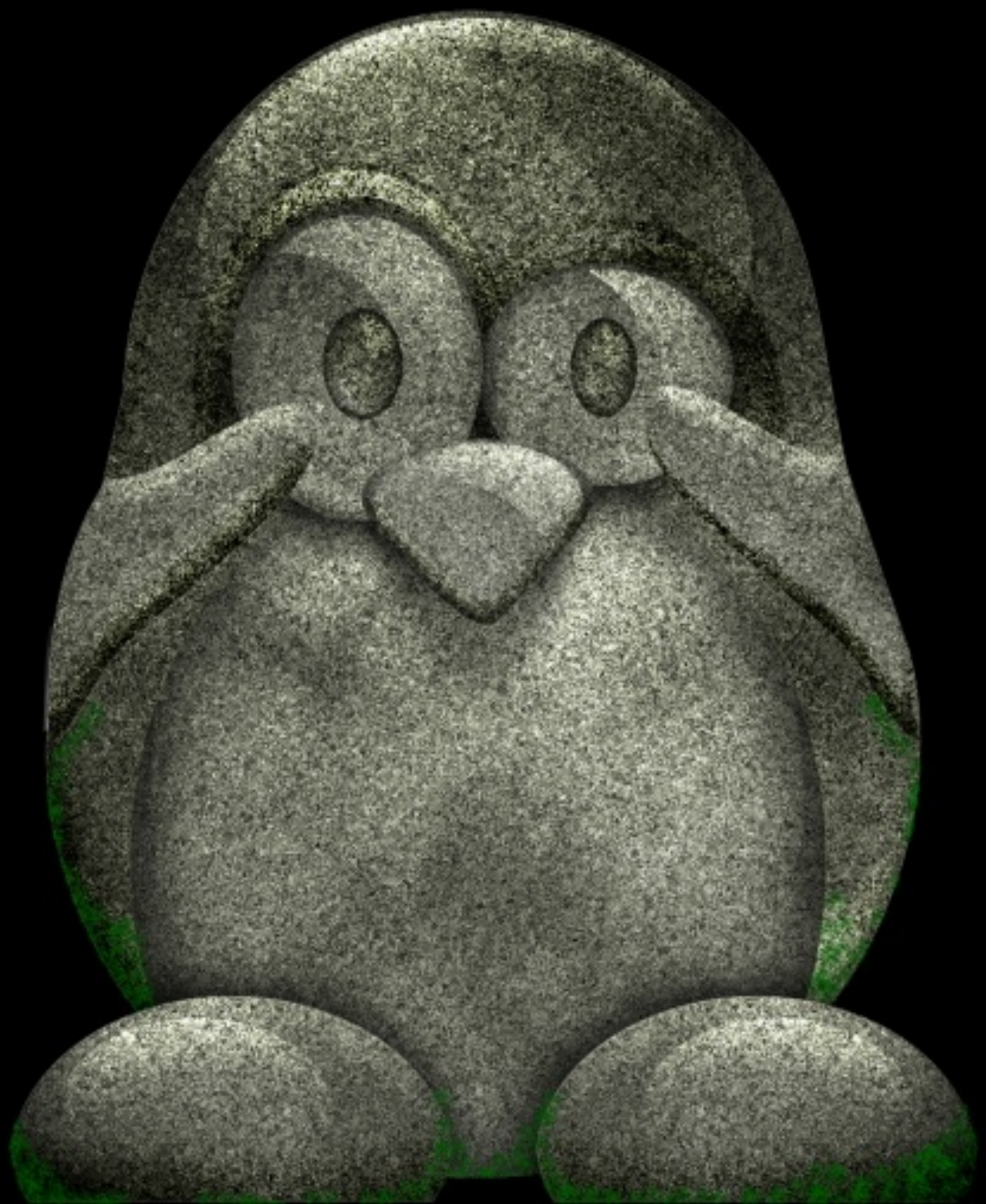
opensourc3

The Premier Unified Computing magazine for IT Professionals

Building IP Networks *Systems Automation with Puppet* *Hypervisor Technologies* *Inside KVM*

Issue 1
August 2009

"xfce"



H@cker | Halted™

USA 2009

Miami | Florida

Academy | September 20 - 22, 2009
Conference | September 23 - 25, 2009

FREE TRAINING
Worth **$599!***

Special Rate of JUST $799
for all OPENSOURC3 Subscribers!*
Promo code: HHUSA-MP-OSC3799

Intriguing . Provocative . Informative

Get certified and obtain new technical skills.
Understand the state of information security.
Stay updated on latest threats and countermeasures.
Network with infosec professionals from around the world.
Be part of the world's largest reunion of Certified Ethical Hackers.

Bonus !

Register for the Conference, and attend one of three special one-day full fledged training workshops (Sep 25) led by EC-Council Master Instructors.

Identifying Threats & Deploying Countermeasures | Incident Response: Principles of Incident Handling | Virtualization: Threats Exposed.

Hackers Are Ready. Are you?

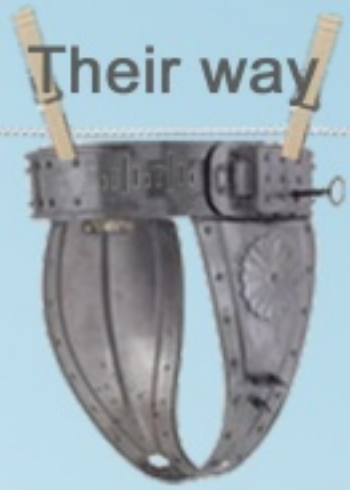
Register Now !

www.hackerhalted.com

Networking Freedom

Our way

Their way



At Vyatta we have nothing to hide. Our users can always see our code and our network OS is not hidden behind some proprietary sheet metal. We think routing & security software should be given room to move. That's why Vyatta has designed the only networking solution that is just as comfortable in virtualization and cloud infrastructures as it is on hardware. Don't get locked in, get Vyatta.

