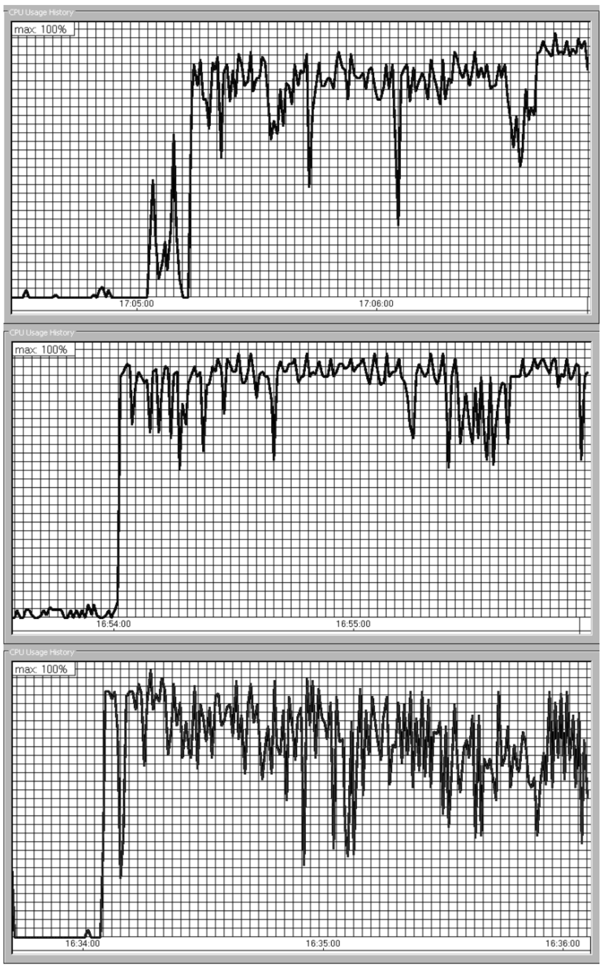

Fig. 4. CPU usage profiles for SuSE Linux (top), Ubuntu (middle) and Windows XP (bottom) – scenario 1.

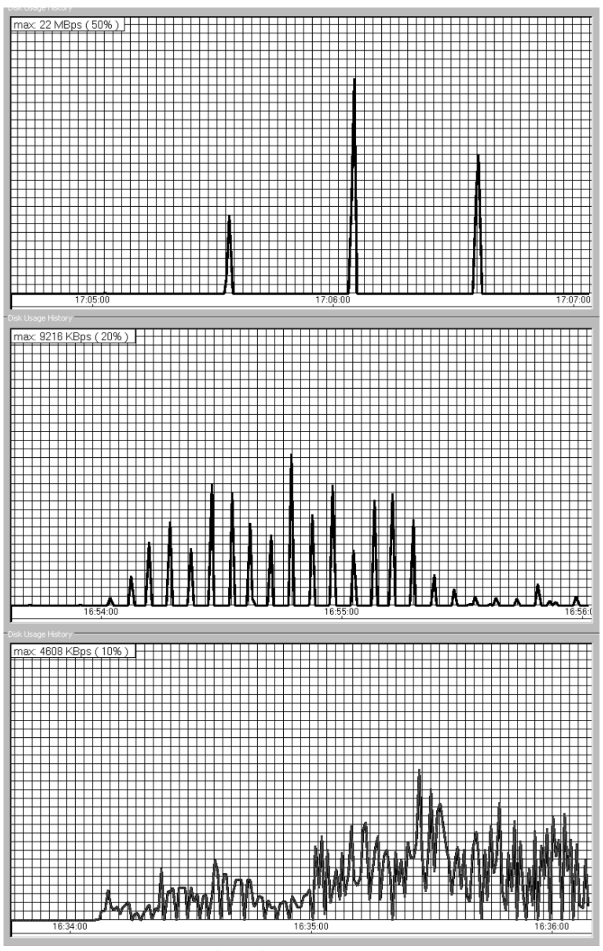

Fig. 5. Disk usage profiles for SuSE Linux (top), Ubuntu (middle) and Windows XP (bottom) – scenario 1.

4608 Kbytes/s (which may be not an optimal behavior).

As remark, the diagrams in Fig. 5 (and also in the following figures) are differently scaled for better details readability.

The authors further investigated this case and found that such an effect is induced by a different pagefile management (disc cache) style for the three OSs above mentioned.

