Testing Virtual Machines for CADRC Server Hosting

A Senior Project

presented to

the Faculty of the Computer Science Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Robin Verweij

December, 2013

1. Introduction

As more and more operating systems are released along with better hardware, the need to test these in combination becomes harder and harder. The solution is elegantly simple - virtual machines. Virtual machines are the software implementation of physical machines. To put it bluntly, they are virtual computers. Users define the hardware and operating system to launch with and the physical hardware they're operating on emulates the desired output on screen.

These emulations have many advantages. Multiple OS environments can co-exist on the same computer with strong isolation from each other. The virtual machine can provide an instruction set architecture that is somewhat different from that of the real machine. VMs provide application provisioning, easy maintenance, high availability, and disaster recovery.

Along with these advantages come some disadvantages. A virtual machine is less efficient than a real machine when it accesses the real machine's physical hardware indirectly (such as the disk). When multiple VMs are concurrently running on the same physical host, each VM may exhibit a performance variance (of speed but not results) which is highly dependent on the workload imposed on the physical system by all VMs running.

Multiple VMs each running their own operating system (called guest operating system) are frequently used in server consolidation, where different services that used to run on individual machines to avoid interference are instead run in separate VMs on the same physical machine. Each of these physical machines has its own hypervisor - or virtual machine monitor - that creates and runs virtual machines.The hypervisor presents the operating system with a virtual operating platform and manages the execution of the operating system. Multiple instances of a variety of operating systems may share the virtualized hardware resources a hypervisor provides.

There also exist virtual machines that are headless. This means that the hypervisor can operate without monitor, keyboard, and mouse. These devices are usually controlled via network connections.

This paper compares six different virtual machine implementations and evaluates them against the requirements and constraints of a particular hosting facility. Because each VM developer created their software for different uses, and none for our desired use of being a server host, this paper will be focused on how each VM handles issues revolving around networking and minimizing performance loss.

2. Background

2.1 Our Criteria

Our main audience, the CADRC - the Collaborative Agent Design Research Center which is part of Cal Poly Research division - is interested in supporting research projects and providing a revenue stream for the Cal Poly research division. But it only has access to a few large capacity servers and that won't help them. This is because most research projects on campus are small and do not require the entirety of a large capacity server to run. Additionally, some projects could take 1-2 years to complete if they continually make little to no progress. Because of this we want to maximize utilization of our hardware (i.e. don't want it to over-provision projects with lots of power and lose a whole blade for a long time). The best way to do this is virtualization. Virtualization allows us to split the physical hardware resources amongst any number of virtual machines. Whenever someone requests resource time from the CADRC, a system admin can create a new virtual machine instance, give the requester all the information necessary to access that new instance, and the requester can immediately start working. Without virtualization, if all servers were in use when the request came in, the requester would have to wait until one became available before being able to start working.

There are few specific criteria that the CADRC server racks require to make sure that the virtual machine implementation they use fits their needs as best as possible. The reason the following criteria were chosen is because they all reflect what the CADRC is looking to bring to the table - an easy user experience for any end users that can be easily maintained without actually having to go to the server room. The criteria are:

- Hypervisor invisibility - we want to minimize performance drops that end users see due to multiple VMs being active under one hypervisor.
- Virtual machines are easily clonable via command line - this allows administration to be done remotely rather than in the server room.
- List of virtual machines is accessible via command line - this allows administration to be done remotely rather than in the server room.
- Snapshot capability - to allow for crash/disaster recovery.
- Hot swap capability - so that administrators can manipulate VMs on the fly without disrupting user experience.

2.2 Target Environment

The actual server rack that we're testing for is the Dell M1000e blade server system. This server system includes embedded modules for storage and I/O and is built for high availability. Each M1000e enclosure can hold up to 8 full-height blade servers (stripped down server computers with modular designs optimized to minimize the use of physical space and energy) that are easily swapped out. In addition, the enclosure itself has an innate redundancy system in place for fans, switches, and power supplies. This redundancy system is controlled by I/O fabrics (fabric is used as a metaphor to illustrate the idea that if someone were to document the system

components and their relationships on paper, the lines would weave back and forth so densely that the diagram would resemble a woven piece of cloth). The I/O fabrics of the enclosure support two switches, two CMC controllers, up to six power supplies, and nine fan units. If the module currently being used by the enclosure fails, the system will automatically switch its functions to the next working module. This ensures that the system will still be available even if a module fails. For example, if the power supply to your computer failed, your computer would just turn off. Using this system, the computer would switch to a second power supply and you wouldn't even know the first one failed. Within the Dell M1000e blade server system we have two different types of blade servers: the Dell PowerEdge M610 and the Dell PowerEdge M610x. [34]

