A comparison of some Virtual Machines, Simulators and Emulators

Nikolas Galanis, Josep Llosa Espuny
{*nikolaos@ac.upc.edu, josepll@ac.upc.edu*}

Abstract

A common problem faced when starting working on a fresh project is making the choice of which of the available tools are the most useful and appropriate for the requirements of the work at hand. A lot of time is wasted trying and testing the various solutions until a decision is made.

The goal of this work is to present some of the most known Virtual Machine Monitors (VMM), emulators and system simulators. Through this presentation, the major characteristics of each of the selected tools will be pointed out, commenting on the advantages and disadvantages of each one and their possible areas of usage. Hopefully, this text will be able to provide users with adequate information about each of the tools examined, bringing them in the position to decide whether and in what degree, these tools serve their needs.

1 Introduction

The success of every research project, relies heavily on the correct choice of the set of tools to be employed for the realization of the work. In the area of computer architecture, simulators are undoubtedly the most commonly used research tools. Their abilities to provide the users with accurate and modifiable substitutes of expensive or otherwise hard to find hardware of fixed characteristics, has made their presence almost a necessity in every research. This need for such tools has led to a development of a significant number of both open source and commercial solutions. Their characteristics range from virtual machines that allow the user to run an OS over another existing OS providing various layers of isolation between the two, to full system simulators that provide hooks for every microarchitectural abstraction in order to suit the needs of the most demanding of users.

1.1 Virtual Machine Monitors

The Virtual Machine Monitors (VMMs) are programs that execute either directly over the hardware or over an existing operating system and provide the user with a new machine (virtual machine) that makes use of the underlying hardware. This virtual machine can be used to execute another Operating System (same or different to the existing one). This provides a relative security when modifying the OS of the virtual machine since any mishap would only affect the virtual machine and not the real one. Another popular use of the VMMs is resource partitioning, since a larger number of them executing over a real machine can provide individual users with access to one VM a subset of the real machine's resources.

1.2 System Emulators

System Emulator is a software that provides the functionality of a machine different to the one over which it is being executed. The use of these emulators is to give users the opportunity to run programs designed for other machines, to develop programs for machines they wouldn't otherwise have access to, as well as to be able to test and optimize their existing programs for execution on other systems. System emulators often sacrifice execution accuracy (though obviously not correctness) for speed gain.

1.3 System Simulators

The System Simulators are, in a way, more advanced system emulators, since they give the user access to the machine configuration as well as a number of hooks so that the user will have the ability to inspect closer the execution process, intervene and even modify it. Simulators are suited for hardware research since the users can introduce changes to the machine configuration and test them. The functionalities provided by simulators come at a great cost in execution speed, making them almost unusable for normal execution of applications for reasons other than benchmarking.

2 Virtutech's Simics

Simics, by Virtutech AB [1, 2, 3, 4, 5] is a commercial, system level instruction set simulator. It can be executed on Linux, Solaris and Windows 2000 and XP host systems. Simics supports a number of target systems including PowerPC, UltraSPARC (II, III and III Cu), Solaris, x86 and x86-64. This provides the user with the ability to simulate whichever of the target systems over anyone of the host systems. Simics can function both as an emulator and a simulator. As an emulator, it can execute a program designed for a specific machine by emulating one such machine. However, its true potential resides in its simulator abilities. Simics is a full system simulator providing the user with a big number of abstractions, hooks and functionalities thus greatly facilitating overview and control of program execution.

2.1 Interface

Simics can be executed both through the use of a graphical user interface (GUI) and through the use of command line instructions (command line interface - CLI). One basic difference of these two options is that the GUI provides some of the most useful functionalities in the form of buttons and menus, whereas in the CLI the specific instructions have to be executed. The greatest advantage of the CLI is that it can be extended by the introduction of new user-defined instructions. Other than that, the two options are identical. The user is not restrained as to how many instances of Simics he can have running on a certain host at any given time. The only restriction is that there cannot be more than one instances of Simics Central on one host. Simics Central is presented later in this document.

Some of the functionalities provided in the form of buttons by the GUI are the Start, Stop and Step buttons that start/resume, pause, and enter instruction-by-instruction execution mode respectively. Starting from the version 2 of Simics additional buttons are provided, such as one that causes the appearance of a window with the machine's registers and their current values, another that provides a disassembled code of an area around the instruction being executed, and a configuration button that provides information about the configuration of the simulated machine.

2.2 Checkpoints and Configurations

In order to save the state of a Simics instance, the user can save checkpoints and later load them, recovering thus the state of the simulated machine at the point the checkpoint was made. These checkpoints are incremental, meaning that for reasons of disk economy, each checkpoints only saves the changes made since the previous one. An immediate effect of that is that in order to be able to load a checkpoint, the user must also have all the previous ones.

Another way to maintain some of the changes since the last time, Simics gives the user the ability to save or load a configuration file. The format of this file is either Simics configuration code, or Python configuration code detailing the simulated machine configuration. The existence of Python code configuration script makes the configuration files easier to modify and process. The difference between these two formats however is a slight one as can be seen from the example below. We present the declaration of the **agp_bus0** object, first in Simics configuration code and then in Python.

Users can use this configuration to execute identical machines on the same or other host systems since the configuration language is portable, or change this configuration and load another, slightly changed version of the machine. An important characteristic of the configuration files is that such a file can be loaded while a machine is already being simulated (with the simulation paused), thus appending more functionalities to the target machine.

```
OBJECT agp_bus0 TYPE pci-bus {
conf_space: agp_bus0_conf
memory_space: pci_mem0
io_space: pci_bus0_io
bridge: pci_to_agp0
interrupt: ()
bus_number: 1
sub_bus_number: 255
send_interrupt_to_bridge: 1
pci_devices: ((0, 0, vga0))
}
```

Declaration of agp_bus0 object in Simics configuration code.

```
OBJECT("agp_bus0", "pci-bus",
conf_space = OBJ("agp_bus0_conf"),
memory_space = OBJ("pci_mem0"),
io_space = OBJ("pci_bus0_io"),
bridge = OBJ("pci_to_agp0"),
interrupt = [],
bus_number = 1,
sub_bus_number = 255,
send_interrupt_to_bridge = 1,
pci_devices = [[0, 0, OBJ("vga0")]]),
```

Declaration of agp_bus0 object in Python.