VYATTA®
Open Networking

Get your free copy of Vyatta open networking software - http://www.vyatta.com

About opensourc3 | INFORMATION

EXECUTIVE EDITOR
JOHN BUSWELL (BUSWELLJ@OPENSOURC3.ORG)

SENIOR COLUMNIST
JUAN LEANIZ (JUAN.LEANIZ@CARBONMOUNTAIN.COM)

SALES MANAGER
CRAIG LUNT (CRAIGLUNT@OPENSOURC3.ORG)

OPENSOURC3 IS PUBLISHED BY, AND IS A TRADEMARK OF, **CARBON MOUNTAIN LLC**

ADVERTISING
PLEASE CONTACT SALES@OPENSOURC3.ORG

SUBSCRIPTIONS
SUBSCRIBERS RECEIVE EARLY ACCESS TO THE PUBLICATION, INVITATIONS TO SPECIAL EVENTS, DISCOUNTS AND OTHER PROMOTIONAL CONSIDERATION. SUBSCRIPTION IS FREE. TO SUBSCRIBE VISIT:

HTTP://SUBSCRIBE.OPENSOURC3.ORG

TUX IMAGES ARE PROVIDED COURTESY OF THE TUX FACTORY.
HTTP://TUX.CRYSTALXP.NET

THIS PUBLICATION WAS CREATED WITH THE FOLLOWING OPEN SOURCE TECHNOLOGIES.

HTTP://WWW.SCRIBUS.NET
HTTP://WWW.GIMP.ORG

OTHELLO

Colocation | Dedicated Servers | VPS | Hosting | Internet Access | IP Transit | Domain Services

European customers are only milliseconds away..
Place content and services near your customers

Othello is your Global Internet Partner

Find out more at http://www.othellotech.net
or call our Business Sales Team on

UK +44 871 277 6875
US +1 845 228 8551

Othello
TECHNOLOGY SYSTEMS LTD

Othello Technology Systems Ltd - Registered in England (UK), Company #03894981

Welcome....

Today the data center is evolving, there are numerous emerging technologies, endless blogs, editorials and other commentary on some form of [INSERT YOUR FAVORITE] computing technology. While the terminology is often alien, the underlaying technology is very familiar to every IT professional. Those familiar technologies are the building blocks for the next generation data center. Whether its Unified Computing, Cloud Computing, Utility Computing or Grid Computing that is the flavor of the month, those building blocks are unlikely to fundamentally change. Each of these computing models needs storage, networking, compute and virtualization technologies. How those technologies are integrated together, deployed and managed may change, but the core technologies themselves are still needed.

This publication focuses on those core technologies, the open source projects that provide and augment them, and the unique ways to leverage those technologies in your production environment. The public cloud services that are available today are built on top of open source technologies, innovative projects such as Eucalyptus and KVM are enabling the private cloud, and other next-generation data center solutions. Each month this publication will take an in-depth look at Virtualization, Storage, Compute / Applications, and Networking technologies in dedicated columns. In addition to these columns, a monthly feature article will explore and explain a related emerging technology.

This publication is targeted at Information Technology professionals. If you are currently (or plan to be) involved in Business / Enterprise grade Information Technology solutions, then this publication is for you. Whether you are the technical lead at a startup, system administrator at an established business or the CIO responsible for a multi-million dollar budget, this publication is designed to save you time by providing focused and factual technical information on open source solutions.

In this inaugural issue, the Virtualization column takes a look at KVM, the Linux Hypervisor that for many is the emerging de-facto virtualization platform. This months storage article introduces iSCSI and examines the necessary steps to get it running in Linux. The compute column examines Puppet, an open source project designed to facilitate configuration management on large numbers of servers, whether physical or virtual. Following several interesting conversations with a number of technical trainers, it is clear that a large number of IT professionals responsible for small and medium sized business networks are unaware of fundamental IP data network concepts. The networking column looks at several best practices when deploying basic IP data networks to lay the foundation for tomorrows larger corporate networks.

The feature article this month takes a high level look at hypervisor and cloud technologies. What exactly is a cloud, do you need a private or public cloud, what are the benefits and how do these technologies tie in with technologies such as Unified Computing, are just some of the questions we try to answer.

Curious as to why "xfce" is on the cover? Each month the cover will reflect a code word, not only to highlight a specific project but also as a competition. The first three subscribers to email buswellj@opensourc3.org with the code word in the subject line will win an exclusive opensourc3 t-shirt. Please include your name, mailing address and desired t-shirt size in the body of the email. With or without a free t-shirt we encourage you to check out this months project at http://www.xfce.org. Xfce is a fast and lightweight desktop environment for a range of *NIX systems.

September 25-26 2009

Hyatt Regency, Columbus

Greater Columbus Convention Center

Diversity in Open Source Workshop
Sunday Sept. 27
First Year Ever!

featuring

M. Douglas McIlroy
Shawn Powers
Dr. Peter Salus
Bdale Garbee
Elizabeth Garbee
Jono Bacon
...and many more!

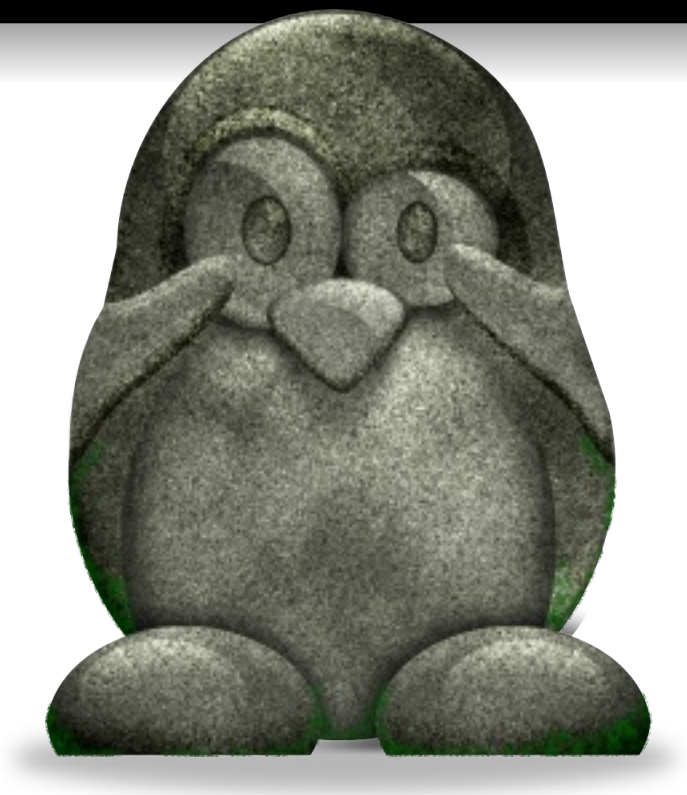
OLFU Friday 25th @ Hyatt Regency
Ohio Linux Fest University
Learn Linux from the Pros!