According to Dell's website, the M610 blade server is "an Intel processor-based 2-socket, half-height blade server built for virtualization, mainstream business applications and front-end database workloads." It features up to 192GB of RAM, the 2.26 GHz Intel Xeon Processor, and even supports boot from SAN. The internal storage capacity for one M610 is 1.80 TB. [35]

Then we have the M610x blade server. According to Dell, the M610x is "an Intel processor-based 2-socket, full-height blade server ideal for organizations with unique I/O or computational needs requiring industry-standard PCIe slots." It boasts the same technical specifications as the M610, but also has room for PCIe cards (which are used by graphics cards, SAS analytics cards, and RAID controllers) and supports solid state drives. [36]

Below is a picture of the M1000e chassis. The left is a picture from the back of the chassis where all the modules are located. On the right is the front side server enclosure that's currently holding 16 half-height servers.

Dell M1000e Chassis Picture [37]

2.3 Testing Setup

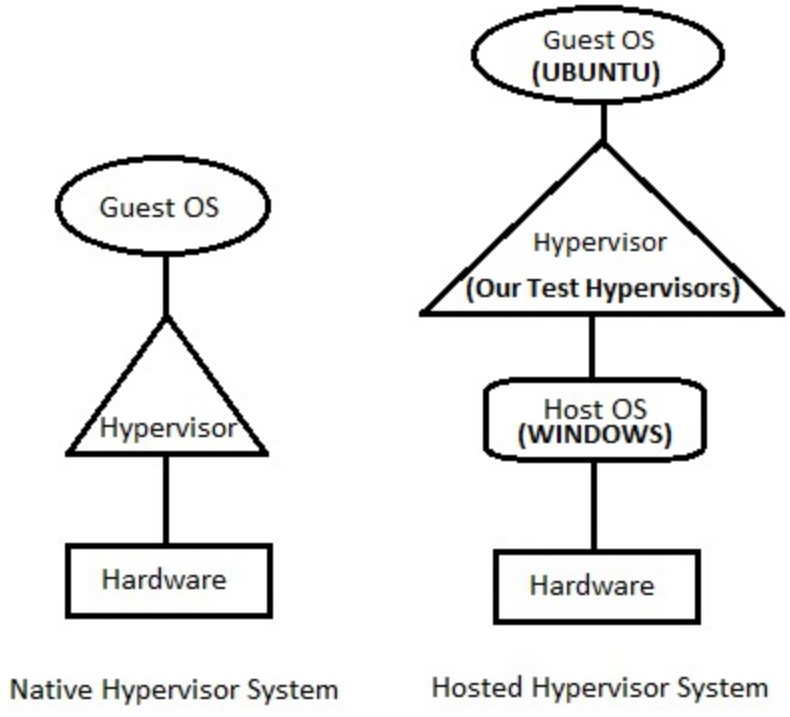
The physical machine setup we used to host and test the hypervisors is as follows:

- Microsoft Windows 7 Professional SP 1 Build 7601 / Ubuntu 13.10 as needed
- Intel Core i7-2630QM CPU @ 2.00 GHz, 2001 Mhz,4 Cores
- 12.0 GB RAM

The VM setup used when testing each hypervisor is as follows:

- Ubuntu 13.10
- 2048 MB RAM
- 1024 MB Disk
- CPU Settings changed to:
    - Allocate 1 virtual CPU
    - set maximum capacity as 100% and minimum capacity as 0% of Host CPU whenever possible or not defaulted.

There are two types of hypervisors: native and hosted. The difference between these two are that native hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems whereas a hosted hypervisor run within a conventional operating-system environment. This means the guest operating system on a native hypervisor is on the second level above the hardware, whereas on a hosted hypervisor the guest operating system is on the third level above the hardware. We are evaluating a series of hosted hypervisors as per the diagram below.



Personally Diagrammed Hypervisor Setup

2.3 Overview of Virtual Machines Researched

Below is a list of all the virtual machines researched, a small overview of the virtual machine, and a link to the virtual machine's current development website.