2.3 Loadable Modules

One of Simics' most important parameters is the support for loadable modules [2]. A module can be a piece of code describing a complete device, that can be compiled in Simics and then loaded into a target machine as a Simics module extending the target machine's functionalities or it can be any other type of extension that extends Simics in other ways. The distinction between a device and an extension is a very subtle one since modules could be both devices and extensions at the same time. Modules are usually written in C/C++, although Python can also be used. Simics comes with some simple sample modules that can be loaded directly into a simulation or used as templates for the design of more complex and complete ones.

Compiling a module is a simple process and the only thing necessary is the existence of some third-party applications (such as cygwin in the case of Windows). Loading and unloading a compiled module into a simulator is even more simple and consists only of executing a **load_module** (or **unload_module**)command. Then, the module can be inserted to the simulator configuration by executing a simple Python script in order to make the necessary interface connections with existing devices.

2.4 Networking

For networking purposes, Simics provides a virtual machine called **Simics Central**. It works as any other target machine with the only difference that

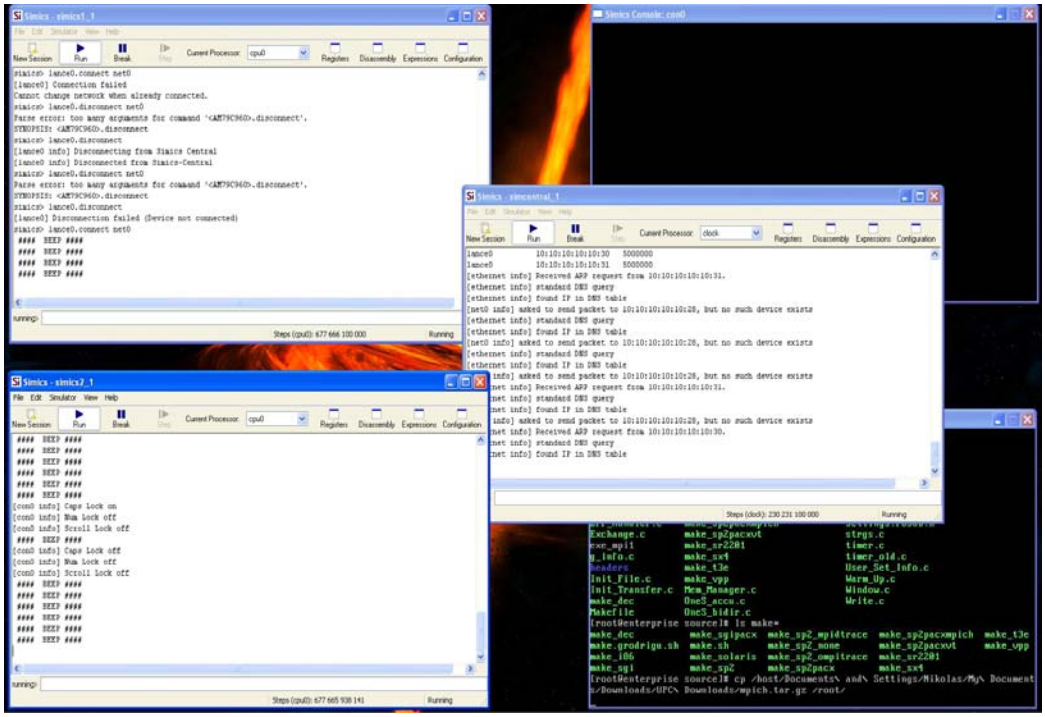

Figure 1: *Two instances of Simics connected with each other through an instance of Simics Central (central window).*

it lacks a CPU. Its core is an ethernet module called **ethernet-central**. Ethernet-central provides a virtual network to which other Simics instances can connect. This connection is equivalent to connecting a network card to a switch using a cable. Simics Central implements *ARP*, *DHCP* and *DNS*.

However, in order for two or more simulated machines on the same host to communicate with each other, no separate Simics Central instance is required. As long as the first machine is started with the option "Local - Allow other to connect" and the subsequent machines with the option "Remote - Connect to other Simics" in the network tab, the first one will start with an incorporated Central module and the subsequent ones will automatically detect it and connect there. After that, an IP address adjustment may be necessary before all machines can communicate with each other.

A restriction of Simics Central is that it cannot route packages to and from the host system that it is running on, meaning that the host system cannot communicate with the virtual network. Another restriction is that WinPCap doesn't support multiprocessor hosts, so the Windows-hosted version of Sim-

ics Central cannot connect to the real network if it is run on a multiprocessor host.

Using Simics Central as a gateway, simulated machines can connect to a real network. Machines on the real network should also be configured to use ethernet-central as a gateway in order to communicate with machines in the simulated network. For that purpose they should define a route to the Simics Central on the host machine.

2.5 Debugging

Simics comes with an important set of debugging functionalities, facilitating program execution overview and control. Execution can be stopped at any time and the machine state can be inspected or even altered. Simics provides the user with tools to view the current register values and even change them, provides disassembled code for an area of the code around the currently executed instruction as well as other functionalities. Execution can also be continued in lock-step execution mode, executing instructions one-by-one or telling Simics to execute a fixed number of instructions and then stop again. A very useful option given to the user is the introduction of automatic *break-points* that can pause the execution when a particular instruction is met or when a certain resource (e.g. a register) is accessed. Finally, Simics has a number of logging and tracing mechanisms that enable the user to see a detailed description of the execution process by printing all instruction memory accesses and exceptions as they occur. This tracing however, comes at the cost of a great reduction of execution speed, making its use prohibitive for extensive execution areas.

