A quantitative comparison of virtual network environments based on performance measurements

Walter M. Fuertes and Jorge E. López de Vergara

Dept. Ingeniería Informática, Escuela Politécnica Superior,
Av. Francisco Tomás y Valiente, 11,
Universidad Autónoma de Madrid, Madrid, Spain.
walter.fuertes@estudiante.uam.es, jorge.lopez_vergara@uam.es

Abstract. Virtualization tools can be an alternative to implement networking scenarios, with the purpose of making measurement experiments similar to those in real networks. For this, it is important to use the virtualization tool that consumes fewer resources. In this work, we have obtained a quantitative comparison of the CPU and memory consumption of those virtualization tools during the boot up and execution of a concrete virtual scenario.

Keywords: Virtualization tools, virtual machine, performance evaluation.

1. Introduction

At present, products and services have to be deployed in networks. However, these networks have other services already in use, and the deployment of new services can cause congestion to the networks, degrading some services and reducing their performance. At the same time, users require that these services assure a certain quality level. Nevertheless, it is uncertain what service quality can be provided a priori on these networks. Then, it is necessary to rely on an infrastructure that estimates the services' behavior, as well as measurement procedures and tools to make tests and to measure the delivered quality of these services.

As alternative we suggest the use of virtual computing platforms, reproducing real infrastructures in a virtual environment. This reduces the risk of damaging real networks, as well as the cost of the development and experimentation. At the same time, the service functionality can be tested, and the results obtained are going to be close to what should be measured in a real scenario, given that it is the real application, and not a simulated model, what is being executed. Even more, as shown in [1], virtualization results are very similar to simulation results. Within this context, the fundamental problem consists of identifying what virtualization tool consume less CPU and memory and is most suitable to make experiments related with the networking.

This research work is intended to verify the functionality of real network situations within virtual environments using a single computer. The result allows making diverse experiments in a virtual network platform to measure the performance and quality of deployed services. These experiments will provide valuable data to calculate the

required network resources. Then, traffic engineering, load balancing, or bandwidth reservation can be estimated based on these experiments.

In order to carry out this comparison, we first performed a synthesis of the bibliography concerning virtualization network and virtualization tools, summarized in next section. Then, we installed the previously enumerated tools, and designed a simple scenario to assess these virtualization tools, as shown in section 3. This work included the installation and configuration of several OS. Next, a set of tests was performed to obtain our evaluation results, presented in section 4. Finally, some conclusions are given based on obtained results.

2. State of the art

2.1. Virtualization tools

Since this research takes advantage of virtualization tools, it is important to make a brief description of the existing ones in the market. In our work we have just focused on the main open source tools that are executed under open source operating systems (OS), with the exception of VMware Server (currently free, but not open source), because of its widespread in Windows and Linux platforms.

It is important to remark that the similarity level between the virtual and real environment also depends on the virtualization technique. Although the industry uses diverse terms to describe these techniques, they are usually known as emulation, complete virtualization, paravirtualization, and OS level virtualization. In this study, we have included all these possibilities, by using the following virtualization tools:

- VNUML (Virtual Network User Mode Linux) [2] is an open-source general-purpose virtualization tool, designed to define and test complex network simulation scenarios based on the User Mode Linux (UML) [3] virtualization software.
- Netkit [4], also based on UML, has been conceived as an environment for setting up and performing networking experiments at low cost. It allows the creation of several virtual network devices that can be interconnected in order to develop a network on a single personal computer.
- VMware Server [5], as we mentioned previously, is a free virtualization product for Windows and Linux OS that implements full virtualization. It allows a physical computer to host some virtual machines, with different guest operating systems.
- Virtual Box [6] is a X86 virtualization software to deploy virtual machines, destined to desktop computers and enterprise Servers, which also implements full virtualization. It allows executing an OS without modification.
- Qemu [7] [8] is an open source generic emulator that reaches an acceptable emulation speed using dynamic translation. It executes virtual machines under Linux or Windows. It has several very useful commands to manage virtual machines.
- Xen [9] is an open source virtualization tool, based on the paravirtualization technique. It is distributed under the GNU/GPL license. It allows the execution of multiple instances of guest OS with all their characteristics.

2.2. Related works

Here we include some of the most relevant related works found in the literature: the work in [10] emphasizes on emulated networks with a complex topology, but just using Netkit. At the same time, reference [11] explains the use of virtualization techniques for the creation of complex network scenarios with VNUML. In [12], the authors made an evaluation of some tools, defining several comparison criteria, but using other virtualization tools. In [13] virtualization is introduced from a perspective of tests. However they do not provide any experiment results. In [14], several UML tools are analyzed, exploring their strengths and weaknesses and defining requirements for an ideal virtual network.

Although other works exist, which we have omitted for brevity, most of those included here present and compare their results qualitatively. The only exception is [12], but their quantitative measurements are just related to traffic (RTT, throughput) of the virtual machines. In our work, we have designed a test scenario, configured the networks and services, and then we have taken measurements related to CPU and memory consumption of the PC hosting the virtual scenario. We have also used the latest versions of the virtualization tools, comparing the results quantitatively.