Ohio Linux Fest

40 Years of Unix

>> Bigger & Better Than EVER!

register FREE today! ohiolinux.org



Hypervisor and Cloud Technologies

Examining the emerging technologies behind the mysterious hype of next-generation data center technologies..

by John Buswell

Change is coming to the data center in the form of Virtualization, Unified Computing and Cloud Computing technologies. One of the biggest problems with emerging technologies though is that they are a "grey area", not clearly defined and often ending up with dozens of interpretations from different parties interested in shaping these new technologies to their benefit. Even clearly defined solutions such as the Hypervisor are still in a state of flux. As with all technologies, most users will end up with a hybrid solution, taking the best components and adapting them to individual business needs. Open source is driving innovation in this area, and is best suited to provide the components for building the next generation data center. This article looks at the Hypervisor, the core technology required for Virtualization. It then goes on to look at Cloud and Unified computing, both emerging technologies that enable these solutions as well as some of the challenges that still need to be solved.

What is Virtualization?

Before looking at the Hypervisor, it is important to understand exactly what Virtualization is and why it is important. Virtualization is the emulation of computer hardware to create a Virtual Computer (or Virtual Machine) that you can run an operating system on. To the user, a Virtual Computer looks no different than a physical one. The two primary methods of Virtualization are full software emulation and hardware assisted virtualization. The QEMU project is a good

example of full software emulation. That project can emulate different microprocessors in software, and provide a set of virtual device models that essentially trick the operating system into thinking that it is running natively on hardware. One of the key problems with emulators is that there can be a steep performance hit, depending on the application.

Hardware assisted virtualization is where the microprocessor itself has the capability to support building a hypervisor and providing isolation between Virtual Machines running on that system. The key benefit this type of technology has is performance, as the software itself has less to do. One thing to note is that hardware assisted virtualization is not emulation, so trying to run a PowerPC operating system on x86 hardware would not usually be possible with hardware assisted virtualization.

Why Virtualization?

So at first glance this technology seems to add a lot of complexity to the normal server environment, why do it? The key benefit to Virtualization is server consolidation through more efficient use of the system resources. Many IT deployments have under utilized servers, machines that have a specific important task but are often idle. Take for example four servers that need to run on separate operating systems but average a 10% hardware utilization rate, with bursts never exceeding 20%. At a minimum, these four servers can be consolidated onto a single server. Now that server will average a 40% hardware utilization rate, with bursts up to 80%. To the user, nothing has changed, but in the data center, three servers have been freed up. With data center real estate at a premium, and out of control power / hvac requirements, the ability to leverage hardware in this manner is critical. This is a very simplistic look at consolidation, a proper audit and leveraging smaller numbers of more powerful servers can reduce racks of servers down to a single rack. Virtualization is all about maximizing the efficient use of resources, and the Hypervisor is the technology that enables it.

What is a Hypervisor?

The Hypervisor is a software / hardware solution that enables Virtualization. It is sometimes referred to as a Virtual Machine Monitor or VMM. The Hypervisor can be categorized into one of two types. A Type 1 Hypervisor or bare-metal Hypervisor, is one that runs directly on the hardware without needing any specific host operating system. A Type 2 Hypervisor does not interface directly with the hardware, instead it needs a host operating system. A good example of a Type 2 Hypervisor is VMware Fusion and Parallels for the MacOS X platform. These Hypervisor solutions sit on top of MacOS X and make it possible to run Linux, BSD and Windows operating systems on Virtual Machines. A production environment, especially one that is support a higher level Unified Computing or Cloud Computing solution, will need to run a Type 1 Hypervisor technology.

Client Hypervisors and Virtual Desktop Infrastructure (VDI)

Having realized the benefits of virtualization in server environments, VDI and the Client Hypervisor are technologies that look to provide similar benefits to the Desktop. VDI is a hosted solution where the desktop runs on a server and is accessed via a remote desktop solution such as RDP or VNC. The advantage of VDI is that the user has the resources available to them when they need to run a resource intensive application. Often upgrading of resource intensive applications is delayed as it requires the replacement of

desktop computers, as newer versions are more resource intensive. In a VDI environment, this is not necessary, enabling a business to extend PC refresh cycles by several years. The problem with a VDI environment is that it is hosted, this is where the Client Hypervisor technology steps in to enable the user to bring their hosted VDI environment with them, even when they are off-line. While there is little difference between a Client Hypervisor and a Type 1 Hypervisor, it brings new management challenges to the table.

Open Source Hypervisors

There are two core Open Source Hypervisor technologies available today. These are Xen and Kernel-based Virtual Machine (KVM). Both solutions are hardware assisted virtualization technologies, requiring Intel or AMD virtualization extensions to exist on the processor. To determine if the processor is capable of supporting these solutions look at the output of /proc/cpuinfo. On AMD platforms the extension is called svm, and vmx on Intel platforms. There is an open source project called kqemu, part of the Qemu project that provides x86 and x86-64 accelerated virtualization support for platforms that do not support these Intel / AMD hardware virtualization extensions. This makes it possible to create Type 1 Hypervisors from hardware that would not normally support such capabilities. KVM is rapidly taking over from Xen, primarily due to its flexibility and the fact that KVM treats virtual guests as processes which can then be managed using traditional mechanisms within Linux. Xen is far from dead though, Xen has a much larger community and many production deployments, it also has a much richer suite of support applications, although many are being ported to work with KVM. Many companies, including Red Hat have indicated that KVM is the future of Virtualization in Linux.

What is Cloud Computing?

Cloud computing is an evolving technology, which is why it is often difficult to pinpoint a specific definition of the technology. Cloud computing today is the leasing of infrastructure, typically in an automated manner using a web based interface. Cloud computing eliminates the need for IT staff to manage hardware directly, instead they can select what they need, choose the desired system image and have instant access to the resources. Cloud computing is a metered technology, which is why its often referred to as being similar to Utility Computing. Its metered in that the user pays for the time and amount of resources they use over that time. It is particularly useful for bursting, where the user may need a large amount of resources for a relatively short period of time. The costs are significantly less than the cost to maintain the physical hardware in-house.

Instances that run in the cloud, are provisioned virtual machines. Where in the cloud they are running is not important, the underlaying architecture of the cloud ensures that system and network security is maintained. The instance in the cloud can be any guest operating system, solutions such as Amazon EC2 offer a wide range of images with pre-configured solutions such as Apache and MySQL already installed, making it quick and easy to deploy services. Each instance is governed by a Service Level Agreement or SLA. This sets the parameters of what resources are available to you in the cloud. As you need more resources, you provision more instances or upgrade the SLA on an existing instance.

What the Cloud offers...