2.6 Simics Timing

The timing accuracy of Simics can vary according to the configuration. It can range from inaccurate, though fast processor timing, to accurate, but significantly slower clock simulation. Fundamentally, Simics is an *event driven* simulator with a maximum time resolution of a clock cycle. Simics, by default is configured to deliver performance and not simulation accuracy. For this, in the in-order execution mode, one instruction takes exactly one clock cycle. The out-of-order execution mode is implemented by a loadable module which assumes control of the execution latencies of the different instruction

phases. This module is configured to be called every simulated cycle and manages the simulation using an interface called Micro-Architectural Interface (MAI) [3]. It is up to the individual user to develop a timing model that better suits his/her needs. This can be done using some sample timing models available in Simics, either as they are or as guidelines to develop something more complex.

When simulating a single processor, there is one notion of time, so the time an execution needs to complete is easily calculated. The real problem surfaces when simulating a multiprocessor machine. In this case, each of the processors can have a different notion of time than the rest. Since perfect synchronization of the processors to achieve parallel execution is extremely slow, Simics chooses to serialize execution. This means that time is divided into segments and subsequently each segment is divided into a number of quantums identical to the number of processors. Each processor is then allocated a quantum of execution time. At the end of every segment, all of the processors have advanced the same, so we have a concise state. Since the order of the distribution of the different quantums to the processors is not defined, communication between two different processors shouldn't span more than one quantum if causality is to be preserved.

As far as memory-space is concerned, there are two different interfaces. A *timing model* and a *snoop device*. The first, gives the user access to the memory transactions before the memory space is accessed and can be used to stall transactions. This interface allows the user to model a memory hierarchy. The second, gives access to the memory transactions after Simics has accessed the memory and can be used to see the loaded or stored values and even change them. It does not allow stalling however. This interface can be used to create a trace of memory transactions.

2.7 Documentation

There is a great volume of documentation provided in electronic form with Simics. However, at least the main documents (Simics User Guide[1], Simics Programming Guide[2]) leave a lot to be considered well-written. Each chapter appears to have been written by a different person, some of which show certain difficulties managing the english language. Moreover, the programming examples are pretty naive and simplistic and do not really help the user understand the programming methodology required by Simics. However, it has to be commented, that the documentation of Simics v2.0.16 is a great

improvement compared to that of the v1.6.11. The point is that one should expect a little more from a commercial application, and more so from one sold at a relative high price, as is Simics.

2.8 Evaluation

To sum things up, Simics appears to be a very strong tool, one of the strongest, if not the strongest in this area. It has a lot of functionalities to cover the needs of most simulation projects and at the same time its interface makes it simple enough for easily carrying out simpler, less demanding jobs without requiring much effort from the inexperienced users. Looking deeper into Simics however, is an entirely different thing. Its code can be very complex at times, and along with the disapointing documentation on Simics Programming and the fact that the users have access to a small part of the source code (the price of Simics beeing a commercial product) Simics could easily become a headache for people trying to develope their own modules.

Maybe the best thing Simics has to offer is the number of processors that it can simulate, thus giving users the oportunity to work on a machine they could not otherwise have access to. However, by not giving the users the ability to develop their own processor modules as well as its very expensive price (except for the one year, renewable, free academic licences) make it lose some ground in the peference of at least the scientific community, who may opt for a free and hugely modifiable open source solution.

3 VMWare

VMWare [6] is a company that develops products in the area of virtualization. There are three main products provided by VMWare: the VMWare Workstation, the VMWare GSX Server and the VMWare ESX Server. These three products are examined closer in the following paragraphs. Weight will be placed more on VMWare Workstation and less on the other two products, since the Workstation is closer to the rest of the tools that are examined in this document. We will thus begin with a description of the two Server products and move on to a more detailed description of the Workstation.

3.1 VMWare Workstation

One of the most popular commercial solutions in the area of Virtual Machines for users is VMWare Workstation. Workstation, is a machine emulator that runs on x86 machines, emulating a machine of the same type (x86) to the host. VMWare Workstation can be run on Windows and Linux and provides the user with a virtual machine based on the host. One of Workstation's characteristics is its emulation speed, that closely matches the host's execution speed. This is made possible by passing the majority of the instructions for native execution. This obviously happens for instructions that won't affect the state of the host machine, otherwise there wouldn't be any level of isolation between the host and the guest. The entire virtual machine is represented by a single file in the host's hard drive. This file can be duplicated, giving two identical VMs or can be moved to another machine and continue execution there. This happens since, although the Workstation's virtual machines rely on the underlying hardware for execution, it is completely independent of it.

