LA-UR-14-26182

Approved for public release; distribution is unlimited.

Title: Design of Performance Analysis of Virtual Systems for Cross Platform Testing

Author(s): Nielsen, Roy S.

Intended for: Final assignment for an operating systems class, will also need to be able to release on general requests.
Report

Issued: 2014-08-07 (rev.1)

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Design of Performance Analysis of Virtual Systems for Cross Platform Testing

Roy S. Nielsen

Los Alamos National Laboratory

Introduction

Using virtual machines for design and test is an every day occurrence on the development services team at Los Alamos National Laboratory(LANL). Our charter is to implement solutions relating to central services. We sometimes use Commercial Off The Shelf(COTS) solutions, and at times we develop solutions ourselves. Either way, we spend time testing the solution in our computing environment. In most scenarios, we use virtual machines for testing our localized solutions before final implementation in production. For projects that require cross platform testing, the problem can be large, complicated, and hard to manage. Also, in today's environment, we are all asked to do our job "better, faster, cheaper"(Sauser, Shenhar, Hoffman, 2005, 396). Performing this analysis will give us better and consistent control over our process, as well as reduce the overall test time and cost less money.

Background

National laboratories with oversight from the National Nuclear Security Administration are required to manage their computers diligently. We follow NIST, DISA, NSA, CIS and other guidance for securing computer systems. About a decade ago, we developed tools to harden systems to an amalgamation of these security standards. Some of these included the Security Tool On Mac(STOM)(Marcus, 2008) and Security Tool on RedHat(STOR)(Kennel, 2009). Running the security tool is required for a machine to be put on the network. The Los Alamos National Laboratory(LANL) unix community wanted support for other unix related operating systems, but we did not have the manpower to accomplish this.

We are now working on a security tool we call Security Tool On *nix or STONIX(Kennel, 2014). This tool will harden unix related systems using methods similar to STOM and STOR. STONIX will cover a wide variety operating systems, such as RedHat,

Ubuntu, Debian, BSD, Gentoo, Solaris, Mac OS and derivatives. As of now, developers test their own code in their own virtual machines, which leads to inconsistencies in testing process and results.

Motivation

Automation will be key to successfully integrating the wide variety of platforms that employees wish to have approved for use at the Laboratory. Building packages for each platform, distributing the product to a system, and testing the project on that platform can be time consuming and complicated.

The major performance concerns in this process are the automation and interaction with virtual guests. These processes include setup and teardown of the guest, distribution of the code to the guest, building and installing the package, then running the tests then reporting the results. We need a balance of programmability and speed of the virtual guest that can also be managed easily as we are short-handed and do not have time to develop intricate systems. We have a need to test technologies that can give us the best performance and manageability.

Types of Virtualization

There are four types of virtualization(Walters, J. P., Chaudhary, V., Cha, M., Guercio, S. Jr., Gallo, S., 2008). First is full virtualization that emulates hardware, controlled by a hypervisor. Second is paravirtualization however, the guest often requires changes to be hypervisor aware. Third is operating system level virtualization, where the operating system isolates an instance of another operating system without a hypervisor. Operating system level virtualization is similar to application, resource or security containers(Soltesz, S., Pötzl, H., Fiuczynski, M., Bavier, A., Peterson, L., 2007; Walters, J. P., et al., 2008) Last is native virtualization where there is hardware support for virtualization and no emulation is required.

The first and the last can sometimes be the same product, or stated as the same(Perera, 2011; Yang, C., Tseng, C., Chou, K., Tsaur, S., Hsu, C., Chen, S., 2010). For our purposes, we will consider full virtualization the same as bare-metal or native virtualization. The purpose of the paper “Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisor”(Soltesz, et al., 2007) is to compare operating system level virtualization with paravirtualization, leaving their study incomplete as it did not include the use of native virtualization.

Table 1

Examples of Virtual Systems

OS Level Virtualization	Paravirtualization	Native Virtualization
Docker (Janin, Vincent, Draffort, 2014)	Xen (Ranadive, A., Kesavan, M., Gavrilovska, A., Schwan, K., 2008; Lee, Y., Hsueh, C., Chang, R., 2011)	Xen (Zhang, X., Dong, Y., 2008)
LXC (Janin, et al., 2014)	KVM (Raghavendra, K. T., Vaddagiri, S., Dadhania, N., Fitzhardinge, J., 2011)	VMware ESX (Walters, J. P., et al., 2008)
PRoot (Janin, et al., 2014)	Proteus (Gilles, K., Groesbrink, S., Baldin, d., Kerstan, T.)	Proteus (Gilles, K., et al.)
VServer (Soltesz, et al., 2007)		

Design

Four Design Variables

There will be four variables to this performance test. We will test each combination as operating system end user licenses will allow.

