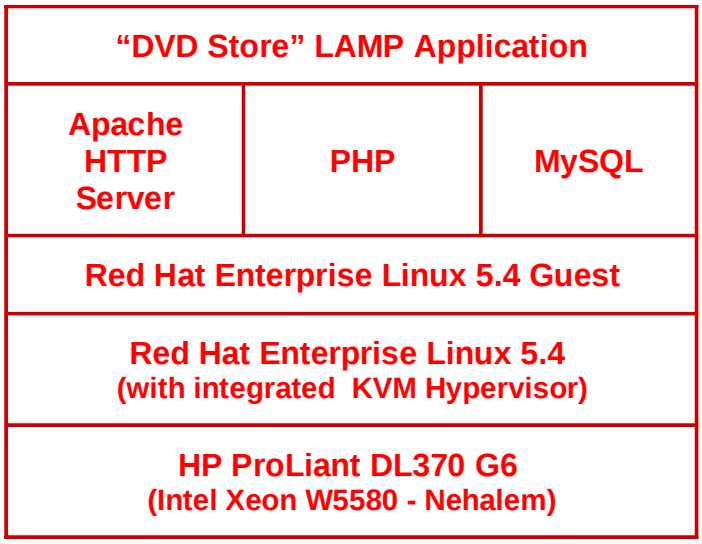
Red Hat Reference Architecture Series

Scaling the LAMP Stack in a Red Hat Enterprise Virtualization Environment



Version 1.0
August 2009

redhat.

Scaling the LAMP Stack in a Red Hat Enterprise Virtualization Environment

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

1 Executive Summary

This paper describes the performance and scaling of the industry-standard LAMP web application stack running in Red Hat Enterprise Linux 5.4 guests on a Red Hat Enterprise Linux 5.4 host with the KVM hypervisor. The host system was deployed on an HP ProLiant DL370 G6 server equipped with 48 GB of RAM and comprising dual sockets each with a 3.2 GHz Intel Xeon W5580 Nehalem processor with support for hyper-threading technology, totaling 8 cores and 16 hyper-threads.

The workload used to exercise the LAMP stack was the open source DVD-Store application. DVD-Store is a complete three tiered e-commerce test application, representing an on-line DVD-Store. The Presentation Layer represents customers using web browsers to search for and purchase DVDs on the on-line DVD-Store. The Application Layer consists of the Apache HTTP web server which hosts the web pages that constitute the application. The web pages, written in PHP contain code that read the requests submitted by the user, access the back-end MySQL database and write the appropriate HTML code back to the browser. The Database Layer consists of the MySQL Database Server.

Scaling Up A Virtual Machine
First, the performance of the DVD-Store application was measured by loading a single VM on the server, and assigning it 1, 2, 4, 6 or 8 vCPUs. The performance scales over 420% as the VM expands from 1 hyper-thread to a complete 4 core/8 hyper-thread server.

Scaling Out Virtual Machines
A second series of tests involved scaling out multiple independent VMs each comprising 2, 4 or 8 vCPUs up to a total of 16 vCPUs on an 8 core/16 hyper-thread Nehalem server. As an example, Red Hat tested the performance one to four concurrent 4-vCPU VMs running DVD-Store. The four VMs performed over 350% of the operations rate of the single VM, with each of the guests retaining nearly an average 90% of the single guest rate.

The data presented in this paper clearly establishes that Red Hat Enterprise Linux 5.4 virtual machines using the KVM hypervisor on a HP ProLiant DL370 provide an effective production-ready platform for hosting multiple virtualized LAMP web application stacks. The combination of low virtualization overhead and the ability to both scale-up and scale-out contribute to the effectiveness of KVM for LAMP web application stack. The number of actual users and throughput supported in any specific customer situation will, of course, depend on the specifics of the customer application used and the intensity of user activity. However, the results demonstrate that in a heavily virtualized environment, good throughput was retained even as the number and size of guests/virtual-machines was increased up until the physical server was fully subscribed.

2 Red Hat Enterprise Virtualization (RHEV) - Overview

2.1 Red Hat Enterprise Virtualization (RHEV) - Portfolio

Server virtualization offers tremendous benefits for enterprise IT organizations – server consolidation, hardware abstraction, and internal clouds deliver a high degree of operational efficiency. However, today, server virtualization is not used pervasively in the production enterprise datacenter. Some of the barriers preventing wide-spread adoption of existing proprietary virtualization solutions are performance, scalability, security, cost, and ecosystem challenges.

The Red Hat Enterprise Virtualization portfolio is an end-to-end virtualization solution, with use cases for both servers and desktops, that is designed to overcome these challenges, enable pervasive datacenter virtualization, and unlock unprecedented capital and operational efficiency. The Red Hat Enterprise Virtualization portfolio builds upon the Red Hat Enterprise Linux platform that is trusted by millions of organizations around the world for their most mission-critical workloads. Combined with KVM, the latest generation of virtualization technology, Red Hat Enterprise Virtualization delivers a secure, robust virtualization platform with unmatched performance and scalability for Red Hat Enterprise Linux and Windows guests.

Red Hat Enterprise Virtualization consists of the following server-focused products:

1. Red Hat Enterprise Virtualization Manager (RHEV-M) for Servers: A feature-rich server virtualization management system that provides advanced management capabilities for hosts and guests, including high availability, live migration, storage management, system scheduler, and more.

2. A modern hypervisor based on KVM (Kernel-based Virtualization Machine) which can be deployed either as:

    - Red Hat Enterprise Virtualization Hypervisor (RHEV-H): A standalone, small footprint, high performance, secure hypervisor based on the Red Hat Enterprise Linux kernel.

    Or

    - Red Hat Enterprise Linux 5.4: The latest Red Hat Enterprise Linux platform release that integrates KVM hypervisor technology, allowing customers to increase their operational and capital efficiency by leveraging the same hosts to run both native Red Hat Enterprise Linux applications and virtual machines running supported guest operating systems.