3.1.1 Interface

VMWare Workstation has a Graphical User Interface through which the user can create a new VM, run an existing one (by choosing the relative file) and modify the configuration of a VM. Using the GUI, the user has quick access to the management of the peripherals loaded in his virtual machine, such as disks, cdroms etc, as well as access to other functionalities such as taking a snapshot of the Virtual Machine. A snapshot saves the state of the virtual

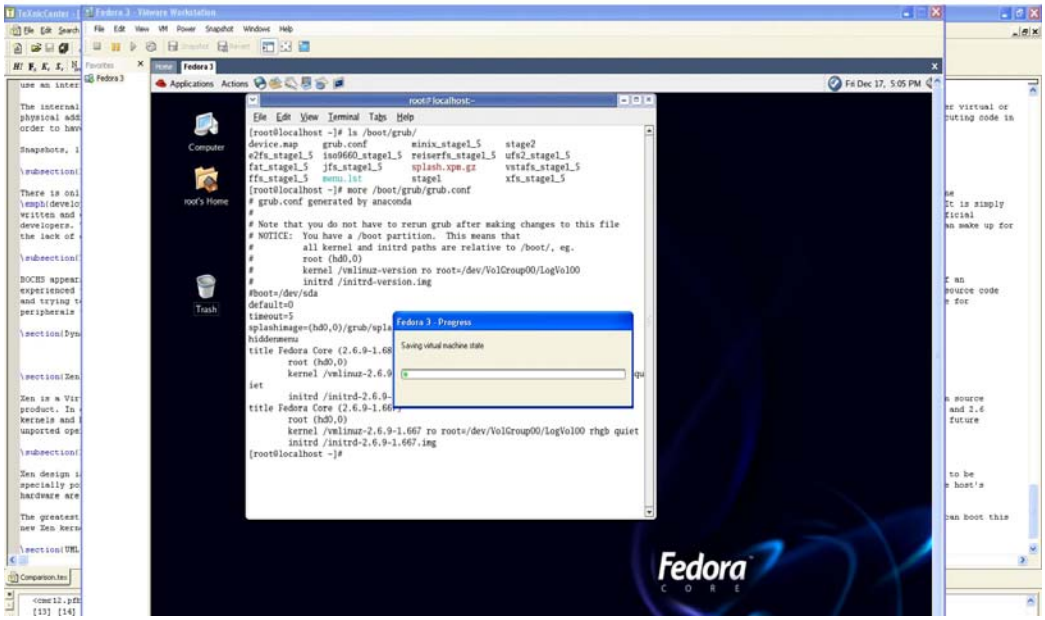

Figure 2: *VMWare Workstation running Fedora Core 3 on a Windows host.*

machine -the data on the disks, and whether the VM was on, off or suspended at that present time, so that the precise state can be recovered later.

3.1.2 Peripherals

Workstation makes use of the host's peripherals, enabling the virtual machines to use them as if they belonged to them. So, for a cdrom drive, a VM could use the host's cdrom, as well as load an image file as a virtual cd. This gives an important degree of flexibility, as far as data transfer is concerned, to and from the virtual machine. As a hard drive, the most common option is to use a virtual disk. However, VMWare Workstation gives the user the ability to use "raw disk", in other words, to use a real disk or a real partition as a hard drive. This is especially useful when setting up multi-boot systems.

Partial emulation of bidirectional PS/2 (parallel) ports is supported and can be used by a variety of devices such as printers, scanners and disk drives. A virtual machine can also be set up to use up to four virtual serial ports. A

virtual serial port can be connected to a real serial port, to a file on the host, or to a port of another virtual machine or that of an application. There is also a two-port USB 1.1 controller that can be used to connect peripherals if the host also supports USB. Finally, generic SCSI support enables the guest to access SCSI machines on the host, such as drives, scanners and hard disks.

Since a virtual machine is nothing but a file in the host computer, great care has to be taken when acting upon the real hard drive. A badly fragmented VM file is likely to produce important speed reductions, due to the trouble the host machine faces when trying to read from it.

3.1.3 Networking

In order to add networking to the VMs, Workstation makes use of the host's network. There are three networking modes available:

1. Bridged Networking,
2. Network Address translation (NAT), and
3. Host-Only Networking.

The bridge in the first option, connects the VM's network adapter to the host's network adapter, enabling the VM to connect to the LAN of the local machine. NAT is used when there is no IP address available in the host network for use by the VM, so the host IP has to be shared between the host and the guest. NAT provides the guest with a unique IP that can be distinguished by the host, thus enabling them to use the same IP when communicating with the real network. In the third option a network connection is provided between the VM and the host computer through a virtual Ethernet adapter that is visible to the host operating system. Finally, it is worth mentioning that Workstation can create up to nine virtual switches in order to connect one or more Virtual Machines.

With all these options mentioned above, VMWare Workstation users can create complex virtual network configurations with or without access to the internet. All that has to be done is start a number of virtual machines on the same, or on different hosts and connect them as suitable.

3.1.4 VMTools

Once a virtual machine is up and running with an operating system installed, VMTools should be downloaded and incorporated to that virtual machine. The installation of VMTools requires an installed operating system in the virtual machine, since they are depended on it. VMTools greatly enhance the functionalities of the virtual machine. The most important additions are:

- Addition of SVGA drivers,
- support for shared files and folders as well as drag and drop operations between the host and the guest,
- time synchronization between guest and host,
- automatic grabbing and releasing of the cursor once it is over the VM's window,
- copy and paste between host and guest and
- improved mouse performance for some operating systems.

Apparently, there may be a problem when installing VMTools, at least for a linux guest, since linux may not be able to detect the mouse afterwards, thus refusing to run the Xserver. This can be repaired, by ininstalling VMTools and modify the file *vmware-config-tools.pl* before reinstalling them. However, it certainly seems strange, a set of tools that supposedly enhances graphic and mouse performance should cause such a problem. Again, we mention that this has happened when installing VMTools in a Fedora Core 2 and 3 guest machine on a Windows XP host. In other host-guest combinations this problem hopefully will not appear.

3.1.5 Documentation

VMWare Workstation comes with a thorough manual [6] that covers all of the program's functionalities in detail. It really is a complete documentation and by far the best compared to the documentation available for the rest of the applications examined in this document.

3.2 VMWare GSX Server

GSX Server is VMWare's "middle" child. It is intended for virtualization and management of department-level machines. It can run on Linux or Windows and its installation process is a very simple one. Once up and running, a number of virtual machines, each capable of executing an Operating System of its own, can be created so that they will execute over a single underlying real machine, thus providing a kind of resource partitioning and a level of isolation between the user and the real machine with its Operating System. This is an ideal solution for departments and companies wanting to make better use of their infrastructure as well as introduce a certain degree of security, by providing each user with a working environment isolated from the real hardware.