Selection of guests. To get a good representation of what platforms STONIX supports, the guests that need to be tested in this study are the latest versions of RedHat, Ubuntu, Debian, FreeBSD, OpenSolaris, and Mac OS X. There are multiple distributions of Linux, BSD and Solaris related systems, but a majority of distributions we need to cover are related to one of the above. All virtual guests will be freshly installed unmodified operating systems, with only the basic toolsets, having just been provisioned with install media. The snapshot feature of virtual technologies will also be key, so we can revert to the freshly installed snapshot between test cycles. Interaction with the virtual guest must be able to be automated or automatable.

Selection of hosts. Host operating systems will be limited to RedHat, Mac OS X and VMware ESX. RedHat and Mac OS X are the most popular unix related operating systems at LANL currently. RedHat and Mac OS X will handle all operating system level virtualization. Redhat will be used to provide paravirtualization. RedHat and Mac OS X will need an application on top of the OS to perform operating system level virtualization and paravirtualization. With VMware ESX we will only test native virtualization.

Selection of virtual technologies. Docker will be the choice for operating system level virtualization, since it will work on RedHat or Mac OS X. We are looking for cross platform solutions as much as possible so we can decrease effort between operating systems. Not having a learning curve for a product for each operating system will make the infrastructure easier to manage. Xen has been chosen for paravirtualization, however it will only run on RedHat and not run on Mac OS X. VMware ESX will be used for native or bare-metal virtualization and does not require a host. It should be possible, although not necessarily supported, to host all guests with these virtualization technologies.

Selection of hardware. We need the hardware to include virtualization in the cpu so we can include hardware virtualization in the performance analysis. Since Intel leads the market (Graziano, M., Lanzi, A., Balzarotti, D. 2011) in processors and number of Intel chips support hardware virtualization, we will choose an Intel processor. To be compliant with the Mac OS X license, Mac OS X guests need to run on Apple hardware. The specs should include 32Gb of memory, a terabyte of flash storage and the maximum number of processors with the best specification for memory and storage. There are Intel processors in the Mac Pro and they support hardware virtualization. Mac laptops don't support more than 16Gb of RAM. This makes the Mac Pro the most appropriate choice for test hardware. The project would require three identical configurations, to split the project into three parts, one for each virtual technology.

Personnel

Several people working closely together will be required to accomplish this performance analysis. If they already have experience with two or more of the systems in Table 1, the performance analysis will take much less time as the learning curve to come up to speed with virtual systems can take time.

Tests for virtual systems

The test suite for guests will be the same across all test setups. This will help us choose the most optimal virtual system and host for our purposes. Running the STONIX product could be compared to a compile of a large project. Qt is our choice for compilation on the guests, as it is available for each platform, giving comparable results. The guest tests will include bringing up the virtual guest, distributing the code to the guest, compilation of Qt, and shutdown of the guest.

We will also run sets of tests from the phoronix test suite(Wen, Y., Lee, J., Liu, Z., Zheng, Q., Shi, W., Xu, S., Suh, T., 2013). Although Wen, et al., combined many of the phoronix tests into one test, we will break the testing into sections that mimic normal running conditions of servers and workstations. Adding these phoronix tests should not add a lot of time and will give general performance characteristics of these configurations for general use. This will give us the data needed for future endeavors.

Discussion

This study will help reduce the amount of time testing takes, as we currently manually move code to the target platform, then package the product then run the tests. Being able to manage a system that is easily automated is essential to meeting our goal. This project will make automation of the process of cross platform build, packaging and testing attainable. Reducing the flaws and inconsistencies in the testing process will help us make a better product and obtain more repeatable results. It will be essential for the code, methods and results to be published and freely available so others may benefit from this study.

Future Process Addition

After automation of this process, we intend to harness it with a Continuous Integration(CI)(Meyer, 2014, 14-16) product both for automation and reporting results. The first step will be having a nightly build, then review the results when we come in in the morning. Before planning the day's activities, the results would be reviewed and work assigned accordingly. The next step is to have the CI process fire off on the commit of code to the revision control system. A CI suite will build, test, log results and notify the team of a success or failure of testing. Combining the results of this study with CI will make our process "better, faster, cheaper"(Sauser, et al., 2005, 396).

