Evaluating Xen and KVM Virtualization Platforms

Leopoldo A. F. Mauricio, Globo.com Inc., and Marcelo G. Rubinstein, Programa de Pós-Graduação em Engenharia Eletrônica (PEL), Universidade do Estado do Rio de Janeiro, RJ, Brazil

{leomauricio@gmail.com, rubi@uerj.br}

I. Introduction

Despite the great success, the Internet has difficult problems to be solved without performing a fundamental change into its architecture. It's not easy to change, for example, the semantics of location and identification of IP to solve mobility problems in the network, or to adapt stronger notions of identity on the Internet to solve its security problems. However, virtual environments permit to carry out tests of new protocols and alternatives to IP into production environments. These new solutions can be implemented using the infrastructure of real networks, which deal with production traffic. This happens because virtualization allows the protocol stack used in each virtual network to be completely different from that of the actual network, which could still run the TCP/IP and other Internet protocols as we know them. So, it is important to identify which virtualization platforms have good performance of CPU, RAM memory access, and I/O network.

II. Xen versus KVM

We have performed many real tests to qualitatively compare the performance of RAM memory access, CPU and I/O network of Xen 4.2 release, configured in routed mode, and KVM, which are hardware virtualization platforms. KVM is a virtualization platform that uses the open source machine emulator and virtualizer QEMU to abstract the native instruction set and create a map between the instructions of the guest OS and the host OS. Thus, each instruction sent by VMs (Virtual Machines) to the hardware is intercepted and handled by the QEMU emulator at runtime [1]. Xen is a hardware virtualization platform that allows the creation of virtual machines using the para-virtualization technique. In this technique, the hypervisor does not simulate physical devices to the VMs. For this reason, each virtual machine OS is changed to be executed with a privilege level that allows them to know all hardware addresses, including the addresses of other virtual domains. However, the device drivers of virtualized OSes interact with Xen hypervisor, which enables them to make direct controlled access to the physical devices. The hypervisor operates, for example, to ensure that each VM allocates only the amount of memory, CPU, and disk set for itself when it was created.

III. Results

We have performed tests using Dell PowerEdge 2950 servers with Quad-Core Intel Xeon processors, 4 MB caches per core processor, 8 GB RAMs, 876 GB hard drives, CentOSes release 6.4, 64-bit Linux kernels 2.6.32-358.2.1. Each experiment was performed 15 times, and a 95% Confidence Interval (CI) for the mean was also obtained.

First, we have performed CPU and memory tests to identify the hypervisor with lower overhead over them. For the CPU tests, we use the Super Pi [2] to calculate Pi to 2^{22} decimal digits. The memory tests used the STREAM benchmark [3] to measure the data transfer rate of one memory location to another, while complex arithmetic operations were performed.

Table 1: CPU and memory test results.

	Time (s)		Rate (Mbps)	
	Mean	CI	Mean	CI
Native Linux	74	0.1	5309	0.3
KVM	77	0.1	5181	2
Xen	323	4.6	4293	70

Table 1 shows that KVM, despite using QEMU to completely virtualize the hardware, performs better than Xen. This probably happens because KVM makes the Linux kernel act as a hypervisor, since the KVM code is integrated into the kernel code when KVM is enabled. At last, the network overhead was evaluated. We have used the Iperf tool in a scenario with three physical machines: one with/without a VM (KVM/Xen) to route packets, and the others running native Linux to generate and receive traffic.

Table 2: Network test results.

	Transmission rate (Mbps)		Reception rate (Mbps)	
	Mean	CI	Mean	CI
Native Linux	939	0.1	903	2
Standard KVM	168	0.5	139	0.2
VIRTIO KVM	939	0.1	888	0.7
Xen	880	1.1	553	6.9

We enabled the VIRTIO lib [4] to virtualize I/O operations into KVM. Thus, I/O network operations of KVM were performed by the kernel forwarding mechanism as if virtual interfaces were physical network interfaces. Results indicate that VIRTIO KVM performs better than Xen and standard KVM.

ACKNOWLEDGMENTS

The authors would like to thank Faperj and CNPq for partially funding this research.

References

[1] Red Hat, Inc., "KVM (Kernel Virtual Machine) documentation", April 2013. Available: http://www.linux-kvm.org/page/Documents.

[2] wPrime Systems Inc., "Super PI: A single threaded benchmark that calculates pi to a specific number of digits," December 2011. Available: http://www.superpi.net.

[3] J. D. McCalpin, "STREAM: Sustainable memory bandwidth in high performance computers," January 2013. Available: http://www.cs.virginia.edu/stream.

[4] VIRTIO, "VIRTIO: Using virtio_net for the guest NIC: throughput tests using Iperf," May, 2013. Available: http://www.linux-kvm.org/page/Using_VirtIO_NIC.