3. Design and implementation

The assessment of these virtualization tools has consisted of measuring the performance of emulated virtual networking scenarios, and their interaction with the physical equipment, the CPU load and memory usage. For this, a network has been defined, organized into sub networks, as depicted in fig. 1, where a client/server application is running. The clients are *nix virtual machines. The server in this case is a web server, running in another virtual machine of another sub network.

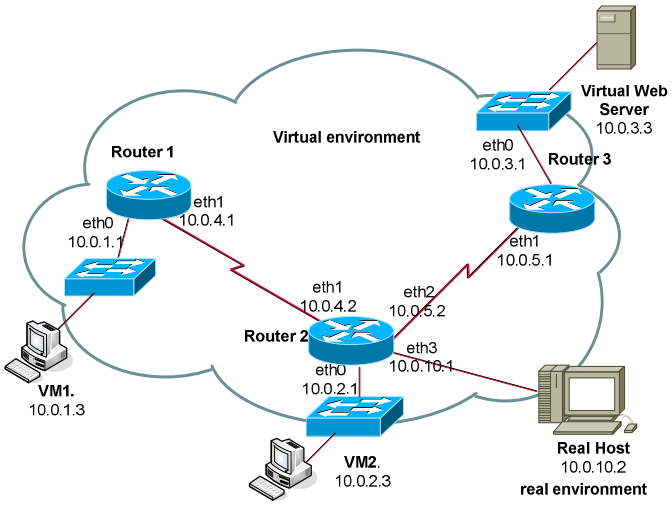

Fig. 1. Virtual network scenario used in the evaluation. We assigned 64 KB of memory to VM1, VM2, VM3 and 128 KB for those that works like routers

All tests have been developed on a Linux Debian 4.0 2.6.18, in Dual Core, 2.80 GHz, 1GB RAM and one Ext3 partition with 60 GB. We recorded transmissions from the real host to a virtual client and from the virtual server to the real host.

The measurements were taken with the following procedure:

1. First, we synchronized the clock with an NTP Server. Then, we installed each virtualization tools, created the first VM and installed guest OS. In this point, we tried to fix same file system and similar kernel on the all tools to more precision. This was possible for all tools except NetKit. Later we cloned the first VM to the remaining VMs, in order to reduce time. Finally we added virtual interfaces, configured IP address and started services.
2. Second we created and executed the respective programs that automatically constructed and started all scenarios, for each virtualization tool. In most cases it was necessary to configure IP address manually, except in VNUML and Netkit, because they provide an excellent functionality.
3. Finally, we implemented a unique shell script algorithm, which allows us the measurement of the CPU and memory consumption during boot up and execution.
4. Once initialized the scenario, as depicted in fig. 2, we have initiated a file transference to collect the measures in the execution stage, by using wget tool, iperf [15], netperf [16], w, free, top and tcpdump [17], with some options to avoid idles times, working in background and updates each second, during the transfer.

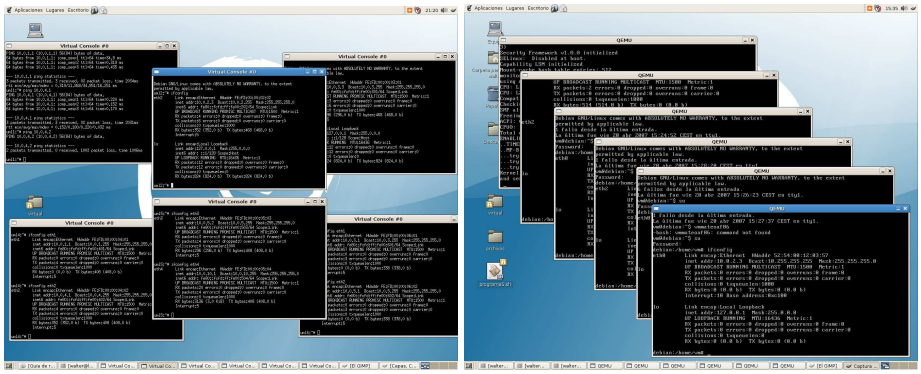

Fig. 2. Networking scenarios deployed with VNUML and Qemu.

4. Results

Table 1 shows the mean of the collected data, calculated from eleven tests performed with each virtualization tool, with top Linux command. These results show that VNUML is the less CPU consuming tool at boot up, with a 31.73%. It is followed by Xen, with a 35.14%; and Netkit, with 48.16%. Qemu is the most CPU consuming tool at boot up, with 63.10% of user time, and a total of 87.51%. It uses kqemu accelerator, executing the guest code directly on the host CPU. VMware registered 68.72% on system field. We have to clarify that the network scenario was implemented in this case by using the VMware Player. In total it consumes 85.49%. And finally, Virtual

Box is the unique one that uses nice field (64.48%), used when the process loses priority. In total it consumes 80.41%.