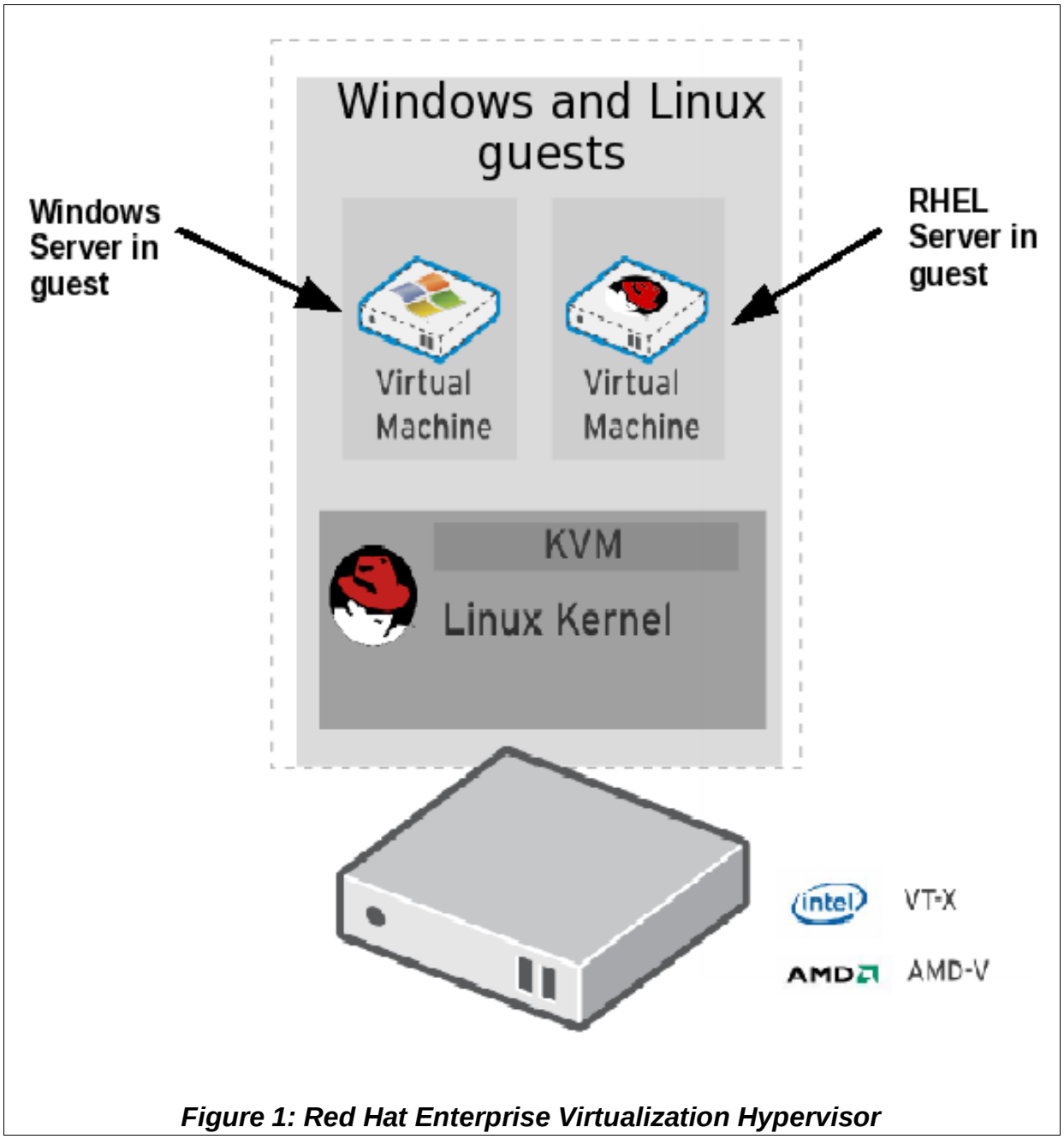


Figure 1: Red Hat Enterprise Virtualization Hypervisor

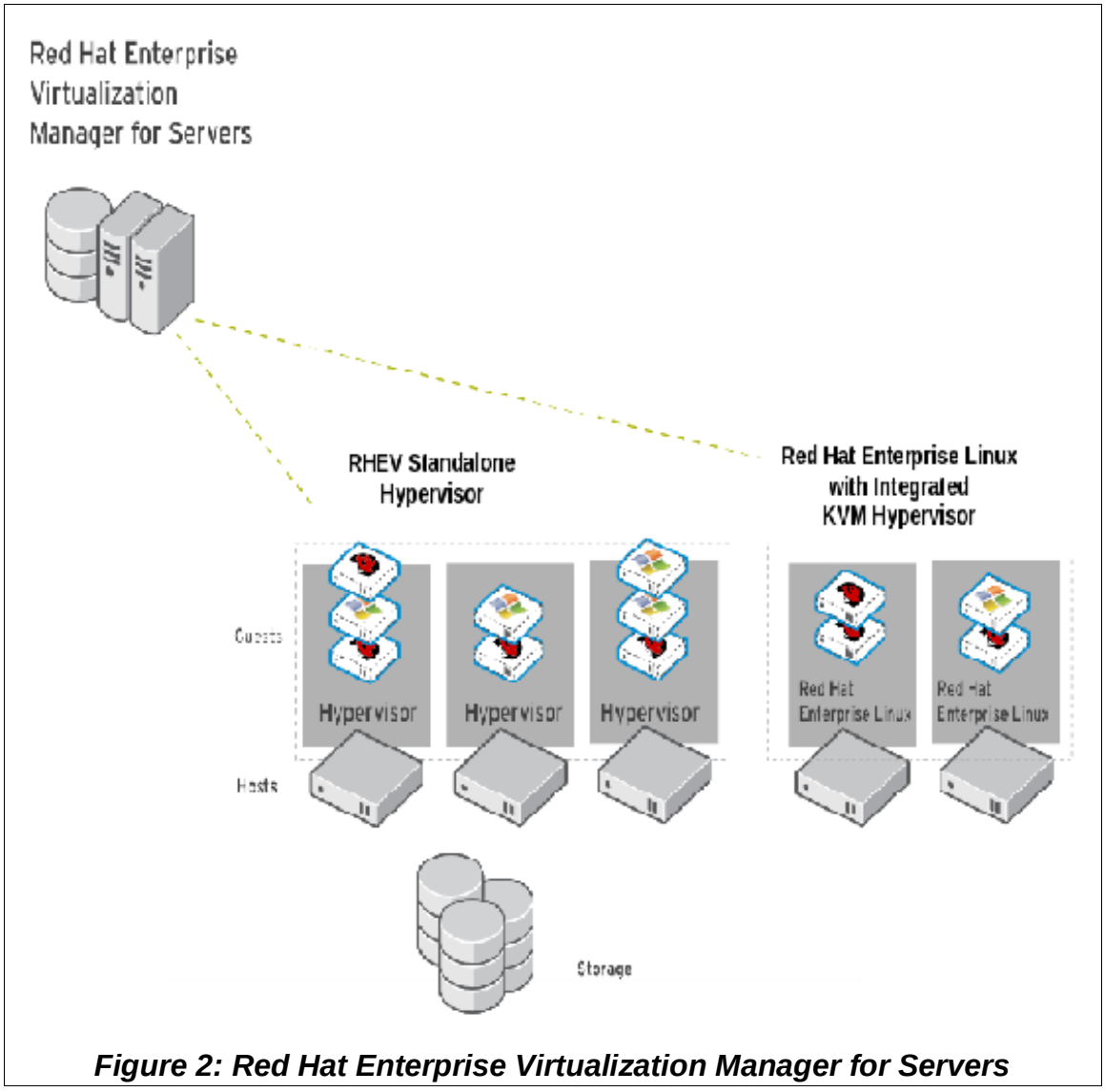


Figure 2: Red Hat Enterprise Virtualization Manager for Servers

2.2 Kernel-based Virtualization Machine (KVM)

A hypervisor, also called virtual machine monitor (VMM), is a computer software platform that allows multiple (“guest”) operating systems to run concurrently on a host computer. The guest virtual machines interact with the hypervisor which translates guest I/O and memory requests into corresponding requests for resources on the host computer.

Running fully-virtualized guests, i.e., guests with unmodified guest operating systems, used to require complex hypervisors and used to incur a performance penalty for emulation and translation of I/O and memory requests.

Over the last couple of years as chip vendors (Intel and AMD) have been steadily adding CPU features that offer hardware enhancements to the support virtualization. Most notable are:

1. First generation hardware assisted virtualization: Removes the need for hypervisor to scan and rewrite privileged kernel instructions using Intel VT (Virtualization Technology) and AMD's SVM (Secure Virtual Machine) technology.

2. Second generation hardware assisted virtualization: Offloads virtual to physical memory address translation to CPU/chip-set using Intel EPT (Extended Page Tables) and AMD RVI (Rapid Virtualization Indexing) technology. This provides significant reduction in memory address translation overhead in virtualized environments.

3. Third generation hardware assisted virtualization: Allows PCI I/O devices to be attached directly to virtual machines using Intel VT-d (Virtualization Technology for directed I/O) and AMD IOMMU. And SR-IOV (Single Root I/O Virtualization) which allows special PCI devices to be split into multiple virtual devices. This provides significant improvement in guest I/O performance.

The great interest in virtualization has led to the creation of several different hypervisors. However, many of these predate hardware-assisted virtualization, and are therefore some-what complex pieces of software. With the advent of the above hardware extensions, writing a hypervisor has become significantly easier and it is now possible to enjoy the benefits of virtualization while leveraging existing open source achievements to date.

Kernel-based Virtual Machine (KVM) turns a Linux kernel into a hypervisor. Red Hat Enterprise Linux 5.4 provides the first commercial-strength implementation of KVM, which is developed as part of the upstream Linux kernel.

2.2.1 Traditional Hypervisor Model

The traditional hypervisor model consists of a software layer which multiplexes the hardware among several guest operating systems. The hypervisor performs basic scheduling and memory management, and typically delegates management and I/O functions to a special, privileged, guest.

Today's hardware, however is becoming increasingly complex. The so-called “basic” scheduling operations have to take into account multiple hardware threads on a core, multiple

cores on a socket, and multiple sockets on a system. Similarly, on-chip memory controllers require that memory management take into effect the Non-Uniform Memory Access (NUMA) characteristics of a system. While great effort is invested into adding these capabilities to hypervisors, we already have a mature scheduler and memory management system that handles these issues very well – the Linux kernel.

2.2.2 Linux as a Hypervisor

By adding virtualization capabilities to a standard Linux kernel, we can enjoy all the fine-tuning work that has gone (and is going) into the kernel, and bring that benefit into a virtualized environment. Under this model, every virtual machine is a regular Linux process scheduled by the standard Linux scheduler. Its memory is allocated by the Linux memory allocator, with its knowledge of NUMA and integration into the scheduler.

By integrating into the kernel, the KVM 'hypervisor' automatically tracks the latest hardware and scalability features without additional effort.

2.2.3 A Minimal System

One of the advantages of the traditional hypervisor model is that it is a minimal system, consisting of only a few hundred thousands lines of code. However, this view does not take into account the privileged guest. This guest has access to all system memory, either through hypercalls or by programming the DMA hardware. A failure of the privileged guest is not recoverable as the hypervisor is not able to restart it if it fails.