The current cloud computing solutions deliver

value to the user. The user can quickly deploy applications and services, eliminate the need to procure and manage hardware, pay for only what they consume and the ability to scale up and down on demand. The cloud offers instant delivery, the user can provide applications anywhere at anytime. The user no longer has to wait for hardware requests to be approved, hardware to be acquired and systems to be setup. Important IT projects can be developed and tested in the cloud and then brought in-house or run in production on a public cloud. It enables solutions to be tested and developed without having to make any up front investment in hardware. Ultimately the cloud provides greater CAPEX and OPEX efficiency by utilizing a shared infrastructure throughout the business. In busy IT environments, there is a tendency to "fire and forget" servers. Once a physical server is rolled out, its often forgotten about until it becomes a problem. The cloud offers a greater insight into the actual utilization of resources and provides an immediate audit trail for the infrastructure. The very image based nature of the cloud makes it possible to redeploy resources instantly rather than having to go through the process of decommissioning and redeploying physical server operating systems. The cloud results in an agile IT department.

Challenges facing the Cloud...

There are a number of challenges facing the cloud. A key factor is trust. In an Enterprise IT environment, the system administrators are in full control of the underlaying network and server architecture. For the public cloud providers, it is critical to build trust and for the private cloud, the IT department needs to become comfortable with the new technology. The challenge of trust is something that will fade over time, as IT departments start to dabble with the cloud, lower tier services which are less important will be migrated off to the cloud, and as the advantages become apparent, IT managers will embrace the cloud more aggressively, similar to how many IT managers are taking more aggressive virtualization strategies today.

Something that goes hand in hand with trust, is the current weak SLA offerings in the cloud. In order for critical services to be placed in the cloud, IT managers will need more assurances and accountability from the cloud providers. As the technology matures, the cloud providers will be able to offer stronger SLAs which will encourage faster cloud adoption.

Probably the two most important and most difficult challenges to overcome are less technical and more political. The cloud challenges the status quo of corporate governance policies, in many cases the entire concept of infrastructure as a service (IaaS) flies in the face of well established corporate governance policies. Those policies will ultimately need to evolve with the technology, but they are often much more difficult and much slower to adapt. The challenge of enabling secure data mobility between the existing enterprise infrastructure and the leased IaaS in the cloud will need to be addressed in an acceptable manner, in order for those policies to change. Both aggressive SLAs from public cloud providers and new technologies that bridge the Enterprise infrastructure with the Cloud in a secure manner, should enable the policies to be maintained in the cloud that will be critical for serious adoption of Cloud computing.

The Cloud is Open Source...

All of the major early cloud providers today are built on Open Source technologies. Whether its Amazon's EC2 or Google's

AppEngine, these major contenders in the cloud space are powered by open source. The high availability requirements of mission critical IT deployments will require that services that run on one cloud can easily be transitioned to run on another. This is done most effectively through Open APIs and open source technologies. This is already apparent today, with projects such as Eucalyptus offering compatibility with Amazon AWS, both Xen and KVM hypervisor technologies and the ability to develop on a private cloud before deployment. As more players enter the market, they will offer their own extensions and innovations, while needing to maintain compatibility with the major players to enable easy migration from or integration with other cloud providers. Initially this maybe managed through third party devices bridging between the Enterprise and public cloud services. Over time technology and business will force a degree of consolidation as the technology matures. Open source platforms are well known for maximizing the resources available to the system and as such they are perfect for cost savings benefits in the cloud. While there are some optimizations that will need to be done, as the technology evolves, users should expect to see optimizations to both operating systems and applications that reduce waste, similar to the way electricity is saved through Energy Star appliances and energy efficient lighting. Open source is well suited to seeing innovations in this area, as end-users and developers look for ways to optimize and reduce waste to save money in the cloud.

Unified Computing - bridge to the clouds

Unified Computing is the convergence of Virtualization, Networking, Compute and Storage technologies into a single seamless system. Instead of these components being managed separately, in a Unified Computing solution, they are managed as a single system. By integrating these technologies into a single system, its possible to simplify the provisioning and management of IT resources, while maintaining tight control and enforcement of policies end to end. Having these components tied together enables a wide range of innovative new features to further leverage virtualization by auto-provisioning of services both locally and in the cloud. Unified Computing, while a new and emerging technology is an important bridge between the Enterprise and the public Cloud. Unified Computing brings the Enterprise environment closer to the cloud environment without the cost or disruptive change of switching to a private cloud environment. With a Unified Computing solution running in the data center, it is a lot easier for the IT department to leverage the public cloud, services and images transition easily between the two. Unified Computing platforms that integrate the public cloud APIs can auto-provision instances but at the same time auto-provision local services such as VPN and required policies to securely enforce corporate governance policies in the cloud. The solution also eases the mobility of data between the Enterprise and the public cloud, while maintaining policies on that data. For example, VLAN, Network Access Control and File System Encryption policies can be extended to the instances running the cloud by auto-provisioning the necessary services on both ends. Going forward, Unified Computing is a critical bridge between traditional data centers and the public cloud.

Ultimately a Hybrid Strategy

The cloud will certainly not replace the data center anytime soon. Ultimately, there will be a hybrid solution where the data center of tomorrow is an efficient and smaller version of data centers today. Utilizing virtualization technology to effectively consolidate servers

and leveraging public clouds to handle on-demand infrastructure needs as a service. Unified computing will provide the interface between the efficient local data center and numerous public cloud services, enforcing corporate policies and maintaining the integrity of data as it moves between the cloud and the on-premise network.

Open Source Technologies to watch

There are a number of key technologies to keep track of in this space. The KVM hypervisor project, libvirt, Cobbler, thincrust, Condor, Eucalyptus, oVirt and our KaOS project. The libvirt project is an API and layer to interact with the KVM hypervisor, it is written in C with wrappers for many languages, including Python. The cobbler and thincrust projects look to make creation and provisioning of virtual servers easier, while oVirt provides a web based management interface for virtual machines. The Eucalyptus project provides an open source Cloud computing solution which is compatible with Amazon. The condor project provides a grid scheduler and opens the possibilities for solutions such as cloud aggregation. The KaOS project is an open source project that Carbon Mountain will be launching in August that provides a universal lightweight Hypervisor / Virtual Server platform based on KVM, designed to make it easier to create and deploy virtual servers, and solves many of the issues related to trying to shoe-horn traditional operating systems into the virtualized environment.