However, GSX Server comes with some disadvantages mainly in the form of restrictions inherent to its nature and design decision to target mainly relatively small businesses. The main disadvantage of GSX Server is that since it runs over an existing Operating System it doesn't have any control on the computer resources. Resource management is done by the native OS and the Virtual Machine has to make do with what is given to it.

3.3 ESX Server

ESX Server is a more advanced and professional version of the GSX Server, providing users with much more functionalities and resource management options. It appears as the ideal tool for every system administrator, providing easy, fast and safe resource management. Its main difference to the GSX Server, is that it executes directly over the native machine's hardware. This means that the resource management is left entirely to the user. Up to 4 CPUs can be assigned to each VM and up to 10 VM can be executing at the same time on a single CPU. The administrator has also the option to manage network hardware and can group up to 10 network adapters and manage them as if they were one.

The functionalities of the ESX Server are greatly enhanced by the use of another VMWare product called VirtualCenter, that mainly deals with the management part of the virtual machines. The Virtual center is a tool for both the GSX and the ESX servers. It provides easy to use graphical interface, giving a single-screen view of all the VMs, along with information about system availability, resource usage and demand by each of the VMs.

However, the most interesting feature of the VirtualCenter is VMotion, that permits the migration of one VM running on a CPU to another CPU. This is done while the VM is running and without disrupting its execution or the user of that VM being affected in any way. This is a great asset, since in the case of something unexpected happening to one CPU, the VMs running on it could be migrated to other processors during the maintainance of the faulty one, and then back again, without affecting the users.

3.4 Evaluation

All in all, VMWare Workstation is a complete solution for small scale virtualization. It is easy to install, fast, relatively cheap and reliable. Its documentation is thorough and well written. From the products examined in this document, Workstation is the one proposed to the user who wants to run a different OS to the one installed on his or her machine and would prefer sacrificing a small portion of execution speed in order not to have to rearrange his partitions. Moreover, Workstation's real benefits are beeing able to run two or more operation synstems at the same time with the ability to easily move data from one to the other, as well as the level of isolation provided between a guest machine and the host. This level gives users the freedom to experiment with sensitive code on the guest machine withour running the risk of damaging the host. Thus VMWare Workstation is an ideal tool for those who enjoy "playing" with the code of their OS.

4 BOCHS

BOCHS [7] is an open source x86 system simulator with experimental support for AMD64 guests. It runs on x86 and non-x86 (Solaris, PowerPC, Alpha, MIPS) systems, running Windows, Linux, OpenBSD, FreeBSD or BeOS. It is one of the few non-commercial solutions if someone wants to run x86 code on non-x86 machines. Beeing open source, BOCHS presents a series of advantages, that have to do mainly with the user having the source code, which is in C++, available and beeing able to freely modify it according to his or her needs. Although everything in BOCHS is simulated as fast as possible, the weight has been thrown on accurate simulation of the guest machine so it is inevitably slow.

4.1 Interface

BOCHS uses a very simple interface, with buttons for inserting/ejecting cdroms and floppies, for enabling the mouse within the emulated machine, for copy-paste between host and guest, for taking snapshots of the simulated machine, for some runtime configuration options and for resetting and powering off the emulation. All in all it is a simple interface - much simpler than the ones met in similar commercial products. However, it is functional and provides the most frequently used options in the form of buttons.

4.2 Function

All of BOCHS' storage devices have the form of image files. The hard disks are image files, as well as the floppy disks and the cdrom drives. All the image files to be used are declared in the configuration file of BOCHS that is named bochsrc. The greatest problem of this process is that the image files have to be declared along with its exact size parameters such as number of cylinders, heads and sectors-per-track. This complicates things when trying to use a third person's image file without these parameters available.

BOCHS comes with a set of tools named *bochs tools* that include a program called **bximage** that is especially designed to create empty image files of disks that can be loaded in a guest machine in BOCHS. **bximage** provides all the parameters that have to be declared in the configuration file for the image to be successfully loaded.