A KVM based system's privilege footprint is truly minimal: only the host kernel plus a few thousand lines of the kernel mode driver have unlimited hardware access.

2.2.4 KVM Summary

Leveraging new silicon capabilities, the KVM model introduces an approach to virtualization that is fully aligned with the Linux architecture and all of its latest achievements. Furthermore, integrating the hypervisor capabilities into a host Linux kernel as a loadable module simplifies management and improves performance in virtualized environments, while minimizing impact on existing systems.

Red Hat Enterprise Linux 5.4 incorporates KVM-based virtualization in addition to the existing Xen-based virtualization. Xen-based virtualization, of course, remains fully supported for the life of the Red Hat Enterprise Linux 5 family.

An important feature of any Red Hat Enterprise Linux update is that kernel and user APIs are unchanged, so that Red Hat Enterprise Linux 5 applications do not need to be rebuilt or re-certified. This extends to virtualized environments: with a fully integrated hypervisor, the application binary interface (ABI) consistency offered by Red Hat Enterprise Linux means that applications certified to run on Red Hat Enterprise Linux on physical machines are also certified when run in virtual machines. So the portfolio of thousands of certified applications for Red Hat Enterprise Linux applies to both environments.

3 LAMP Stack - Overview

LAMP is an acronym which stands for:

1. Linux, the operating system
2. Apache HTTP server, the web server
3. MySQL, the database management system or database server (sometimes substituted with PostgreSQL)
4. PHP, the scripting language (sometimes substituted with other scripting/programming languages - Python, Perl, Ruby)

As the web has evolved from initially serving static web pages to its current state where the ability to handle dynamic pages and web services is a standard requirement, many solution stacks designed to augment the basic (HTTP) web server have become available. The following lists include some of the more popular web server stacks.

Non-Microsoft (also available on Windows):

1. LAMP stack
2. Tomcat Java-based stack
3. Full JEE (Java Enterprise Edition) stack

Microsoft (available only on Windows):

1. WISA stack – Windows (operating system), Internet Information Services (web server), Microsoft SQL Server (database) and ASP (scripting language).
2. Full .NET stack

Despite the feature-richness of J2EE and .NET, and the fact that LAMP has not had the same level of commercial promotion, LAMP continues to enjoy unprecedented success and market share.

What is the attraction to the LAMP stack for developers around the world? In part, it is the open source underpinnings of LAMP components. They are freely available, easily configured, and very robust. They are in a constant state of development and improvement, adding features suggested by the user community at large. They can be easily deployed, fully configured, and maintained with a minimal amount of effort. In short, the LAMP stack allows developers to do what they do best: develop, without spending a disproportionate amount of time in the administrative details.

All these elements are addressed in the package of LAMP components provided by Red Hat Enterprise Linux. Red Hat Enterprise Linux helps to assure that configuring and administering a LAMP server will be as painless a process as is possible.

3.1 Linux

The most important element of the LAMP stack is the Linux operating system installed on the server. With dozens of Linux distributions available, the choice can be a bit perplexing. Of the available distributions, however, Red Hat Enterprise Linux maintains a stronghold the in enterprise-grade LAMP servers for several reasons. It offers a huge ecosystem of hardware and software partners, offering both services and certified solutions, making Red Hat the industry leader. This powerful combination provides:

- Thousands of certified applications from Independent Software Vendors (ISVs)
- Hundreds of certified hardware systems and peripherals from leading OEM vendors spanning multiple processor architectures
- A range of partner programs
- Comprehensive service offerings, up to 24x7 support with 1-hour response, available from Red Hat and selected ISV/OEM partners
- Excellent performance, security, scalability, and availability, with audited industry benchmarks
- Open source technologies rigorously tested and matured through the Red Hat sponsored Fedora project
- With each major version, stable application interfaces and seven years of product support

3.2 Apache

The second element of the LAMP stack is the Apache web server. The web server is the application that accepts request for pages from a browser, interprets the request, and returns the results. For static HTML pages, it simply retrieves the HTML file that the browser requests. For dynamic pages, when a browser requests a page, the web server transfers control to a program or module that interprets the script and returns the results.

Apache is another open source tool with a rich and mature code base. Created in the early 1990s, the HTTP daemon (httpd) package today operates nearly 50% of the web servers worldwide.

Apache is highly configurable and modular. A completely customized configuration can be achieved simply by modifying the text configuration file, */etc/httpd/conf/httpd.conf*. This file is commented in depth, providing configuration guidance to both the novice and expert webmaster. The code base can also be extended by means of *modules*, chunks of code that can be loaded at the time the server is started or dynamically, as needed. Hundreds of these modules — most developed by interested third parties — exist in the official Apache code base today.

Apache is part of the default installation of Red Hat Enterprise Linux. In short, installing Apache does not require the additional action of selecting it as an optional package during installation. The Apache package is referred to as httpd in the standard Red Hat Enterprise Linux configuration. Configuration and related files are named accordingly.

The following June 2009 survey from Netcraft Ltd. (of 238,027,855 sites) shows market share of web servers across all domains. Apache remains in the lead, as it has since 1996!
http://www.netcraft.com/survey/

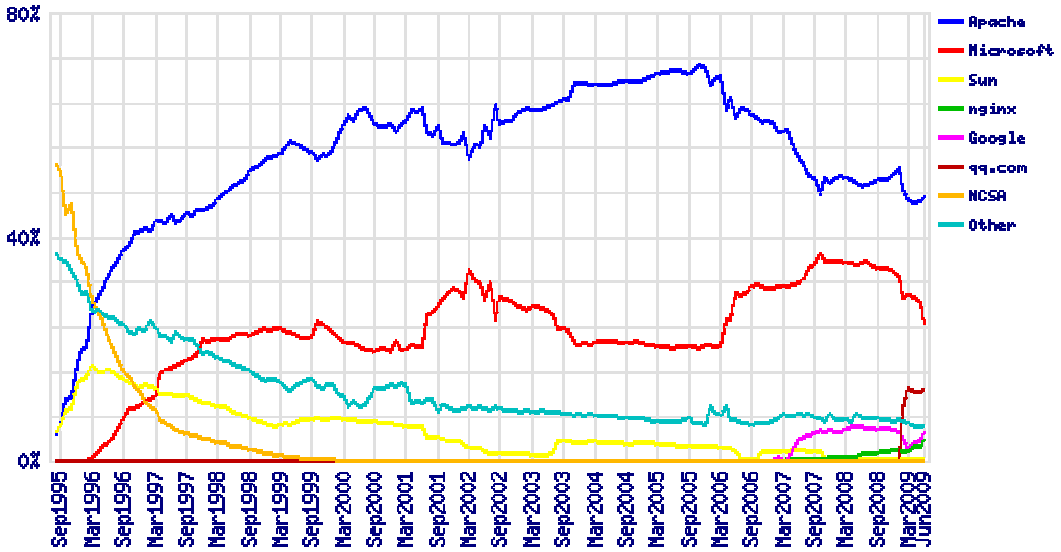


Figure 3: Apache Market Share

Also, the clear leader amongst web servers used by the million busiest websites is Apache with a 66% share. It has a 47% lead over its closest competitor, Microsoft-IIS, much greater than on the web as a whole.

Server Share amongst the Million Busiest Sites, March 2009

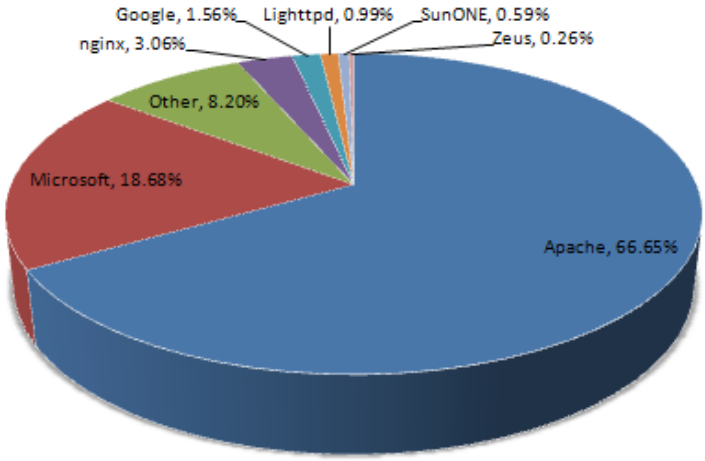


Figure 4: Apache Market Share Among Busiest Sites

3.3 MySQL

The third element of the LAMP tool set is the MySQL database, another robust open source tool that has revolutionized the way web pages, graphics, tables, and data sets of all sorts are served on the web. Databases in general, and MySQL in particular, have made it possible to build and present fully dynamic websites, capable of presenting content in real time. They have also helped to further the goal of separating content from formatting, speeding the web site load time while making them far more manageable than in the past.