Table 1. CPU consumption when started scenarios.

	user	system	nice	idle	wa	Hw int	sw int	steal time
VNUML	10.78	20.75	0.00	60.29	0.13	0.05	0.03	0.00
Netkit	16.03	31.60	0.00	51.84	0.34	0.04	0.15	0.00
Qemu	63.10	23.95	0.00	10.91	0.18	0.06	0.22	0.00
Xen	2.04	4.17	0.00	58.86	22.01	0.03	0.13	6.76
Vmware	13.29	68.72	0.00	14.34	3.23	0.11	0.14	0.00
VirtualBox	3.15	4.64	65.48	19.37	6.94	0.08	0.12	0.00

Table 2 provides the memory usage. It shows that Netkit is the less memory consuming tool, with less than 185 MB. However, as stated before, the kernel version used with NetKit is different from the other ones. Apart from Virtual Box, the other virtualization tools need up to 286.76 MB to boot up.

Since that both Xen and VNUML are the less CPU consuming tools, we have performed other tests with them, included the tables 3 and 4. These tests show the CPU and memory usage during a file download from the virtual server to the real host. In both cases, Xen performs better than VNUML.

Table 2. Memory consumption when started scenarios

	Consumed	Buffers	Cache	Total (B)	Total (MB)
VNUML	613,652.98	89,257.40	247,900.77	276,494.81	270.01
Netkit	466,763.82	34,243.32	242,244.45	190.276.05	185.82
Qemu	717,314.60	18,740.75	451,260.44	247,313.41	241.52
Xen	345,967.69	11,539.29	124,614.41	209,814.00	204.90
Vmware	643,106.22	8,102.36	341,359.51	293,644.34	286.76
Virtual Box	810,683.96	2,610.21	123,162.98	684,910.77	668.86

Table 3. CPU consumptions during execution

	User	system	nice	idle	wa	hw int	sw int	Steal time
VNUML	12.95	36.64	0.02	41.90	7.38	0.03	2.07	0.00
Xen	0.17	1.80	0.00	62.80	32.11	0.04	1.81	0.15

Table 4. Memory consumption during execution

	Consumed	Buffers	Cache	Total (B)	Total (MB)
VNUML	887,291,75	5388.96	635,327.15	246,555.64	240.78
Xen	338,394.64	964.26	133,312.61	204,117.77	199.33

5. Conclusions

There are several virtualization tools that allow us the implementation of networking scenarios. All of them use the UML network utilities or the Bridge utilities, to support

virtual network devices. All of them need both hardware and software requirements to perform correctly.

The collected measures of CPU and memory consumption during the boot up and execution scenarios of each virtualization tool empirically demonstrate that these values are considerably dependant of their virtualization technique, as well as by each tool own features.

Xen and VNUML have been the tools with less consumption in the boot up experiment. Nevertheless, Xen has performed better than VNUML during the virtual network traffic experiment. In these terms, Xen would be the better virtualization tool to implement network scenarios.

It is also worth mentioning that VNUML and NetKit provide a unified tool to describe and deploy the network scenarios, reducing implementation costs. If this factor is important, then such tools should be borne in mind.

References

1. Alex Muñoz: Academic OPNET Research and Educational Projects, University of Basque Country, Departamento de Electrónica y Comunicaciones. http://det.bi.ehu.es/NQAS/opnet/
2. VNUML Virtual Network User Mode Linux, http://jungla.dit.upm.es/~vnuml/
3. The User-mode Linux Kernel Home Page, http://user-mode-linux.sourceforge.net/
4. NetKit, http://www.netkit.org/
5. Vmware Server, http://www.vmware.com/products/server/
6. Virtual Box, home page: http://www.virtualbox.org/
7. Qemu Open source processors emulator, http://fabrice.bellard.free.fr/qemu
8. Qemu, How to use Networks. http://www.h7.dion.ne.jp/~qemu-win/HowToNetwork-en.html
9. Xen, home page: http://www.xensource.com/
10. Massimo Rimondini: Emulation of Computer Networks with Netkit, Department of Computer science and Automatization of the University of Rome, Italy. January 2007.
11. Virtual Network User Mode Linux: Herramienta de creación de Escenarios de red basada en Virtualización para la Internet de Nueva Generación, Memoria Descriptiva. 6º Edición del Premio de Nuevas Aplicaciones de Internet. November 2006.
12. Andreas Grau, Harald Weinschrott, Christopher Schwarzer: Evaluating the Scalability of Virtual Machines for Use in Computer Network Emulation, Stuttgart University. Oct/06.
13. Swaminathan Seetharaman and Krishna Murthy: Test Optimization Using Software Virtualization, IEEE Software, September/October 2006, Vol. 23, No. 5, pp. 66-69
14. Jeroen Van Der and Gert Jan Verhoog: Virtual environments for networking experiments. Analytical Network Project, Technical Report, University of Amsterdam, Jul/04.
15. Iperf, http://dast.nlanr.net/Projects/Iperf/
16. Netperf, http://www.netperf.org/netperf/NetperfPage.html.
17. Tcpdump & libpcap, http://www.tcpdump.org/