SuSE Linux has an intense but shorter action on its swap partition (which keeps the pages ready to send into memory, respectively the dumped-to-file memory content) than Windows XP, although both have the same maximum access peak (2560 Kbytes/s), as seen in Fig. 6. During the last minute, for about 40 s, SuSE does not even use its disc cache, whereas Windows still keeps accessing the pagefile until the MPEG file playback finishes. Such a policy of *take it as needed* may be convenient when the hardware resources are continuously available during the working session, but may put serious problems when the limits are touched. Regarding Ubuntu, it has an almost inexistent swapping effect (maximum 512 Kbytes/s transfer speed) because it frequently access the HDD when reading small parts of the MPEG file and they seem to always fit into the free memory, with no need to swap memory pages.

2) The 3 minutes testing session results (concurrent run)

As shown above, the second scenario was based on simultaneously performing the video playback in all VMs, in order to reveal if the VMM has a good policy when trying to honestly allocate resources for the managed virtual computers.

In order to distinguish the stabilizing time at start, the MPlayer sessions were launched with short delays between them (approx. 3 s), first under Windows XP, Ubuntu and finally in the SuSE Linux VM, as Fig. 7 depicts.

It can be observed that, for all OS instances, after a short transient time the CPU usage practically follows the same average envelope, meaning that VMM (VMware) succeeds to fairly implement the resources allocation. It is also to remark the different profile shapes in comparison with the ones from Fig. 4. A careful look at Fig. 7 shows that, when – for instance – the Ubuntu machine CPU usage lowers, the remaining computing power is re-allocated to SuSE and Windows VMs, this way the overall efficiency being increased. There is no surprise when analyzing the HDD usage in this new context. Indeed, the main characteristics observed when performing the first scenario are preserved, excepting the maximum peak value which lowers to 9216 Kbytes/s (instead of 22 Mbytes/s) for the SuSE virtual

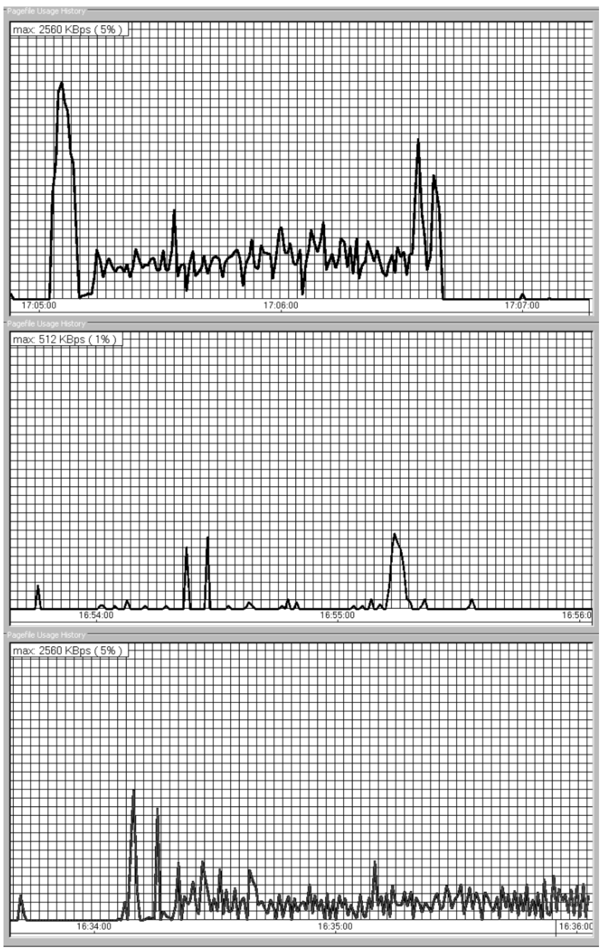

Fig. 6. Swap/pagefile usage profiles for SuSE Linux (top), Ubuntu (middle) and Windows XP (bottom) – scenario 1.