3.4 PHP

PHP was originally an acronym for Personal Home Page. It began in 1994 as a set of Common Gateway Interface (CGI) binaries written in the C programming language. Today, PHP is a general-purpose scripting language that is especially suited for web development and can be easily embedded into HTML. PHP generally runs on a web server, taking PHP code for input and creating web pages as output. It can also be used for command-line scripting and client-side GUI applications.

In just a few short years, PHP has become one of the predominant scripting languages on the web. It is another integral element of LAMP development and can be found anywhere from personal homepages to content management systems (such as Drupal) to large-scale corporate intranets. With a relatively easy syntax and open source licensing, webmasters and developers around the world have migrated to PHP from more difficult and syntactically challenging scripting languages like Perl.

The latest version of PHP fully supports object-oriented syntax and provides a command line capability for quick code testing.

PHP is part of the default installation of Red Hat Enterprise Linux. However, in order to interact properly with a MySQL database, the php_mysql module must be chosen at install time. This module provides the interaction between PHP and MySQL in the form of an Apache module.

According to a recent survey by Nexen.net, PHP has a market share of more than 30%. The number of internet sites using PHP was around 20 million in 2006. However, this figure does not take into consideration the growing number of internal corporate servers used for intranet applications or development purposes – statistics about this usage are still unclear.

4 Test Configuration

4.1 Hardware Configuration

HP ProLiant DL370 G6	Dual Socket, Quad Core, Hyper Threading Technology (Total of 16 processing threads) Intel(R) Xeon(R) CPU W5580 @ 3.20GHz
	12 x 4 GB DIMMs- total: 48 GB
	6 x 146 GB SAS 15K dual port drives

Table 1: Hardware

4.2 Software Configuration

Red Hat Enterprise Linux 5.4 - Beta	2.6.18-155.el5
KVM	kvm-83-80.el5
Apache Httpd	v2.2.3
MySQL	v5.0.77
PHP	v5.1.6
APC	v3.0.19

Table 2: Software

5 Test Methodology

5.1 Workload

The software system used to demonstrate the successful deployment of the LAMP stack is the *DVD-Store* Application. The *DVD-Store* Application is a complete three tiered e-commerce test application, representing an on-line DVD-Store. The *Presentation* Layer represents customers using web browsers to search for and purchase DVDs on the on-line DVD-Store. The *Application* Layer consists of the Apache HTTP web server (from the Apache Software Foundation) which hosts the web pages that constitute the application. The web pages, written in PHP contain code that read the requests submitted by the user, access the backend MySQL database and write the appropriate Hypertext Markup Language (HTML) code back to the browser. The *Database* Layer consists of the MySQL® Database Server (from MySQL AB).

The *DVD-Store* Release 2 (DS2) is available from http://linux.dell.com/dvdstore. It includes support for a backend database component, a PHP web application layer, and driver programs to simulate users. The goal in designing the database component as well as the mid-tier application was to utilize many advanced database features (transactions, stored procedures, triggers, referential integrity) while keeping the database easy to install and understand. The DS2 workload may be used to test databases or as a stress tool for any purpose. The code is licensed under the GNU General Public License (GPL).

The DS2 distribution includes code for the MySQL database. Included are data generation programs, shell scripts to build data for the *DVD-Store*, database build scripts and stored procedures, PHP web pages, and a driver program to simulate web browser users.

DS2 comes in 3 standard sizes:

Database	Size	Customers	Orders	Products
Small	10 MB	20,000	1,000/month	10,000
Medium	1 GB	2,000,000	100,000/month	100,000
Large	100 GB	200,000,000	10,000,000/month	1000000

Table 3: DVD Store Standard Sizes

A customized 4 GB database size was used for the majority of the testing, while the large database size was used for the scale up data. This custom 4 GB database consisted of 8,000,000 customers, 400,000 orders per month, and 140,000 products.

The workload used in this paper simulates an on-line transaction processing environment using the DVD-Store application.

The *DVD-Store* application consists of four web pages or transaction types:

1. *Login,*

2. *NewCustomer,*
3. *Browse* and
4. *Purchase*.

Customers who have already created an account access the *Login* page to start a new order.

The code in the *Login* page checks the user name and password entered by the customer, and verifies the customer's account number. Additionally, the page returns the previous ten titles ordered by the customer and, for each title, a title recommended by others who also enjoyed the ordered title.

New customers use the *NewCustomer* page to create a new account by entering a username, personal data and credit card information. The *NewCustomer* code first checks that the new username is not already in use, and then inserts a new row in the CUSTOMERS table with all the information entered on the page.

After successful login or new account creation, the customer is presented the *Browse* page, which enables the customer to search for DVDs by title, lead actor or category. Titles returned by the searches may be added to the customer's shopping cart.

Finally, the *Purchase* page allows the user to specify quantities, optionally delete titles from the shopping cart, and finally complete the purchase. The code in the *Purchase* page first checks that there is sufficient quantity in stock for every title in the order, then updates the appropriate database tables. For simplicity, there is no partial order handling in this version.

The entire LAMP Stack configuration is a three-tier model. Tier 1 executes the driver that emulates the activities of Web users. Tier 2 comprises Web application servers that intercept the requests and send database transactions to a DB2 9 data server. All three tiers were configured to run within the same virtual machine instance.

The simulated workload ran 35 users for each vCPU used in the test. As additional vCPUs were added the number of simulated users was increased in increments of 35 users, starting with 35 users for a one vCPU guest and increasing to 280 users for a eight vCPU guest. To simulate a real-world scenario, the think time was set to a default of 0.25 seconds. Each test ran for 10 minutes.

The throughput metric used was the DVD-Sore Operations Per Minute (OPMs) reported by the application. OPMs can also be thought of as transactions per minute.

5.2 Configuration & Workload

Demonstrating the scaling of KVM based virtualization meant several aspects of the workload and configuration were scaled with the size of the guest. The database was held constant to demonstrate that the scaling was the effect of scaling the guests and not the application.

A constant database size was used for all the testing in this paper. While DVD Store is distributed with support for 10 MB, 1GB, and 100GB, a customized database size of 4GB was used.

The system is configured with two Intel W5580 processors. These are 3.2 GHz quad-core processors that support Hyper-Threading Technology. While each thread is a CPU for Red Hat Enterprise Linux, two threads share the processing power of the hyper threaded core with hardware support. For 2-vCPU guests, a core was assigned to each virtual machine using the `numactl` command. Two cores from the same processor were assigned to 4-vCPU guests. A full processor was allocated to 8-vCPU guests.

The host system has 48 GB of memory. Evenly distributing this to the vCPUs would allow 3GB per vCPU, however, only 2.5GB was used leaving some memory for the hypervisor and for guests that may have oversubscribed the available CPUs. The MySQL option *innodb_buffer_pool_size* was set to 80% of the guest memory.

Since the amount of memory and the number vCPUs changed for various tests, the workload applied to the system was correspondingly scaled. A load of 35 users per vCPU was selected after some test trials demonstrate that this sufficiently loaded a guest.

vCPUs per Guest	Guest Memory	User Load	InnoDB Buffer Pool
1	2.5 GB	35	2 GB
2	5 GB	70	4 GB
4	10 GB	140	8 GB
6	15 GB	210	12 GB
8	20 GB	280	16 GB

Table 4: Guest Configuration and Workload Parameters

5.3 Performance Test Plan

Using the guidance of the parameters set forth in the previous section (**Configuration & Workload**), limited additional tuning was performed as the test configurations were changed.

Scale-out:
The scale-out data highlights the results of the increasing of the number of independent concurrent 2-vCPU, 4-vCPU, or 8-vCPU guests running the DVD Store LAMP application. The single guest numbers reported are the average of the data from single instance runs for all the guest of the same size.

Scale-up:
The scale-up data was collected by increasing the number of vCPUs and the amount of memory when running the workload for a single guest.

Virtualization Efficiency:
Virtualization efficiency is the data when all the 8 cores (16 hyper-threads) are allocated to executing the DVD Store LAMP application using bare metal (no virtualization), eight 2-vCPU guests, four 4-vCPU guests, and two 8-vCPU guests.

The DVD Store load driver specified the parameters specified in **Table 6,** otherwise the default values were used for the remaining parameters.