Inside Kernel-based Virtual Machine

KVM is a kernel module that transforms Linux into a Hypervisor.

by John Buswell

Kernel-based Virtual Machine, or KVM for short is a relatively new technology embedded into the Linux kernel that converts Linux into a Hypervisor. KVM is loaded as a module either using kvm-intel or kvm-amd, depending on the hardware platform you are running. KVM requires hardware virtualization extensions to exist on the processor. These are tagged as either svm (amd) or vmx (intel) in /proc/cpuinfo. KVM is extremely easy to setup, aside from a couple of Kernel options, it needs a kvm-enabled version of Qemu and within a few minutes you will have a Linux Hypervisor.

What can KVM do?

KVM offers a competitive feature set including the ability to overcommit CPU and memory, high performance i/o, hot-plugging capabilities on cpu, block, NIC and PCI based devices, multi-processor guests, live migration, along with advanced features such as NUMA support, Power Management and Page sharing. Combined with the already comprehensive feature-set in the Linux kernel itself, from scheduler to hardware support, KVM rapidly starts to look like a very flexible and powerful Hypervisor solution.

Quick Start

There are a number of prerequisites that need to be installed on your system. In Debian, these can be installed with the following commands as root:

apt-get install gcc
apt-get install libsdl1.2-dev
apt-get install zlib1g-dev
apt-get install libasound2-dev
apt-get install linux-kernel-headers
apt-get install pkg-config
apt-get install libgnutls-dev
apt-get install libpci-dev

Taking the latest kvm-release.tar.gz, you untar it and run *./configure --prefix=/usr/local/kvm*. If you already have a patched kernel, you can add *--with-patched-kernel* to skip building the modules. Run *make && make install*. Then simply *modprobe kvm-intel* or m*odprobe kvm-amd*.

Now create a disk image for the virtual machine using qemu-img. The command */usr/local/kvm/bin/qemu-img create -f qcow test.img 4G*. This creates a test.img file that is 4G is size, suitable for our Fedora Core 11 test installation. To run the installation process, we have downloaded Fedora-11-x86_64.iso and stored it in /tmp. So next we run */usr/local/kvm/bin/qemu-system-x86_64 -hda test.img -cdrom /tmp/Fedora-11-x86_64.iso -boot d -m 384*. Here we have told kvm to use test.img, the iso image for the virtual cdrom, and to use 384MB of memory for the guest. KVM does not make any distinction between x86 and x86-64, so its safe to run these commands on 32-bit x86 systems. From this point you will need to access the virtual guest console. This can be done via virtual serial console, and a few other means, but the easiest way for testing is to use virt-manager. The virt-manager application is available in both Fedora and Ubuntu, and it provides a GUI console into the

virtual guest. Simply select the running guest from the list in virt-manager and it will connect. From here, you can do the usual Fedora installation to your virtual drive.

Once the installation is complete, simply run */usr/local/kvm/bin/qemu-system-x86_64 test.img -m 384* to start the virtual guest. Here you can SSH into the virtual guest or continue to use virt-manager for console access.

Inside KVM

Linux already has a best of breed scheduler, memory management, i/o and network stacks. Rather than re-inventing the wheel, KVM takes the sensible approach of building on top of these already solid and time-test components within the Linux kernel. KVM is tightly integrated into Linux, as such it can utilize a lot of the existing features and capabilities. There are three key features that are worth mentioning - virtio, DMA binding and Virtual CPUs.

Virtio - Faster I/O

One of the problems KVM had was that its network i/o performance was a significant bottleneck. To be perfectly honest, compared to other platforms, KVMs performance was poor. To solve this particular problem, a set of drivers and capabilities was added to the Linux kernel. These are collectively known as virtio. These new features require a kernel equal to or later than 2.6.25 and you must activate the *CONFIG_VIRTIO_** options in the kernel configuration. These provide PCI, Balloon, Block, and Network device support. The nice thing about Virtio is on the Virtual Guest kernel you can disable all other NIC and SATA/SCSI drivers as they are no longer necessary. To utilize virtio, you simple add *if=virtio* to the qemu command line for disk, and *-net nic,model=virtio* for networking. For disk resources you must use the *-drive* option.

DMA binding

One of the neat things that KVM supports is the ability to unbind hardware devices attached to the Hypervisor and provide direct access to the Virtual guest. This enables a virtual guest that maybe acting as a firewall between two networks to directly access the NICs being used, removing the extra I/O bottleneck that you would have passing the traffic to and from the Hypervisor. This is a key advantage over Hypervisor platforms. The PCI device remapping options in the kernel must be enabled, along with the PCI Stub driver for this to work. The device and vendor-id are passed into the sys interface then -pcidevice along with the device-id are passed into the startup command for the KVM guest. An upcoming article in opensourc3 will investigate this capability in-depth.

Virtual CPUs

One of the most interesting aspects of KVM is how it implements virtual CPUs. The solution used by KVM is to add a guest thread to the existing user and kernel thread types. The virtual cpu is then treated like a native linux thread, enabling the management of that virtual cpu to be handled by the scheduler just like other Linux threads. This clever design not only leverages the powerful Linux scheduler but opens up a wide range of strategies related to resource allocation on a KVM based Hypervisor.

Conclusion

KVM converts Linux into a Hypervisor by adding a virtualization layer. KVM requires VT extensions to exist on the processor itself. A special copy of qemu is used to manage KVM virtual guests, and new capabilities as part of Virtio provides KVM with competitive performance. KVM is tightly integrated into the kernel, providing a powerful and flexible Hypervisor platform.

open source agile product management

http://www.yoxel.com

Deploying iSCSI on Linux

iSCSI is a protocol for encapsulating SCSI commands over IP Networks, enabling users to build SANs.

by John Buswell

The iSCSI protocol encapsulates SCSI commands over an IP network and is an essential building block for creating IP based Storage Area Networks or SANs. In an iSCSI solution, the clients are called initiators and the server is called a target. By sending SCSI commands over the IP network, it enables data transfers over local networks and management of storage solutions over large distances. iSCSI is a popular technology as it does not require any specialized hardware. The iSCSI protocol works over traditional IP networks. iSCSI runs over TCP and utilizes ports 860 and 3260.

Software vs Hardware solution

Similar to the differences between Software RAID and Hardware RAID, iSCSI initiators can be done in both software or in hardware. With modern hardware today, software based iSCSI initiators are sufficient for most applications. Hardware initiators typically come in the form of TCP offload and reducing the overhead of iSCSI from the CPU. Some Intel based NICs provide special off-load capabilities that can help improve performance. Most modern NICs that support acceleration in Linux would likely offer similar improvement to the iSCSI protocol.