KVM - "KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream."
http://www.linux-kvm.org/page/Main_Page

LXC - "LXC provides operating system-level virtualization not via a virtual machine, but rather provides a virtual environment that has its own process and network space. LXC relies on the Linux kernel cgroups functionality that was released in version 2.6.24. It also relies on other kinds of namespace-isolation functionality, which were developed and integrated into the mainline Linux kernel."
http://linuxcontainers.org/

OpenVZ - "OpenVZ is container-based virtualization for Linux. OpenVZ creates multiple secure, isolated Linux containers (otherwise known as VEs or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict. Each container performs and executes exactly like a stand-alone server; a container can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files."
http://openvz.org/Main_Page

QEMU - "QEMU is a generic and open source machine emulator and virtualizer. When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance. When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. When using KVM, QEMU can virtualize x86, server and embedded PowerPC, and S390 guests."
http://wiki.qemu.org/Main_Page

VirtualBox - "VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2."
https://www.virtualbox.org/

Xen - “Xen is an open-source (GPL) type-1 or baremetal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host).”
http://www.xenproject.org/

3. Criterion 1 - Hypervisor Invisibility

3.1 Why Chosen and How It Was Tested

Hypervisor invisibility refers to the performance drop that VMs will see under a hypervisor when the two exhibit heavy work loads. As a reminder, hypervisor invisibility was chosen because we want to minimize performance drops that end users see due to multiple VMs being active under one hypervisor. To test this, we put Ubuntu 13.10 under each hypervisor, applied heavy CPU, memory, and disk loads, then measured efficiency by using a program called IOMeter. To establish a baseline of comparison, we ran IOMeter (www.iometer.org) on our test machine beforehand each time, then directly afterward started the Hypervisor / VM and ran the program on there to get a measure of efficiency. “Noticeable overhead” means that in some of the fields tested an efficiency rate less than 75% was shown. “Very minimal overhead” means that efficiency was over 75% for all cases.

3.2 Results

KVM - noticeable overhead. [13]

LXC - very minimal overhead. [10]

OpenVZ - very minimal overhead. [12]

QEMU - noticeable overhead. [19]

VirtualBox - noticeable overhead. [7]

Xen - noticeable overhead. [12]

4. Criterion 2 - Easily Cloneable via Command Line

4.1 Why Chosen and How It Was Tested

Easily cloneable via command line refers to there being a hypervisor command on the command line that allows someone to clone a guest OS (virtual machine). Most hypervisors use a GUI and don’t have much command line integration. We want command line capability because it allows us to perform maintenance on the server rack without having to physically go to the server room. There is no real way to test this other than finding out if the hypervisor has the capability or not, so by doing research and searching the menus I tried to determine if each hypervisor had the capability or not.

4.2 Results

KVM - takes some steps, but do-able with some modifications and scripting. [5] This includes establishing a workflow and established practice of VM cloning or different admins could manage to create very different clones / overwrite existing VMs. Another option is to install the VMs using the libvirt hypervisor management library and using the then built-in command "virt-clone". [24]

LXC - built in command "lxc-clone". [23]

OpenVZ - do-able by using built-in command "vzmlocal", but it's effects aren't consistent. [25] Consistent effects can be had by making scripts to make sure what you need to be done gets done manually. [27]

QEMU - possible by installing the VMs using the libvirt hypervisor management library and using the then built-in command "virt-clone". [24]

VirtualBox - built in command "clonevm". [3]

Xen - possible by installing the VMs using the libvirt hypervisor management library and using the then built-in command "virt-clone". [24]

5. Criterion 3 - List of VMs Accessible via Command Line

5.1 Why Chosen and How It Was Tested

List of virtual machines accessible via command line refers to there being a hypervisor command on the command line that allows someone to see a list of all guest OS's (virtual machines) currently residing on it in any state. Again, most hypervisors use a GUI and don't have much command line integration, but we want the capability because it allows remote server management. There is no real way to test this other than finding out if the hypervisor has the capability or not, so by researching and searching the menus we tried to determine if each hypervisor had the capability or not.

5.2 Results

KVM - full command line capability using "virsh list". [15] [22]

LXC - full command line capability using "lxc-list". [18]

OpenVZ - full command line capability using "vzlist". [28]

QEMU - no command line capability.