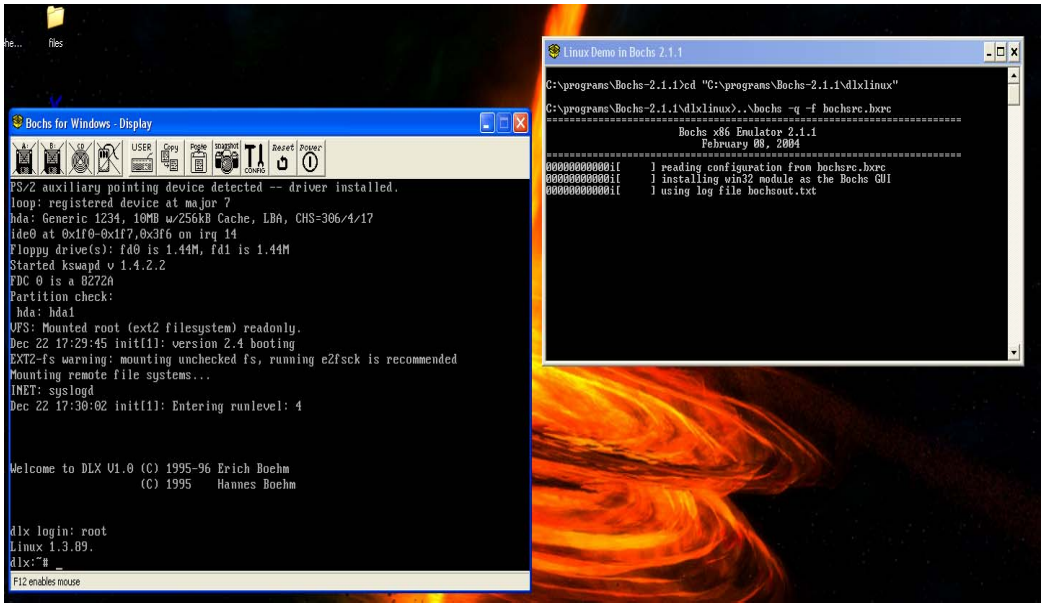

Figure 3: *BOCHS running a DLX Linux demo (provided with BOCHS-2.1.1).*

Once the image files are in place and the configuration file is complete, an emulation can be started. This can be done in two ways: either execute BOCHS and from the interface choose the configuration file to load, or rename the configuration file to **.bxsrc* and directly execute the configuration file. The two ways are identical as to their results, however the second is simpler and faster. BOCHS' web page has some image files of hard drives that have pre-installed operating systems. It is also possible to create a blank disc and boot from an image of a cd with an OS and make a clean installation of that OS. However, the process of the installation is extremely slow and there is a relative big possibility of crashes, especially if graphic environments are beeing installed. It is of great importance to have a very clear idea of the guest configuration and a great deal of patience in order to successfully manage to install an operating system.

4.3 Networking

The emulation of a NE2000 compatible network card, enables the simulated machines to connect to a network. On UNIX machines BOCHS networking cannot talk to the host machine. This means that the host cannot be used as a gateway between the guest and a real network. Another machine of the

network has to be assigned as a gateway for the guest. This restriction does not apply on win32 systems.

Apart from the network adapter emulation, BOCHS has support for USB and serial ports that can be connected to the real ports or to files for communication.

4.4 Utilities

Some of the most basic utilities provided by simulators and emulators are present in BOCHS. So the user has the ability to copy and paste text between the guest and the host, take snapshots and also use an internal debugger provided in BOCHS for his or her programs.

The internal debugger of BOCHS allows to set breakpoints and step through instructions for a closer control of execution flow. There is also the ability to keep track of memory using either virtual or physical addressing to view and manipulate the contents of a particular memory location. Register values can also be viewed and changed. Finally there is the option to disassemble the executing code in order to have a better view of the instructions beeing executed.

Snapshots, like in the case of VMWare Workstation, make a copy of the state of the simulated machine so that this state could be completely restored later if it is needed.

4.5 Documentation

There is online documentation available at BOCHS' web site that can be found at ***http://bochs.sourceforge.net***. It is divided into two html documents: one *user guide* and one *developer's guide*. the user guide is a very good introduction to the basixs of BOCHS and helps the users to install, setup and understand the basic functionalities of the program. It is simply written and easy to understand. On the other hand, the developer's guide is far from complete, as the vast majority of the document is incomplete, or is part of the future plans of the official developers. This means that the user who is willing to have a deeper understanding of BOCHS will have to read through the source code as well as check the BOCHS fora. Beeing open-source can make up for the lack of consise documentation this way,

even though it becomes a bit harder for users aspiring to become BOCHS developers.

4.6 Evaluation

BOCHS appears to be a very potent tool. Although it appears less appealing than some of its commercial counterparts, its open source nature makes it a really powerfull tool in the hands of an experienced user. Becoming an experienced BOCHS user/developer is the hard part though. Lack of official documentation means that users have to dedicate a great deal of time studying the source code and trying to find third party documents in order to achieve the required level. Once there however, the user will have the freedom to tamper with the entire BOCHS code, changing from code for peripherals to the processor itself.

5 DynamoRIO

DynamoRio Runtime Introspection and Modification system [11] is something quite different to the rest of the programs examined here. Its difference resides more in its purpose and less in its nature. Its purpose is not to provide users with an isolated environment where they will be able to do anything they choose without endangering the host system. The purpose of DynamoRio is to provide a relative isolated execution environment that will allow it to interfere with the execution progress, modifying and dynamically optimizing it. DynamoRio is essentially a run-time dynamic optimizer, based on HP's Dynamo project [10].

5.1 DynamoRio Basics

When a program is executed under DynamoRio control, DynamoRIO creates an isolation layer between the host operating system and the execution layer, taking control of the execution process. Under DynamoRIO's control, program instructions never execute natively. Copies of the instructions are put into DynamoRIO's caches and executed from there. There are two such caches: a code block cach and a trace cache. In DynamoRIO, the program

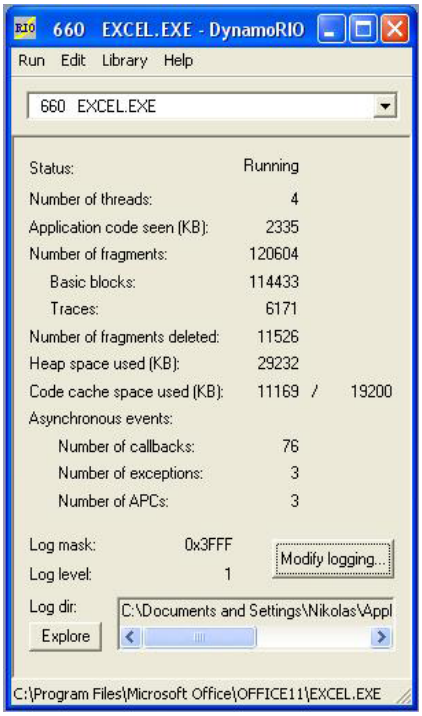

Figure 4: *The DynamoRIO interface for Windows, while executing Microsoft Excel 2003.*

instructions are interpreted and put into the block cache where they are organised in basic blocks. These blocks in the code cache are called fragments under DynamoRIO's terminology.