Simple Network

For the purpose of this article we will utilize a simple network, the iSCSI client will be on 10.19.12.15 and the iSCSI server will be on 10.19.12.25.

iSCSI Enterprise Target

This project is a fork of the Ardis target implementation, adding SMP, Linux 2.6, iSNS, Dynamic configuration and 64-bit architecture support. You can download it from http://iscsitarget.sourceforge.net or by running apt-get install iscsitarget. Some configuration is required as well as setting up the logical storage. Logical storage can be a variety of native storage solutions from LVM logical volumes, image files, physical drives such as /dev/sda, partitions such as /dev/sdb2 or software raid devices such as /dev/md0.

Configuring the iSCSI Target

The iSCSI target requires three configuration files to be modified. These are */etc/default/iscsitarget, /etc/ietd.conf and /etc/initiators.allow*. If you are familiar with NFS, initiators.allow is a bit like /etc/exports, defining a target device and what can access it. The ietd.conf file provides basic target configuration and iscsitarget does some system level configuration. *echo "ISCSITARGETENABLE=true" > /etc/default/iscsitarget*. Then create ietd.conf as follows:

```
Target iqn.2009-08.org.opensourc3:test.lun1
        IncomingUser theuser thepassword
        OutgoingUser
        Lun 0 Path=/dev/vg0/test_lun1,Type=fileio
        Alias TEST
        MaxConnections  2
```

The target is an iSCSI Qualified Name. The format is iqn.YYYY-MM.<reverse-

domain>:[identifier]. Where YYYY-MM is the date from which the iSCSI target is valid, <reverse-domain> is the reverse domain name, for example opensourc3.org becomes org.opensourc3. The identifier can be whatever you want. The LUN option specifies the path to the actual storage medium being used.

Finally we configure initiators.allow, *echo "iqn.2009-08.org.opensourc3:test.lun1 10.19.12.15" > /etc/initiators.allow*. If it doesn't already exist we use *lvcreate -L5G -n test_lun1 vg0* to create the LVM device. This will create test_lun1 on a Volume group called vg0. Start up the target with /etc/init.d/iscsitarget start. Remember to inspect the iptables rules to make sure that tcp/860 and tcp/3260 are accessible to the initiator on the target.

Open-iSCSI

The Open-iSCSI project provides the initiator for Linux. There are a number of iSCSI initiators available but the Open-iSCSI project seems to provide the best solution in terms of completeness and stability. The project can be downloaded from http://www.open-iscsi.org or in Ubuntu by running apt-get install open-iscsi. The open-iscsi initiator has a configuration file located in /etc/iscsi/iscsid.conf. Look for the line node.startup and change its setting to automatic. It should look as follows:

node.startup = automatic

Start the initiator with */etc/init.d/open-iscsi restart*. Now we utilize the iscsiadm command to trigger a connection to the iSCSI target. To do so we use, *iscsiadm -m discovery -t st -p 10.19.12.25*. This command will produce 10.19.12.25:3260,1 iqn.2009-08.org.opensourc3:test.lun1.

Authentication and Configuration

The configuration file for the target is stored in */etc/iscsi/nodes/iqn.2009-08.org.opensourc3:test.lun1/10.19.12.25,3260,1/default*. The following commands will enable it to be configured:

iscsiadm -m node

iscsiadm -m node --targetname "iqn.2009-08.org.opensourc3:test.lun1" --portal "10.19.12.25:3260" --op=update --name node.session.auth.authmethod --value=CHAP

iscsiadm -m node --targetname "iqn.2009-08.org.opensourc3:test.lun1" --portal "10.19.12.25:3260" --op=update --name node.session.auth.username --value=theuser

iscsiadm -m node --targetname "iqn.2009-08.org.opensourc3:test.lun1" --portal "10.19.12.25:3260" --op=update --name node.session.auth.password --value=thepassword

Simply restart the initiator to login with */etc/init.d/open-iscsi restart*.

Using the iSCSI disk

The iSCSI disk will be available as the next available scsi device, typically /dev/sdb or /dev/sdc if you already have one or two SCSI type drives in your machine. Keep in mind that SATA drives are treated as SCSI devices in Linux. Once you have found the new drive (dmesg is a good option), simply run fdisk and partition the drive as you would a physical drive. Then format it with the desired filesystem as you would a physical drive.

The Only Conference Created by Developers for Developers

SEPTEMBER 14-17, 2009
SANTA CLARA, CA
HYATT REGENCY HOTEL

SNIA's 6th Annual Storage Developer Conference

Co-located with the following plugfests:
CIFS/SMB/SMB2
XAM (eXtensible Access Method)

WWW.STORAGE-DEVELOPER.ORG

- CIFS/SMB/SMB2
- NFS
- iSCSI
- Fibre Channel Protocols (including FCoE)
- XAM (eXtensible Access Method)
- Green and Energy Efficient IT Technologies
- Solid State Storage
- Storage for Virtual Platforms
- Storage for Cloud/Grid/Utility Computing
- Storage Utilizing Blade Technology
- Consumer/Personal Storage
- Storage Management Initiative-Specification
- Information and Data Management
- Storage Security
- Emerging Storage and Data Technologies
- Development Techniques and Tools
- Professional Development/Soft Skills
  - Making the Leap from Engineer to Manager
  - Working with Remote Technical Teams
- Leveraging Open Source Storage Technologies

TRY BEFORE YOU BUY
HANDS-ON-LAB FOR DEVELOPERS
(HOLD) SESSIONS PLANNED

SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2009

STORAGE NETWORKING INDUSTRY ASSOCIATION

System Configuration with Puppet

Puppet is a framework for automating the deployment of applications such as nginx.

by Juan Leaniz

Puppet is an Open Source framework that it allows you to automate tasks such as adding users,installing software or even updating system configurations regardless of the operating system you are running, it takes generic instructions from a configuration file and performs the task properly, according to the operating system it finds. How does Puppet do this? Simple, it uses Facter, a system assay tool. Facter creates a profile based on the system and sets a number of variables that puppet can use to identify the system's components. Puppet uses a server and a client for distributing configurations, so all you need is a "Puppet master", and clients.

Advantages of using Puppet
System administration can be a complicated job if you have to administer 100 servers, each one running very specific configurations. Here's where Puppet comes in, and provides us with a declarative language to express system configurations, and the possibility to update N number of servers automatically by modifying one single file.

The language has the following features:

- A wide number of Resource types to use (file, user, exec, service, etc).
- Classes and inheritance
- Configuration files templates
- Variables and arrays
- Conditional operators (if/else, case statements)
- Functions
- Dynamism

As you can see it's a rich language which allows us to do nearly everything you can think of.