Parameter	Value
run_time	10
n_threads	35 * #vCPUs
db_size_str	custom
warmup_time	0
ramp_rate	0
think_time	0.25
pct_newcustomers	10

Table 5: DVD Store Driver Options

5.4 Tuning & Optimizations

Host:
The host was installed with the first Red Hat Enterprise Linux 5.4 Beta which was available via RHN. The primary purpose of this server is to be a KVM hypervisor to guest virtual machines. Several processes that are unnecessary for this role were disabled using `chkconfig`. Additionally, SELinux was disabled. The services which were disabled were the following:

cpuspeed	kdump	xfs
haldaemon	isdn	saslauthd
yum-updatesd	iptables	ricci
smartd	ip6tables	pcscd
setroubleshoot	hplip	iscsi
sendmail	hidd	iscsid
rpcgssd	gpm	libvirtd
rpcidmapd	cups	modclusterd
rpcsvcgssd	bluetooth	cmirror
rhnsd	avahi-daemon	xend
pcscd	restorecond	xendomains
mdmonitor	auditd	
mcstrans	xinetd	

The host allocated memory in the form of Huge Pages (2048 kB) which was then made available to guests. With the guest using memory backed by huge pages, the number of accesses to translation lookaside buffers (TLB) should be reduced, thereby improving performance. To allocate huge pages, the following commands were issued on the host. First, the current allocation of huge pages is checked, then over 40 GB were allocated and confirmed. The last command mounts the huge table file system which will be passed to the guest.

```
# tail /proc/meminfo
Bounce:              0 kB
CommitLimit:  33115432 kB
Committed_AS:    99592 kB
VmallocTotal: 34359738367 kB
VmallocUsed:    284828 kB
VmallocChunk: 34359451703 kB
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
Hugepagesize:     2048 kB

# echo 20500 > /proc/sys/vm/nr_hugepages

# tail /proc/meminfo
```

```
Bounce:               0 kB
CommitLimit:  12123432 kB
Committed_AS:    97028 kB
VmallocTotal: 34359738367 kB
VmallocUsed:    284828 kB
VmallocChunk: 34359451703 kB
HugePages_Total: 20500
HugePages_Free:  20500
HugePages_Rsvd:      0
Hugepagesize:     2048 kB

# mount -t hugetlbfs hugetlbfs /mnt/libhugetblfs
```

The host was equipped with six internal drives. One was used for the OS installation. Four drives were configured into four equal partitions for use by the guests. The remaining drive was unused.

Guest:

The guests were also installed with the Red Hat Enterprise Linux 5.4 Beta. The same set of services disabled on the host were disabled on each guest.

The guests were started using the qemu-kvm command line. This allowed for the use of `numactl` to specify CPU and memory locality, the use of huge pages, and specifying the cache mechanism of the disks. The example which follows following allocates a guest with:

- two CPUs (**-smp 2**)
- bound CPUs a single core (**--physcpubind=7,15**)
- 5GB of memory(**-m 5120**)
- restricting memory from NUMA node 1 (**-m 1**)
- huge page memory (**--mem-path /mnt/libhugetblfs**)
- two drives using writethrough cache (**cache=writethrough**)
- a single network

```
# numactl -m 1 --physcpubind=7,15 /usr/libexec/qemu-kvm -M pc -m 5120 -smp 2 -name ra-vm8 -uuid 58dbf832-fbc4-443a-9bbc-94c71f220369 -monitor pty -pidfile /var/run/libvirt/qemu/ra-vm8.pid -boot c -drive file=/dev/cciss/c0d2p4,if=virtio,index=0,boot=on,cache=writethrough -drive file=/dev/cciss/c0d3p4,if=virtio,index=1,cache=writethrough -net nic,macaddr=00:16:3e:10:de:ff,vlan=0,model=virtio -net tap,script=/kvm/qemu-ifup,vlan=0,if-name=qnet7 -serial pty -parallel none -usb -vnc 127.0.0.1:7 -k en-us --mem-path /mnt/libhuget-blfs
```

The following, which is the contents of */kvm/qemu-ifup*, was used to add guest network interfaces to bridge *br0*.

```
#!/bin/sh
/sbin/ifconfig $1 0.0.0.0 up
/usr/sbin/brctl addif br0 $1
```

LAMP:

The "*Deploying the LAMP Stack on Red Hat Enterprise Linux 5*" Reference Architecture paper was used as a reference for the installation of the LAMP stack, however further tweaks were performed to increase performance. First, the small 10 MB database was not used and a customized 4 GB database was created using the tools provided with the DVD distribution. MySQL was configured with the *Innodb* storage engine with *innodb_buffer_pool_size* being assigned 80% of the available memory of the guest. The data disk was on a separate file system from the OS, each using one of the partitions that were created on the host.

For the Apache httpd daemon, KeepAlive was enabled, the number of started servers were adjusted, and logging was disabled.

PHP performance was increased with the installation of Alternative PHP Cache (APC). The steps required to install APC were as follows:

- Install compiler, PHP pear, http and PHP development packages
  ```
  # yum install php-pear php-devel httpd-devel gcc
  ```
- Use pear (PHP Extension and Application Repository) to compile and install APC
  Answer *no* to the "*Use apxs to set compile flags*" question
  ```
  # pear install pecl/apc
  ```
- Tell PHP to include APC
  ```
  # echo extension=apc.so > /etc/php.d/apc.ini
  ```

6 Test Results

Multiple factors can effect scaling. Among these are hardware characteristics, application characteristics and virtualization overhead.

Hardware Characteristics:
The most prominent hardware characteristics relevant to the tests in this paper include limited storage throughput and system architecture. While the disk IO requirements of a single instance of the LAMP stack were not extravagant, this quickly compounds as multiple systems were executed in parallel on the limited set of disks that were available.

The system's architecture includes Intel's Hyper Threading Technology which provides a boost in performance over eight cores, however the performance to the two threads on any hyper threaded core is not expected to be equal to the performance of two non-hyper threaded cores. Linux treats each processing thread as a separate CPU. By assigning a complete core to each 2-vCPUs, the impact of hyper-threading is minimized.

The system was designed with Non-Uniform Memory Architecture (NUMA), which allows quicker access to nearby memory, but conversely, slower accesses to remote memory. This architecture has two NUMA nodes, one for each processor. Assigning a process within a NUMA node allows cache sharing and memory access performance boots.

Application Characteristics:
The type of scaling, up (increased amounts of memory and increased CPU count per guest) or out (multiple instances of similar sized guests), can effect various applications in different ways. The added memory and CPU power will typically help applications that do not contend for a limited resource, where scaling out may provided a multiple of a limited resource. However, scaling out may not be suited for applications requiring a high degree of coordination for the application, which would occur in memory for a scale-up configuration.

Additionally, virtualization can be used to consolidate multiple independent homogenous or heterogeneous workloads onto a single server.

Virtualization Overhead:
As it is not completely running directly on hardware and requires the hypervisor layer, which consumes some processing cycles, some overhead is associated with any virtualization. The amount of virtualization overhead can vary depending on the efficiency of the hypervisor and the use of drivers of varying efficiency.

6.1 Scaling Multiple 2-vCPU Guests

This section presents the results obtained when running multiple 2-vCPU guests (each running an independent LAMP Stack with the DVD Store workload) on a two-socket, quad-core HP ProLiant DL370 G6 host which has 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread. **Figure 5** is a schematic depicting the configuration as multiple 2-vCPU guests are added.