VirtualBox - full command line capability using "list". [3]

Xen - full command line capability using “xl vm-list”. [17]

6. Criterion 4 - Snapshot Capability

6.1 Why Chosen and How It Was Tested

Snapshot capability refers to there being some sort of capability to allow for preservation of the state and data of a virtual machine at a specific point in time. Keep in mind that a snapshot is not a backup, it is merely a changelog of the original VM. These are used so that if a risky change is made and messes up the system, the system can be reverted back to the state of the snapshot. Again there is no real way to test this other than finding out if the hypervisor has the capability or not, so by researching and searching the menus we tried to determine if each hypervisor had the capability or not.

6.2 Results

All virtual machines being researched are compatible with libvirt [21] [25], meaning if it’s installed they will all have snapshot capability. The results below show if the bare hypervisors without libvirt have the capability or not.

KVM - full command line capability. [2]

LXC - full command line capability. [1]

OpenVZ - full command line capability. [28]

QEMU - full command line capability. [2]

VirtualBox - full command line capability. [3]

Xen - full command line capability. [32]

7. Criterion 5 - Hot Swap Capability

7.1 Why Chosen and How It Was Tested

For regular computers and most VMs, there needs to be a full reboot for hardware to be recognized. Hot swap capability refers to the ability of changing the physical/virtualized hardware of a system while it’s still running. This is useful for when an end user is running out of disk space or using all the RAM specified while running a program and need more on the fly without interrupting their program. To test this, we tried to find out what hot swap capabilities each hypervisor had and then perform such a hot swap on my local hypervisors. More specifically we tried to hot swap disks, cpus, and memory.

7.2 Results

KVM - disks: yes [14], cpus: yes [22], memory: yes [11], general kvm hot swap info [22].

LXC - no.

OpenVZ - no.

QEMU - can hot swap drives

VirtualBox - can only hot swap cpus [4].

Xen - disks: yes [29], cpus: yes [31], memory: yes [30].

8. Results

8.1 Conclusion and Future Work

The CADRC is interested in supporting research projects and providing a revenue stream for the Cal Poly research division. But it has only has access to a few large capacity servers and that won't help them because most research projects on campus are small and do not require the entirety of a large capacity server to run. So, we examined hypervisors emulating virtual machines and focused on certain criteria in order to help them maximize utilization of the few large servers they have access to - in regards to hosting multitudes of small projects.

The criteria we looked at were: hypervisor invisibility, VMs are easily cloneable via command line, list of VMs is accessible via command line, snapshot capability, hot swap capability. The results of these examinations are nicely formatted in the table in section 8.3. With regards to these results, and looking at KVM, LXC, OpenVZ, QEMU, VirtualBox, and Xen, we believe that the CADRC would benefit the most using either KVM or Xen, based on the preference of the current system administrator. Both of the these meet most of the criteria listed, with the except of hypervisor invisibility where the overhead was somewhat noticeable, but shouldn’t drag the system down too much.

Looking forward there’s still more that can be done in regards to the area this White Paper examined. The evaluations are by no means a conclusive study of what exact VM the CADRC should be using. Further evaluations should include items such as:

- running the evaluations on the actual target platform rather than local machines
- running actual target applications with multiple VMs
- evaluation under some stress of high load testing conditions
- evaluation of network performance with multiple VMs working

8.2 Supported Features and Platforms

	HostCPU	GuestCPU	HostOS	GuestOS
KVM	x86, x86-64, IA-64, s390, PowerPC	same as host	FreeBSD, Linux, illumos	FreeBSD, Linux, Solaris, Windows, Plan 9
LXC	x86, x86-64, IA-64, PowerPC 64, SPARC64	same as host	Linux	Linux
OpenVZ	x86, x86-64, IA-64, PowerPC 64, SPARC64	same as host	Linux	Linux
QEMU	x86, x86-64, IA-64, PowerPC, SPARC 32/64, ARM, S/390, MIPS	x86, x86-64, Alpha, ARM, CRIS, LM32, M68k, MicroBlaze, MIPS, OpenRisc32, PowerPC, S/390, SH4, SPARC 32/64, Unicore32, Xtensa	Windows, Linux, Mac OS X, Solaris, FreeBSD, OpenBSD, BeOS	changes regularly [20]
VirtualBox	x86, x86-64, Intel VT-x, AMD-V	x86, (x86-64 only on VirtualBox 2 and later with hardware virtualization)	Windows, Linux, Mac OS X x86, Solaris, FreeBSD, eComStation	DOS, Linux, Mac OS X Server, FreeBSD, Haiku, OS/2, Solaris, Syllable, Windows.
Xen	x86, x86-64, ARM, IA-64 (inactive, PowerPC (inactive)	same as host	NetBSD, Linux, Solaris, MiniOS	FreeBSD, NetBSD, Linux, MiniOS, Solaris, Windows XP & 2003 Server, Plan 9