Puppet command line tools and daemons
Puppet comes with a few command line tools that help us run Puppet manifests manually, sign our own SSL certificates, read documentation and run the daemons.

puppet: This tool runs site manifest scripts locally, it parses and evaluates them

puppetmasterd: This daemon runs on the host(s) that distribute the configuration files to the clients

puppetd: Puppetd is the client daemon which runs on every puppet client. It communicates with the host(s) running puppetmasterd to get the configuration files. It requests a specific site manifest and runs it locally

puppetca: The SSL CA server used to receive certificates requests from puppet clients. All puppet hosts are required to have their SSL certificates signed by the puppet master server before they can be authenticated to retrieve site manifests

puppetdoc: This is the command line tool for printing library reference documentation.

puppetrun: Command line tool for manually triggering Puppet configuration runs.

Introduction to the Puppet Language

Resources

Resources are built from a type, title and a list of attributes, and each resource type has its very own set of attributes.
For example the File resource type has the following attributes: owner, group, mode. By setting these attributes we define what permissions and owner we want the file to have. An example would be:

```
file { "/etc/shadow":
    owner => root,
    group => root,
    mode => 400,
}
```

The file owner will be the user “root” with group “root” and permissions 400. The most common resource types are: File, User, Exec, Package, Service. There is a complete list of resource types at http://reductivelabs.com/trac/puppet/wiki/TypeReference

Classes

Classes are defined using the class keyword and as in other languages they are wrapped in curly braces { }. Usually, services are defined within classes where the packages, configuration files and required services will also be included.

```
1. class unix {
2.         file {
3.             “/etc/shadow”:
4.                 owner => “root”,
5.                 group => “root”,
6.                 mode => 400;
7.         }
8. }
```

In line 1 we define the name for our new class to be “unix”. Lines 2 through 7 set the attributes for our file “/etc/shadow”.

Inheritance

We can inherit from a parent class in case we need to create a sub-class.

```
1. class freebsd inherits unix {
2.         File["/etc/shadow"] { group =>
   wheel }
3. }
```

In the first line we define the name for our new class like we did before but this time we add the inherits keyword and the name of the class to inherit from. Our “freebsd” class will inherit the File we defined in the “unix” class. In line 2 we change one of the attributes for “/etc/shadow” that applies specifically to “freebsd” systems.

Definitions

Definitions are similar to classes except that they support arguments but not inheritance.

```
1. define list_dir($path) {
2.         exec { “/bin/ls -l $path”:
3.                 unless => “/bin/test -d $path”,
4.         }
5. }
6. list_dir { puppet: path => “/some/dir” }
```

In line 1 we name our definition “list_dir” and give it 1 argument “$path” which we will set when we use the definition. In line 2 we introduce a new Resource type called “Exec”, it allows us to execute external commands from Puppet. The unless keywords will check if $path exists using the unix tool /bin/test and if it doesn't exist it won't execute our external command. Definitions are very useful if we wanted to do something like downloading our own version of nginx, unpacking it and compiling it from the source, where we would

use the Exec resource a lot.

Modules
Think of a situation where we have a collection of classes, definitions and resources so we want to put them all together and use them when we need to, that is when Modules become useful. For example, a module may contain all the necessary resources to configure a certain service, whenever we need to use that service we can just import our module.

Nodes
Nodes look like classes a lot, they even support inheritance, but the difference is that when a node connects to the Puppet master daemon it will look up it's name in the node definitions and that specific node definition will be evaluated.

```
1. node default {
2.         $node_name = "testing"
3.         include nginx
4. }
```

Line 1 defines our node with no specific name, just a default one that will be used for every node. Line 2 defines a variable which is used in the nginx class that we include in line 3. By including a class from a node definition we tell Puppet to evaluate that class for this specific node.

Deploying nginx using Puppet
In this article i will focus on deploying a web server using nginx and Puppet. This same configuration could be used to deploy as many instances of nginx as you need on any number of different servers. Nginx is an open source HTTP server with proxy capabilities, very stable, secure and scalable. If you want to read more about nginx check out http://www.nginx.net

Installing Puppet using package managers
Let's get down to business and start by installing Puppet and Puppet-server. First we will need to enable EPEL packages on CentOS or RHEL systems:

```
su -c 'rpm -Uvh
http://download.fedora.redhat.com/pub/epel/5/i
386/epel-release-5-3.noarch.rpm'
```

That will enable us to simply install Puppet using RPMs:

```
yum install puppet-server puppet
```

On debian based systems you would run:

```
apt-get install puppet puppet-server
```

It may install a few needed dependencies as well (eg: Ruby, facter).

Installing Puppet from source
Before installing Puppet from the source code you will need to make sure you have Ruby on your system. Puppet has the following requirements to be compiled from source:

- Ruby 1.8.2 or later
- Git (if you want to check out the source instead of downloading a tar.gz file)
- Facter

Run the following commands:

```
SETUP_DIR=~/git
mkdir -p $SETUP_DIR
cd $SETUP_DIR
git clone git://github.com/reductivelabs/facter
git clone git://reductivelabs.com/puppet
```

These commands will create a new directory for our setup, and checkout the latest source for Puppet and Facter using the git

repositories. After that, we have to tell Ruby how to find Puppet:

PATH=$PATH:$SETUP_DIR/facter/bin:$SETUP_DIR/puppet/bin

RUBYLIB=$SETUP_DIR/facter/lib:$SETUP_DIR/puppet/lib

export PATH RUBYLIB

Finally, we can run Puppet's install script:
./install.rb

Configuring Puppet

To configure our Puppet environment we will need to modify a few files. Namely site.pp, puppet.conf and nginx.erb. The first one is our puppet manifest and it contains one class for nginx and our nodes' information. Nginx.erb is a template for nginx's configuration file nginx.conf. Open /etc/puppet/puppet.conf with your favourite text editor and add the following:

```
[puppetmasterd]
        certname = puppet
```

"puppet" will be your puppet master's FQDN, so make sure you set that right or Puppet will complain about it. We need to make sure we put our template file in the right directory, run the following command to find out where your template directory is:

puppet --configprint templatedir

Then if it doesn't exist:

mkdir /var/lib/puppet/templates

Start puppetmaster:
/etc/init.d/puppetmaster start

Now that puppetmaster is up we can go on and edit our manifest file and template.