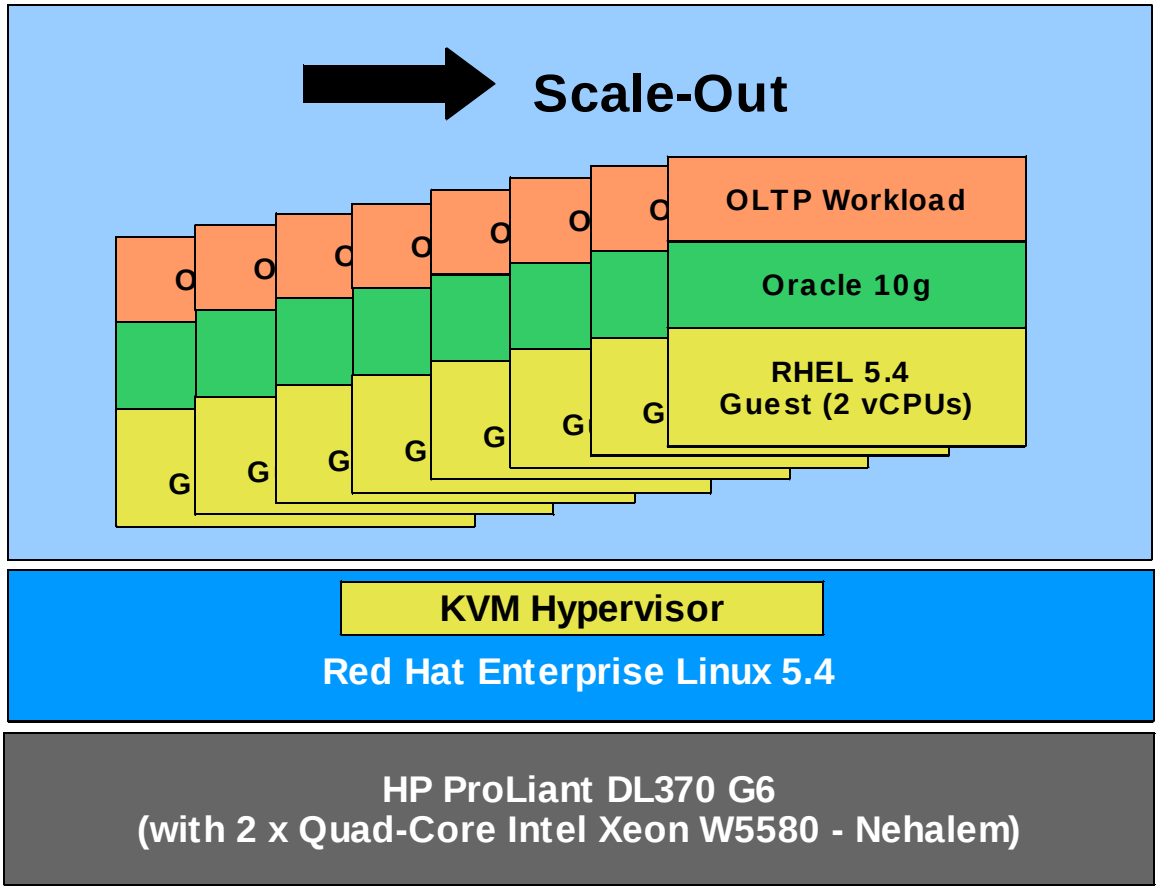


Figure 5: Scaling Multiple 2-vCPU Guests

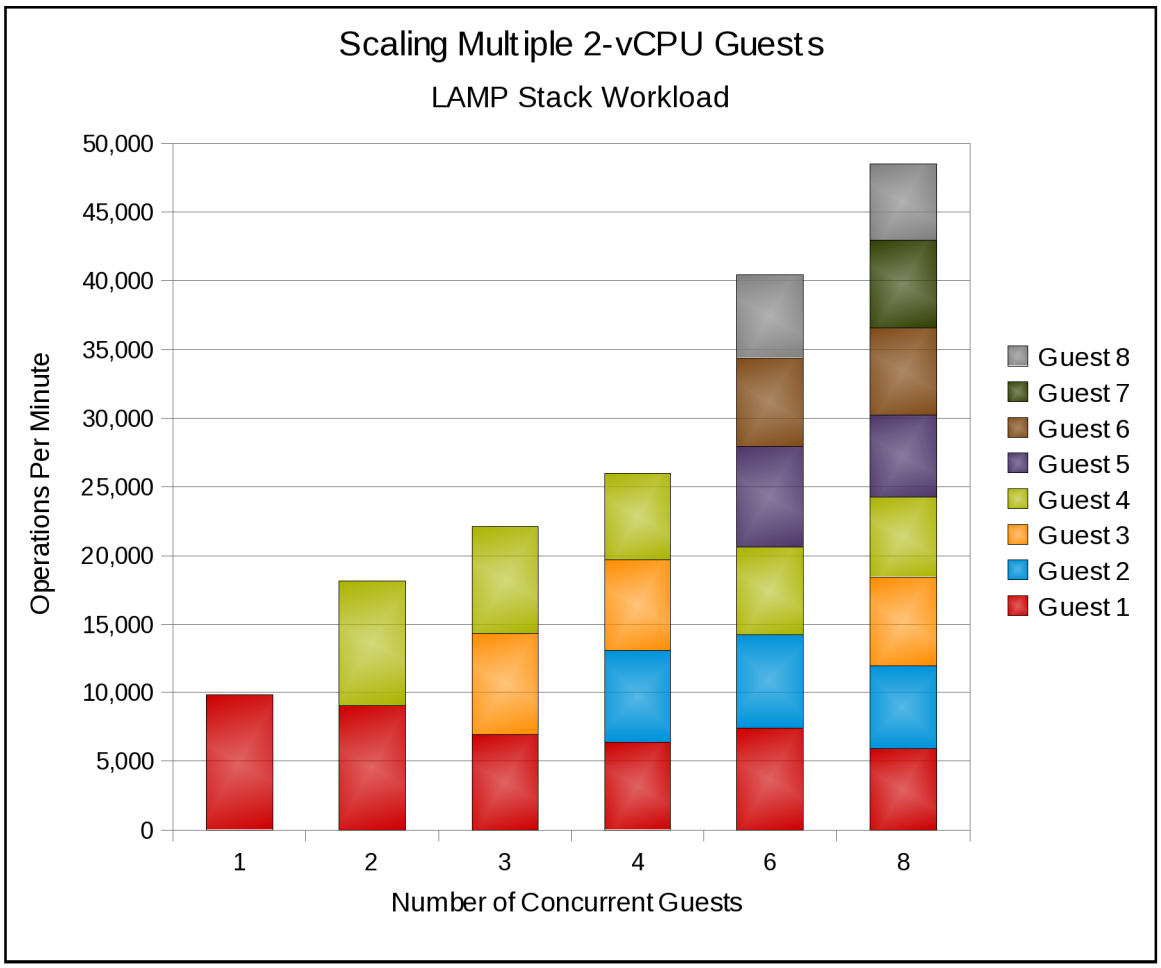


Figure 6: Results of Scaling Multiple 2-vCPU LAMP Guests

6.2 Scaling Multiple 4-vCPU Guests

This section provides the results obtained when running multiple 4-vCPU guests (each running an independent LAMP Stack with the DVD Store workload) on a two-socket, quad-core HP ProLiant DL370 G6 host which has 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread. **Figure 7** is a schematic depicting the configuration as multiple 4-vCPU guests are added.