Table information taken from [6]

8.3 Results

	Hypervisor Invisibility	Easily Cloneable via Command Line	List of VMs Accessible via Command Line	Snapshot Capability	Hot Swap Capability
KVM	noticeable overhead. [13]	takes some steps, but do-able. [5] use built in command "virt-clone". [24]	full command line capability using “virsh list”. [15] [22]	full command line capability. [2] [21][25]	disks: yes [14], cpus: yes [22], memory: yes [11]
LXC	very minimal overhead. [10]	built in command "lxc-clone". [23]	full command line capability using "lxc-list". [18]	full command line capability [1] [21][25]	no
OpenVZ	very minimal overhead. [12]	inconsistent command “vzmlocal”. [26] Consistent effects by scripting. [27]	full command line capability. [28]	full command line capability. [28] [21][25]	no
QEMU	noticeable overhead. [19]	built in command "virt-clone". [24]	no command line capability.	full command line capability. [2] [21][25]	can hot swap drives
VirtualBox	noticeable overhead. [7]	built in command "clonevm". [3]	full command line capability using "list". [3]	full command line capability. [3] [21][25]	can only hot swap cpus [4]
Xen	noticeable overhead. [12]	use built in command "virt-clone". [24]	full command line capability using "xl vm-list". [32]	full command line capability. [32] [21][25]	disks: yes [29], cpus: yes [31], memory: yes [30].

9. Bibliography - Works Cited

[1] "Announcing lxc-snapshot | S3hh's Blog." S3hhs Blog. 09 Dec. 2013 <http://s3hh.wordpress.com/2013/09/13/announcing-lxc-snapshot/>.

[2] "Chapter 14. Administrating Virtual Machines with QEMU Monitor." Opensuse. 09 Dec. 2013 <http://doc.opensuse.org/products/draft/SLES/SLES-kvm_sd_draft/cha.qemu.monitor.html>.

[3] "Chapter 8. VBoxManage." VirtualBox.org. 09 Dec. 2013 <https://www.virtualbox.org/manual/ch08.html>.

[4] "Chapter 9. Advanced topics." VirtualBox.org. 09 Dec. 2013 <http://www.virtualbox.org/manual/ch09.html>.

[5] "Cloning VMs with KVM." Cloning VMs with KVM. 09 Dec. 2013 <http://www.greenhills.co.uk/2013/03/24/cloning-vms-with-kvm.html>.

[6] "Comparison of platform virtual machines." Wikipedia. 12 June 2013. Wikimedia Foundation. 09 Dec. 2013 <http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines>.

[7] Damodaran, Deepak K., Biju R. Mohan, Vasudevan M. S, and Dinesh Naik. Performance Evaluation of VMware and VirtualBox. , National Institute of Technology Karnataka, India. 2012. PDF.

[8] De la Rosa, Jose. KVM Virtualization in RHEL 6 Made Easy. Dell Linux Engineering. 2011. PDF.

[9] "Features/Snapshots." QEMU RSS. 09 Dec. 2013 <http://wiki.qemu.org/Features/Snapshots>.

[10] "Heiko's Admin Blog." Heikos Admin Blog RSS. 09 Dec. 2013 <http://honeybutcher.de/2012/09/openstack-lxc/>.

[11] "Increase kvm guest memory runtime." Ubuntu Forums RSS. 09 Dec. 2013 <http://ubuntuforums.org/showthread.php?t=1933391>.

[12] "Knowledgebase." OpenVZ or Xen VPS. 09 Dec. 2013 <https://my.vps6.net/knowledgebase/69/OpenVZ-or-Xen-VPS---Which-is-faster-and-which-is-better.html>.