Nginx deployment using Puppet Classes and Templates

Puppet will read /etc/puppet/manifests/site.pp as the default file containing our configuration. Open /etc/puppet/manifests/site.pp and paste the following data:

```
1. class nginx {
2.
3.      $rootdir = "/var/www/html"
4.
5.    package { "nginx":
6.              ensure => installed;
7.    }
8.
9.    file { "/etc/nginx/nginx.conf":
10.             owner => root,
11.             group => root,
12.             mode => 644,
13.             content =>
    template("nginx.erb"),
14.         }
15.
16.         user { "nginx":
17.              ensure => present,
18.         }
19.
20.           service {
21.              "nginx":
22.                   ensure => running,
23.                   enable => true,
24.                   hasrestart => true,
25.                   hasstatus => false,
26.                   require =>
    Package["nginx"],
27.                   subscribe =>
    File["/etc/nginx/nginx.conf"];
28.           }
29.      }
30.
31.      node default {
```

```
32.             $node_name = "testing"
33.             include nginx
34.     }
```

By now you should be familiar with most of the resource types in this manifest. However there are a few new things that I didn't discuss before. In lines 5 through 7 we define a Package resource called nginx and set the ensure attribute to installed. This means that when we run this manifest it will make sure the “nginx” package is installed on the system before implementing the configuration file. Lines 16-18 introduce the User resource, we use this resource to ensure that the nginx user exists on the system. Since our site manifest is ready we can continue by creating our template for nginx's configuration file in /var/lib/puppet/templates/nginx.erb which is listed on page 27.

This is simply a template for an nginx configuration file to server static content over HTTP. You will notice in lines 27, 30 and 35 the <%= and %> symbols. What we do here is add the variable names we defined in our nginx class, these variable names will be replaced when the puppet client executes the manifest file with the actual root directory and server name for our node.

Deploying the configuration

On the servers we want to deploy nginx we run puppetd so it can connect to our puppet master server and deploy.

/usr/sbin/puppetd --server puppet --waitforcert 60 --ssldir /var/lib/puppet/ssl

As I explained before when I commented on the command line tools, we need to make sure our client's SSL certificate is signed by the puppet master server, so on our master server we run:

puppetca --list

You will see a list of certificates you need to sign with:

puppetca --sign <server-name>

If everything worked as it should have, you now have a working nginx setup on your puppet client's box.

Conclusion

Using Puppet will save a lot of people plenty of time dealing with configuration files and system backups, this method can also be used to deploy configurations on a number of environments, for instance, we could deploy any amount of nginx instances on different virtual servers instantly. This tool is a must for any serious IT professional who needs to handle a considerable quantity of servers.

About the Author

Juan Leaniz is an experienced Linux professional based out of Montevideo, Uruguay. Juan is currently the CIO for Carbon Mountain LLC.

Code Listing: nginx.erb

```
1. user nginx nginx;
2. worker_processes 4;
3.
4. error_log logs/error.log;
5. pid        logs/nginx.pid;
6.
7. events {
8.     worker_connections 1024;
9. }
10.
11.     http {
12.         include       mime.types;
13.         default_type application/octet-stream;
14.
15.         log_format main '$remote_addr - $remote_user [$time_local] $request '
16.                         '"$status" $body_bytes_sent "$http_referer" '
17.                         '"$http_user_agent" "$http_x_forwarded_for"';
18.
19.         access_log logs/access.log main;
20.         server_names_hash_bucket_size 128;
21.         sendfile        on;
22.         tcp_nopush      on;
23.         keepalive_timeout 65;
24.
25.         server {
26.             listen 80;
27.             server_name <%= node_name %> www.<%= node_name %>;
28.
29.             location / {
30.                 root <%= rootdir %>;
31.                 index index.htm index.html
32.             }
33.
34.             location ~* ^.+\.(jpg|jpeg|gif|png|ico|css|zip|tgz|gz|
    rar|bz2|doc|xls|exe|pdf|ppt|txt|tar|mid|midi|wav|bmp|rtf|js|mov)$ {
35.                 root <%= rootdir %>;
36.             }
37.     }
```

Linux Hypervisor Platform

is

/* coming soon */

Building IP Networks

Best practices for building small but scalable IP networks. A primer for new system administrators

by John Buswell

Last month I attended a VMware security presentation ran by a reputable training company. To my surprise the presenter was demonstrating how to perform an ARP-cache poisoning attack against VMware ESX. I asked them why they were demonstrating such an attack as basic VLANs would guard against the specific attack they were demonstrating. To my surprise, he explained that the majority of IT managers that attend their courses have large flat network segments. Trying to build a Unified Computing or Cloud Computing environment on top of such networks is just asking for trouble. So this article is intended as a preparation guide for Network Administrators.

VLANs

Virtual LANs are logical broadcast domains, or network segments. Broadcast traffic within a VLAN is delivered to the members of that VLAN as if they were connected on a dedicated network segment, such as via a hub or switch. The best practice is to assign one VLAN per IP subnet, and assign one IP subnet to a logical group. A logical group might be the server network, the wifi network, the sales department or the engineering department.

Layer 3 switching and routers

Layer 3 switches or routers can be used to route traffic between IP subnets isolated by separate VLANs. A common mistake made by Network Administrators is that after configure the VLANs and the IP networks they cannot pass traffic between them. In order for traffic to pass between IP networks, they have to be routed. This means the router or Layer 3 switch has to have an interface in VLAN X and an interface in VLAN Y. The clients on VLAN X must have the layer 3 switch or router's IP configured as the default gateway. For example, 192.168.10.0/24 on VLAN 123 and 172.30.10.0/24 on VLAN 300. If the router is configured as 192.168.10.1 and 172.30.10.1. Clients on VLAN 123 will route traffic for 172.30.10.0/24 to the gateway which will forward the traffic out to VLAN 300. The clients in VLAN 300 must have a return path (the default gateway of 172.30.10.1) in order to send back a response.

802.1Q

VLAN Tagging or 802.1Q adds a VLAN tag to the Ethernet frame header between the Source MAC Address and the EtherType size. This VLAN tag contains information tagging that particular Ethernet frame to a specific VLAN. In a Unified Computing or Virtualized environment where the virtual guests may belong to different VLANs, the 802.1Q VLAN tagging allows a single link between a switch and the Hypervisor to be shared for multiple VLANs. The switch will add the 802.1Q tag to the frame, and when it arrives at the Hypervisor, it will process it utilizing the correct VLAN association. This network isolation is lost on a flat Ethernet segment.

Conclusion

Flat ethernet segments lead to network performance problems, increased security risks and messy networks. So use VLANs!

Better usability improves your sales by 100%.

Research backs up what we all knew: better design = bigger profits.

Find out how  www.locusfoc.us

The 100% improvement claim is based on the ROI usability report published by the Nielsen Norman Group.