The whole process of interpretation allows DynamoRIO to identify hot traces in the execution (by incrementing counters every time a trace is identified). Once such a trace has been identified (the respective counter has surpassed a predefined level), execution enters *trace creation mode*. When in this mode, the blocks comprising the hot trace are copied into the trace cache.

As more and more fragments find their way into the trace cache, program execution is sped up since branches to hot traces are directly sent to the respective entries of the trace cache and thus served faster.

5.2 Evaluation

DynamoRIO runs on IA-32 systems and does not require any special OS support. There are both Windows and Linux versions and can be used with all native binaries without any need for prior modification. All these characteristics of DynamoRIO make it a very promising and easy to apply tool. Moreover, it is freely distributed for non-commercial applications.

A problem with DynamoRio is that it may be unable to execute a number of applications due to the existence of self-modifuing code which is apparently not supported. This is made even worse by the presence of many plugins with characteristics not supported by DynamoRIO, that make it impossible to execute applications otherwise supported by the program. For example, while DynamoRIO can execute Microsoft Excel 2003, if Adobe's pdfmaker is installed into Excel, the program crashes with a message warning of pdfmaker's failure to iniciate.

5.3 Documentation

The documentation of DynamoRio can be found either online or with the programs distribution files. It is extensive and relatively well presented. What really helps and presents a definite gain for the reader, is the extensive list of papers that have been publicated by DynamoRIO's developers that give an inside view of the heart of the program.

6 Xen

Xen [9] is a Virtual Machine Monitor that runs on x86 architectures. It is developed by the *Computer Laboratory* of Cambridge University. It can execute multiple virtual machines, each runing its own OS and achieve performance close to the native. It is an open source product. In order to function, Xen requires especially ported operaing systems. This is required so that high emulation speeds can be achieved. At present, there exist ports for Linux 2.4 and 2.6 kernels and NetBSD. Windows is not currently ported, nor are there any plans to do so due to licensing reasons. However, there are plans to make modifications to Xen's code so that in the future unported operating systems will be supported.

6.1 Xen Basics

Xen design is based on a virtualization technique called *paravirtualizaton*. It is a technique to permit high performance virtualization. For this reason, the operating systems have to be specially ported to Xen. Different virtual machines are called *domains* in Xen. Unlike other virtual machine monitors, Xen does not virtualize all the hardware. Instead, parts of the host's hardware are modified to work in Xen.

The greatest difference between Xen and other VMMs, is that Xen is installed on a system and creates a modified version of the running kernel (a Xen ported version). Afterwards, the user can boot this new Xen kernel and be able to start new domains, each one able to run a different operating system. From all the VMMs examined in this paper, Xen is the only one that requires tampering with the host kernel in order to execute the virtual machines. This could obviously cause some apprehension to some users unwilling to have their kernel modified, however this is the way Xen achieves its fast performance.

6.2 Documentation

Xen's documentation can be found in Xen's homepage [9], in HTML and pdf form. It is a very extended and thorough documentation. The user is guided through the instalation steps and all the aspects of Xen's functions are

explained in enough detail. It is one of the most complete documentations found for open source products.

6.3 Evaluation

Xen was not thoroughly examined since it was decided less apropriate for our research purposes, partly because other available open source solutions (e.g. BOCHS) could be executed over more host operating systems and partly because no modification to the host OS was required in order to execute the VMM. Although Xen's documentation is clearly superior to the other solutions', this alone was not enough to counterbalance the other factors. So Xen was examined only superficially.

7 UML

User-Mode Linux (UML) [8] is an open source Linux VMM project offering a safe and secure way of running Linux and Linux processes. It provides the user with a virtual machine with allocated hardware resources that can be a subset of the real available resources, or can even exceed the available ones.

7.1 UML Basics

The idea behind UML is to modify a Linux kernel so that a user will be able to call it from inside Linux. Essentially this means that a user without requiring root priviledges, will be able to run Linux with full priviledges inside the host's Linux. UML implements handlers and traps for the system calls sent from within the VMM so that they will not reach the host kernel, but they will be attended to by the user-executed kernel. In order be able to execute a UML kernel, the user will have to compile a specially patched Linux kernel for UML arcitecture (*ARCH=uml*). In the UML web page ***http://user-mode-linux.sourceforge.net/*** there are patches for the kernels, as well as already patched kernels available for download.

Since UML is a patched kernel, it retaines all the properties of the original kernel. This means that every time a new kernel is released, UML can immediatelly make use of its new properties. Also, since UML is nothing but

a patched Linux kernel, it is kernel code, but it can also be turned into a shared library so that other programs could link to it and use its functionalities, it can use stdin/stdout and it can be started as a subshell of an existing application.

As far a networking is concerned, the VMM can be connected to the Internet through the host using an unused IP address of the domain of the host.

7.2 Documentation

In the web page of UML, there is an extensive documentation on the application. From installation, to use, to development. There is enough material to help the users, however, there are important parts still missing, and waiting for volunteers to write them. Despite this it is a pretty good documentation for an open source program that usually suffer from poor or practically inexistent documentation.

7.3 Evaluation

Although UML seems like a simple and good idea, its implementation failed us while trying to install and execute it. An error message (*Outer trampoline didn't exit with SIGKILL*) appeared that after a search in fora and FAQs, we found out that this certain error had, supposedly, been dealt with, a couple of versions prior to the one we tried to install. We even tried patching the UML kernel with a number of patches proposed and offered by some other users but without any result. So, UML was not even tried.

8 Comparison

A true comparison between all the applications presented above is difficult, if not impossible to make, since what differentiates them, essentially, are their different characteristics, that make each of them suitable for different jobs compared to the others. That said, we have taken some speed measurements, using part of the SpecCPU2000 benchmark suite. A graph of the execution times attained by some configurations is presented below.