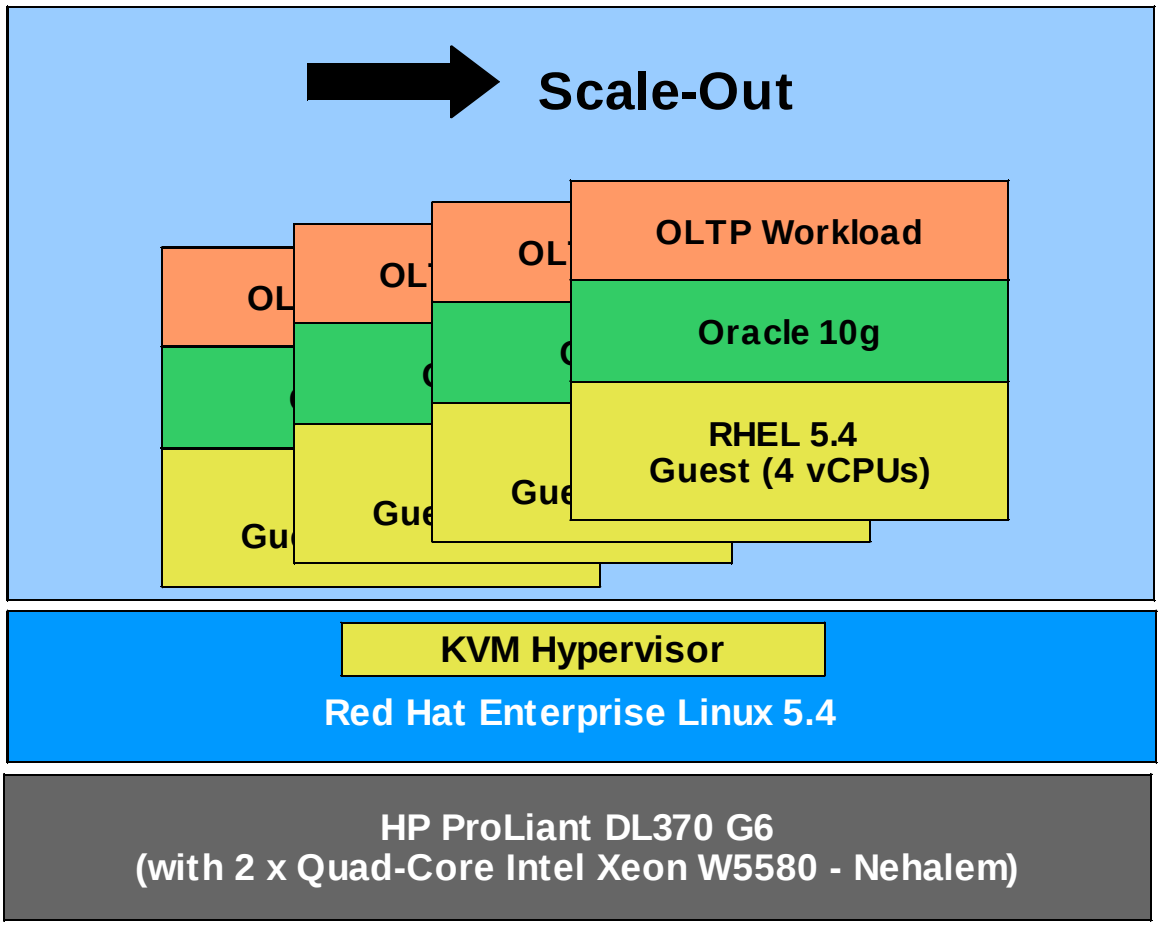


Figure 7: Scaling Multiple 4-vCPU Guests

Figure 8 plots the scalability achieved by incrementally increasing the number of 4-vCPU guests from one to four running the LAMP based DVD Store application. The throughput demonstrates excellent scalability despite increased I/O contention and virtualization overhead.

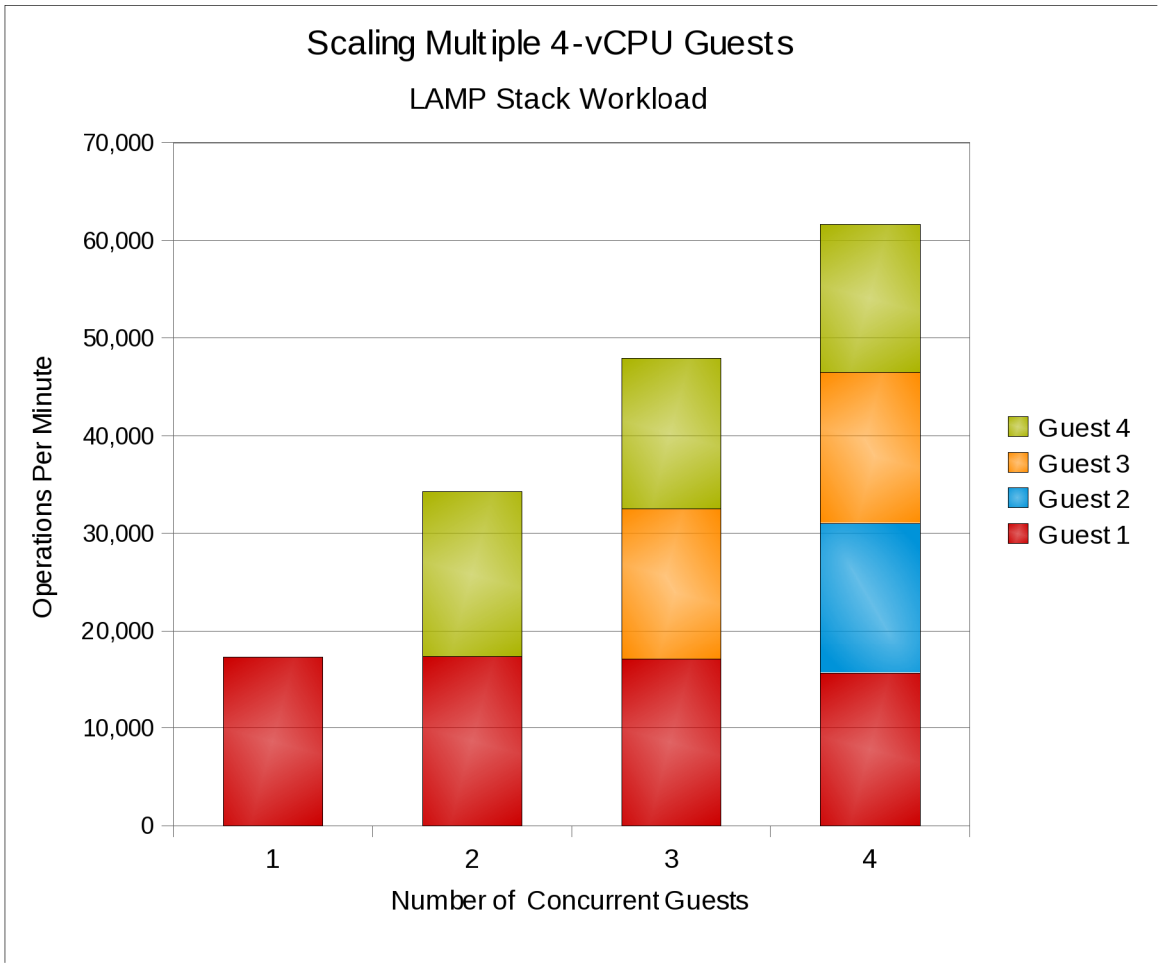


Figure 8: Results of Scaling Multiple 4-vCPU Guests

6.3 Scaling Multiple 8-vCPU Guests

This section supplies the results obtained when running one and two 8-vCPU guests (each running an independent LAMP Stack with the DVD Store workload) on a two-socket, quad-core HP ProLiant DL370 G6 host which has 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread. **Figure 9** is a schematic depicting the configuration as a second 8-vCPU guest is added.

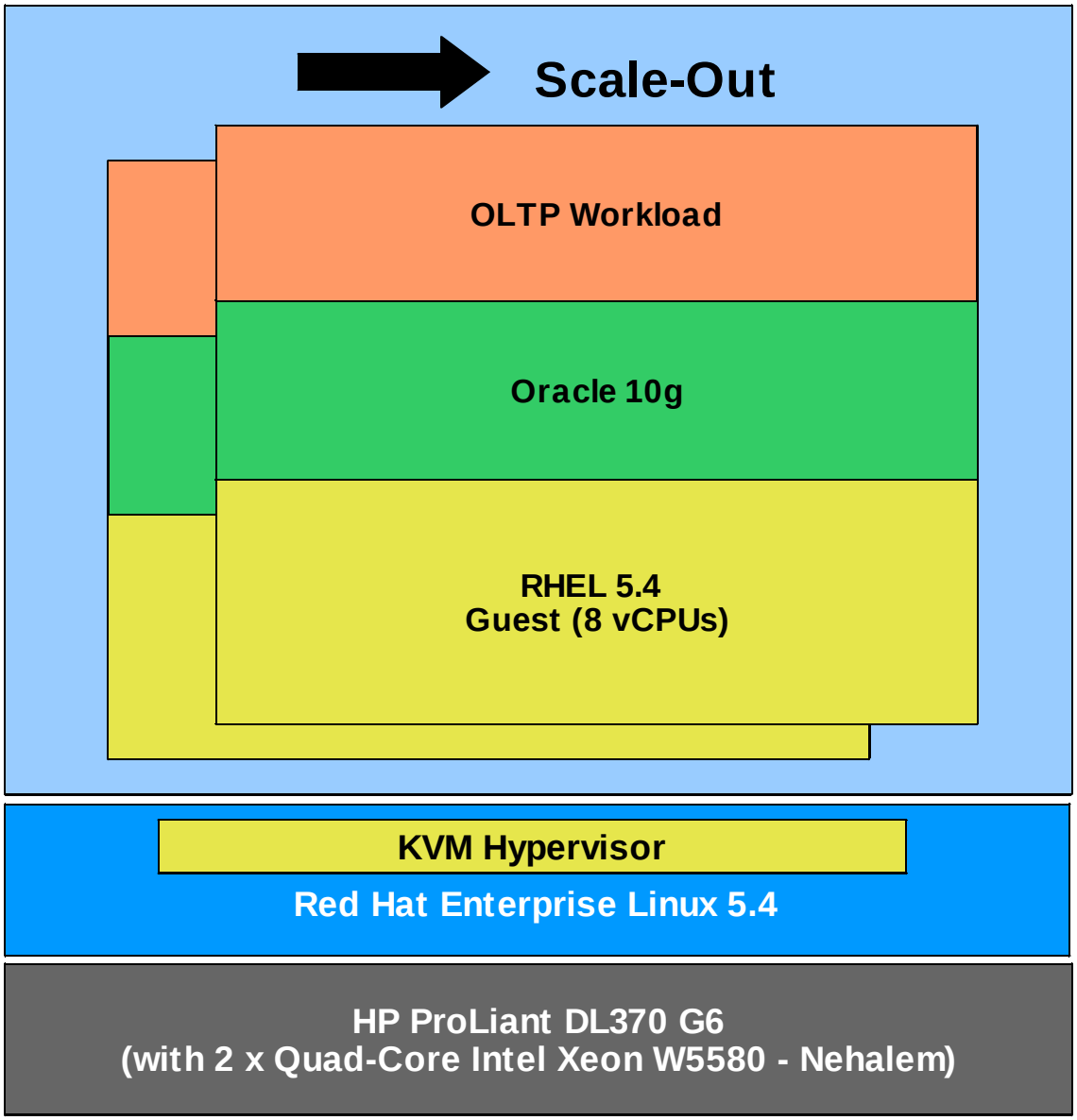


Figure 9: Scaling One & Two 8-vCPU Guests

Figure 10 plots the scalability achieved by adding a second 8-vCPU guest running the LAMP based DVD Store application. The throughput demonstrates excellent (near-linear) scalability.