[13] KVM - Kernel Based Virtual Machine. Red Hat. 2009. PDF. http://www.redhat.com/rhecm/rest-rhecm/jcr/repository/collaboration/jcr:system/jcr:versionStorage/5e7884ed7f00000102c317385572f1b1/1/jcr:frozenNode/rh:pdfFile.pdf

[14] "KVM and Libvirt - How do I hotplug a new virtio disk?" Serverfault.com. 09 Dec. 2013 <http://serverfault.com/questions/457250/kvm-and-libvirt-how-do-i-hotplug-a-new-virtio-disk>

[15] "KVM get hypervisor and guest virtual machine details." The Linux Juggernaut. 09 Dec. 2013 <http://www.linuxnix.com/2013/02/kvm-get-hypervisor-and-guest-virtual-machine-details.html>

[16] "Kvm hot-add CPUs." Mayfirst support. 09 Dec. 2013 <https://support.mayfirst.org/ticket/3899>.

[17] "Managing VMs." GitHub. 09 Dec. 2013 <https://github.com/Xenapi-Admin-Project/xenapi-admin-docs/blob/master/Guides/Beginners-Guide/Managing_VMs.txt>.

[18] "Man/vzlist." OpenVZ Linux Containers Wiki. 09 Dec. 2013 <http://openvz.org/Man/vzlist.8>.

[19] "Nested Virtualization Shootout: Ravello, VMware, QEmu." Ravello. 09 Dec. 2013 <http://www.ravellosystems.com/blog/nested-virtualization-shootout-ravello-vmware-qemu/>.

[20] "QEMU Official OS Support List Version 2.0." QEMU Official OS Support List Version 2.0. 09 Dec. 2013 <http://www.claunia.com/qemu/old/index.php>.

[21] "Running libvirt with KVM - KVM." Running libvirt with KVM - KVM. 09 Dec. 2013 <http://www.linux-kvm.org/page/Running_libvirt_with_KVM>.

[22] "Ubuntu Documentation." KVM/Managing. 09 Dec. 2013 <https://help.ubuntu.com/community/KVM/Managing>.
[23] "Ubuntu Documentation." LXC. 09 Dec. 2013 <https://help.ubuntu.com/lts/serverguide/lxc.html>.
[24] "Virt-clone(1)." Linux man page. 09 Dec. 2013 <http://linux.die.net/man/1/virt-clone>.
[25] "The virtualization API." Libvirt:. 09 Dec. 2013 <http://libvirt.org/formatsnapshot.html>.
[26] "VZ clone script to clone vz containers." C S Shyam Sundars Thoughtpad. 09 Dec. 2013 <http://thoughtpad.cs3.in/2010/05/25/vz-clone-script-to-clone-vz-containers/>.
[27] "Vz windows reference." Vzmlocal. 09 Dec. 2013 <http://download.swsoft.com/pvc/46/win/docs/en/VzWindowsReference/4556.htm>.
[28] "Vzctl manual." - Proxmox VE. 09 Dec. 2013 <http://pve.proxmox.com/wiki/Vzctl_manual>.
[29] "Xen Disk Hot Add." Backdrift RSS. 09 Dec. 2013 <http://backdrift.org/xen-disk-hot-add-block-device-howto>.
[30] "Xen Hot Add/Remove Memory." Backdrift RSS. 09 Dec. 2013 <http://backdrift.org/xen-memory-hot-add-and-remove>.
[31] "Xen Hot Add/Remove VCPUs." Backdrift RSS. 09 Dec. 2013 <http://backdrift.org/how-to-hot-addremove-vcpus-from-a-xen-domain>.
[32] "XL man page." Xenbits.xen.org. 09 Dec. 2013 <http://xenbits.xen.org/docs/unstable/man/xl.1.html>.
[33] "Z/Architecture." Wikipedia. 12 Aug. 2013. Wikimedia Foundation. 09 Dec. 2013 <http://en.wikipedia.org/wiki/Z/Architecture>.
[34] "Dell M1000e Server Enclosure." Wikipedia. 21 Nov. 2013. Wikimedia Foundation. 13 Dec. 2013 <http://en.wikipedia.org/wiki/Dell_M1000e>.
[35] "PowerEdge M610 Blade Server." Dell. 13 Dec. 2013 <http://www.dell.com/us/business/p/poweredge-m610/pd>.
[36] "PowerEdge M610x Blade Server." Dell. 13 Dec. 2013 <http://www.dell.com/us/business/p/poweredge-m610x/pd>.
[37] Dell M1000e Chassis. Digital image. Dell. Dell. 13 Dec. 2013 <http://media.community.dell.com/en/dtc/txmna3r6dlhieyzcutcj5g40120.jpeg>.