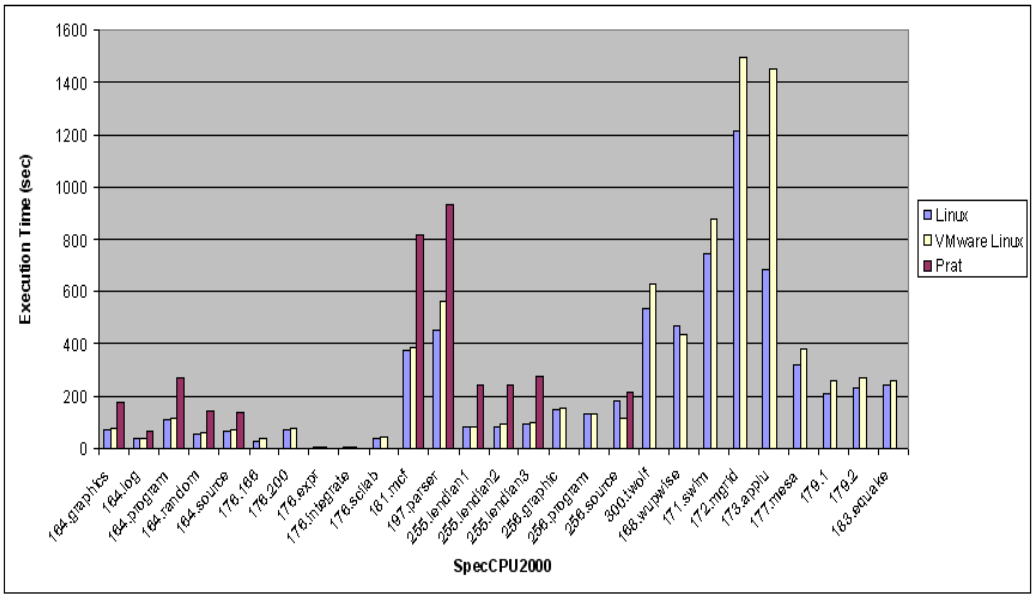

Figure 5: *SpecCPU2000 execution times for* ***Linux****: RedHat 9.0 running natively on an Intel Centrino 1,7GHz with 1GB RAM,* ***VMWare Linux****: RedHat 9.0 running on VMWare Workstation on Microsoft Windows XP on an Intel Centrino 1,7GHz with 1GB RAM,* ***Prat****: PCI, 4xIntel Itanium 800 Mhz 2MB Cache running RedHat 7.1*

Numbers for Prat for some of the benchmarks are missing from the chart because we were unable to compile these benchmarks for the configuration of that machine. Also, the same benchmarks were executed on Simics running on Windows on the same Centrino machine. Simics was simulating on one case Itanium with 512MB RAM, and on another one an x86 machine with 512MB RAM with and without a cache hierarchy. The reason that no numbers are presented here, is that the execution times were too high to be able to make all the executions required in time. For example, 164.gzip.source required 1h:26' to execute in Simics simulating an Itanium. Simulating an x86 system, times were similarly big, and adding a simple 2-level cache hierarchy timing model, execution time was gravely increased.

From the above graph, we can see that VMWare workstation offers a great solution for virtualization, since its performance is very close to that of native Linux. We should note here that all timing experiments were done with the system at normal working state, meaning that it was connected to the internet, with antivirus and firewall enabled and other processes running in the background. This was done in order to simulate a true working environment.

The rest of the applications presented, were not tested for a number of reasons. In the case of BOCHS, an attempt to install a linux distribution failed halfways (after about 4 hours) due to a BOCHS failure, so BOCHS was left for the time beeing, and the rest of the applications were ruled out as options for future use due to their characteristics and functionalities, so no testing was deemed necessary. BOCHS however, will robably be given a second chance in the near future since it appears as a very promising solution. This document will be updated as new data become available during the course of future investigation.

9 Conclusions

In this document, we presented the general characteristics of some of the most popular Virtual Machine Monitors and Simulators. Some of them were looked more closely than the others, due to their increased popularity, as well as due to their potential usefulness to our line of investigation. Some others are simply presented with a brief explanation of their characteristics. The purpose of this document is not to reach any conclusions. Its purpose is mainly to facilitate readers to distinguish between the presented applications and help them choose the most appropriate(s) for their purpose without having to waste time looking in different places for each one of them.

Obviously there was not time nor motives to look into each and everyone of the applications in the same level of detail, so interested readers are encouraged to have a look of their own at the respective web pages of the applications that interest them so as to get a more detailed presentation of each one.

References

[1] Virtutech AB: *Simics User Guide*, Simics 2.0.16, July 11, 2004

[2] Virtutech AB: *Simics Programming Guide*, Simics 2.0.16, July 11, 2004

[3] Virtutech AB: *Simics Micro-Architectural Interface*, Simics 2.0.16, July 11, 2004

[4] Virtutech AB: *Simics/x86 Target Guide*, Simics 2.0.16, July 11, 2004

[5] Virtutech AB: *Simics/IA-64 Target Guide*, Simics 2.0.16, July 11, 2004

[6] VMWare Inc.: *VMWare Workstation 4 User's Manual*, WS-ENG-Q203-003, 2004

[7] BOCHS Web Page: *http://bochs.sourceforge.net*

[8] UML Linux Web Page: *http://user-mode-linux.sourceforge.net/*

[9] XEN Web Page: *http://www.cl.cam.ac.uk/Research/SRG/netos/xen/*

[10] Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia: *Transparent Dynamic Optimization: The Design and Implementation of Dynamo*, HP Laboratories Cambridge, HPL-1999-78, June 1999

[11] Dynamo RIO Web Page: *http://www.cag.lcs.mit.edu/dynamorio*