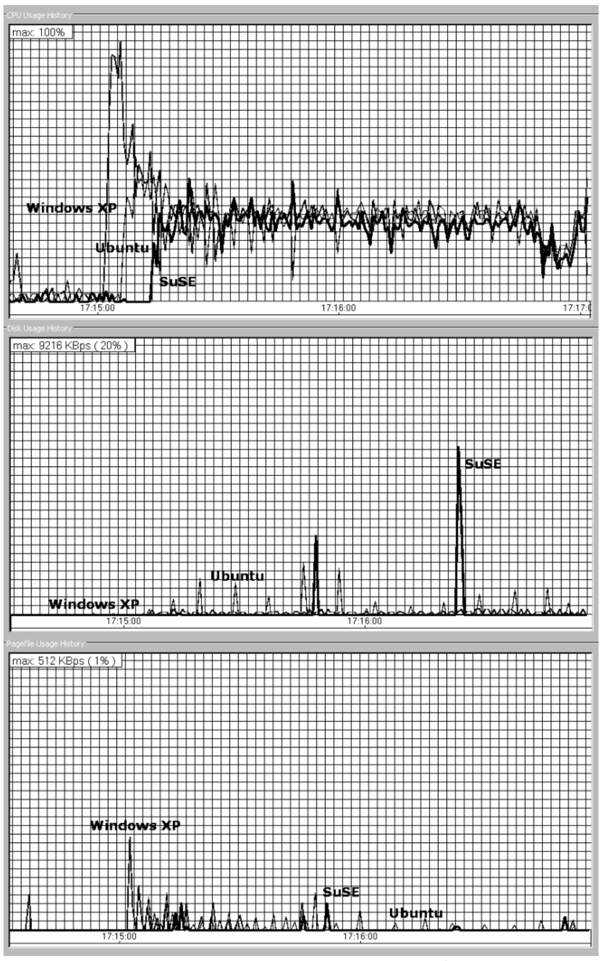

Fig. 7. CPU (top), HDD (middle) and swap/pagefile (bottom) usage profiles for SuSE Linux, Ubuntu and Windows XP VMs running simultaneously (scenario 2).

machine. In a similar way, the memory swapping (pagefile access) seems to differ only by the highest peak value (512 Kbytes/s instead of 2560 Kbytes/s – as for the first scenario).

IV. Conclusion

The complex hardware/software virtualization technology allows new and revolutionary approaches in computer science, especially in the field of operating systems practical studies. In this context, the present paper presented some selected results over the complex interactions between multiple operating systems (SuSE Linux, Ubuntu, Windows XP) and a homogeneous virtualized hardware platform. The studies were performed by using the non-invasive monitoring technique of VMBR. The behavioral characteristics of the three mentioned OSs are also outlined.

From the authors' point of view, extending such a research approach has a true practical relevance, as time as it may be used to reach a perfect fit between user's needs and the software environment particularities.

References

[1] G. Rădulescu, N. Paraschiv, "Virtualization techniques in computer science – a practical point of view", *Proc. of the UNIVERSITARIA SIMPRO 2006 (Petroşani) Symposium*, pp. 41-44, 2006.

[2] G. Rădulescu, N. Paraschiv, "Developing a virtualization techniques – based platform for advanced studies on operating systems", *The Petroleum-Gas Univ. of Ploieşti Bulletin*, Vol. LIX, Technical Series, No. 3, pp. 1-6, 2007.

[3] 5. R. P. Goldberg, "Survey of virtual machine research", *IEEE Computer*, pp. 34–45, June 1974.

[4] 9. J. Sugerman, J., G. Venkitachalam , B. H. Lim, "Virtualizing I/O devices on VMware Workstation's hosted virtual machine monitor", *Proc. of the 2001 USENIX Technical Conference*, June 2001.

[5] 8. C. P. Sapuntzakis *et al.*, "Optimizing the migration of virtual computers", *Proc. of the 2002 Symposium on Operating Systems Design and Implementation*, December 2002.

[6] 1. K. G. Anagnostakis *et al.*, "Detecting targeted attacks using shadow honeypots". *Proc. of the 2004 USENIX Security Symposium*, August 2005.

[7] O. Agensen, D. Detlefs, "Mixed-mode bytecode execution", *Technical Report SMLI TR-200-87*, Sun Microsystems, Inc., 2000.

[8] S. T. King, G. W. Dunlap, P. M. Chen, "Debugging operating systems with time-traveling virtual machines", *Proceedings of the USENIX Annual Technical Conference (USENIX'05)*, pp. 1-15, April 2005.

[9] A. Whitaker, R. S. Cox, S. D. Gribble, "Configuration debugging as search: finding the needle in the haystack", *Proc. of the 2004 OSDI Symposium*, December 2004.