References

- Sauser, B., Shenhar, A. J., Hoffman, E. J. (2005, July, 396) *Identifying differences in space programs*. Technology Management: A Unifying Discipline for Melting the Boundaries,
- Kennel, D. (2009,) *LANL-stor and Challenges of Evolutinoary Development*. Presented at the National Laboratory Information Technology conference, Oak Ridge, Tennessee, USA.
- Marcus, A. (2008, May) *Securing Macs in LANL's Environment*. National Laboratory Information Technology Summit, Chicago, Illinois, USA.
- Kennel, D. (2014, March) *STONIX – Can a hardening tool be Agile?* Presented at the National Laboratory Information Technology(NLIT) conference, San Francisco, California, USA.
- Soltész, S., Pötl, H., Fiuczynski, M., Bavier, A., Peterson, L. (2007, March) *Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors*. Paper presented at the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, Lisboa, Portugal.
- Walters, J. P., Chaudhary, V., Cha, M., Guercio, S. Jr., Gallo, S. (2008, March) *A Comparison of Virtualization Technologies for HPC**. Paper presented at the 22nd International Conference on Advanced Information Networking and Applications, Naha City, Okinawa, Japan.
- Perera, M., Chamath, K. (2011, September) *A performance comparison of hypervisors*. Paper presented at the International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka.
- Yang, C., Tseng, C., Chou, K., Tsaur, S., Hsu, C., Chen, S. (2010, May) *A Xen-Based Paravirtualization System toward Efficient High Performance Computing Environments*.

Presented at the Methods and Tools of Parallel Programming Multicomputers, Lecture Notes in Computer Science, Second Russia-Taiwan Symposium, Vladivostok, Russia.

Janin, Y., Vincent, C., Duraffort, R. (2014, June) *CARE, the comprehensive archiver for reproducible execution*. Paper presented at the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering, Edinburgh, UK.

Ranadive, A., Kesavan, M., Gavrilovska, A., Schwan, K. (2008, March) *Performance Implications of Virtualizing Multicore Cluster Machines*. Paper presented at the 2nd Workshop on System-level Virtualization for High Performance Computing (HPCVIRT'08), Glasgow, Scotland.

Zang, X., Dong, Y. (2008, January) *Optimizing Xen VMM Based on Intel® Virtualization Technology*. Paper presented at the International conference on Internet Computing in Science and Engineering, Harbin, Leilongjiang, China.

Raghavendra, K. T., Vaddagiri, S., Dadhanian, N., Fitzhardinge, J. (2011, September) *Paravirtualization for scalable Kernel-based Virtual Machine (KVM)*. Paper presented at the IEEE 13th International Conference on High Performance Computing and Communications (HPCC), Banff, Canada.

Lee, Y., Hsueh, C., Chang, R. (2011, August) *Inline Emulation for Paravirtualization Environment on Embedded Systems*. 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Toyama, Japan.

Gilles, K., Groesbrink, S., Baldin, D., Kerstan, T. (2013, June) *Proteus Hypervisor: Full Virtualization and Paravirtualization for Multi-core Embedded Systems*. 4th IFIP TC 10 International Embedded Systems Symposium, IESS 2013, Paderborn, Germany.

Graziano, M., Lanzi, A., Balzarotti, D. (2013, October) *Hypervisor Memory Forensics*. 16th International Symposium, RAID 2013, Rodney Bay, St. Lucia.

Wen, Y., Lee, J., Liu, Z., Zheng, Q., Shi, W., Xu, S., Suh, T. (2013, May) *Multi-Processor Architectural Support for Protecting Virtual Machine Privacy in Untrusted Cloud Environment*. Paper presented at the ACM International Conference on Computing Frontiers, Ischia, Italy.

Meyer, M., (2014, May/June) Continuous Integration and Its Tools. *IEEE SOFTWARE*, 14-16.

Links to products considered

<http://www.docker.com>

<https://linuxcontainers.org> (lxc)

<http://proot.me>

http://linux-vserver.org/Welcome_to_Linux-VServer.org

<http://www.xenproject.org>

<http://www.vmware.com>

http://www.linux-kvm.org/page/Main_Page

http://www.labcenter.com/products/vsm/vsm_overview.cfm (proteus)