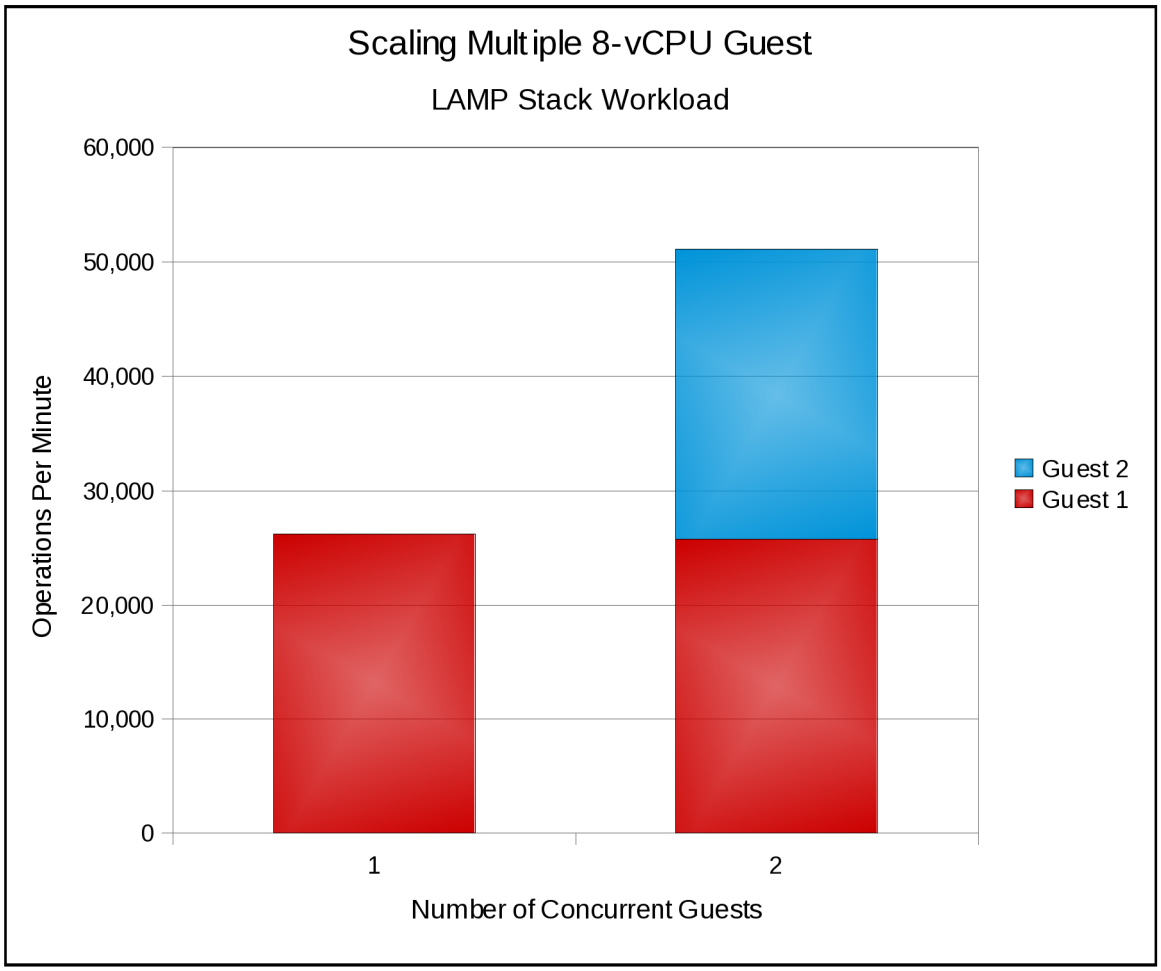


Figure 10: Results of One & Two 8-vCPU Guests

6.4 Scaling-Up by Increasing the Number of vCPUs in a Single Guest

This section demonstrates the results obtained when running the LAMP Stack on a single guest with increased amounts of memory and number of vCPUs. **Figure 11** illustrates the configuration as vCPUs and memory are added.

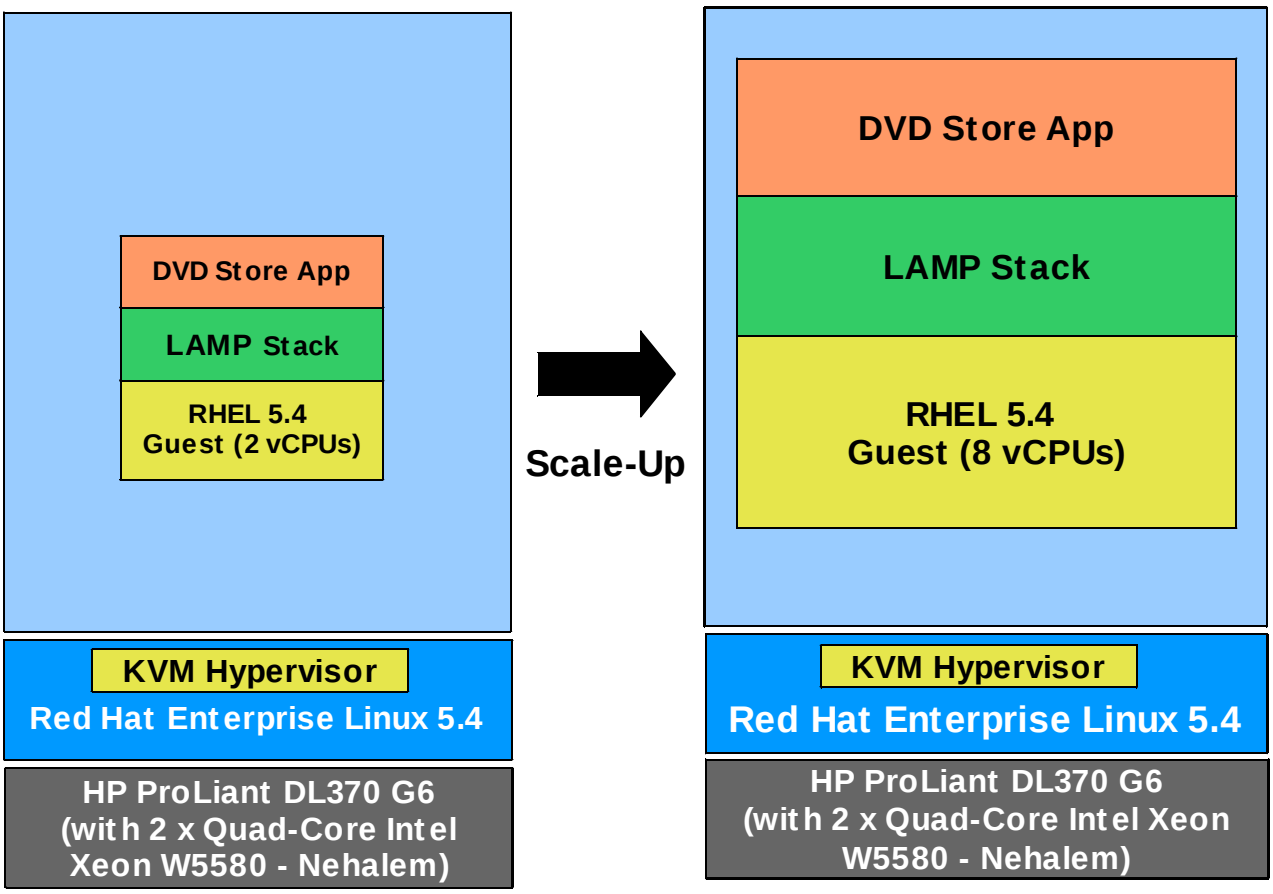


Figure 11: Scale-Up of a RHEV Guest

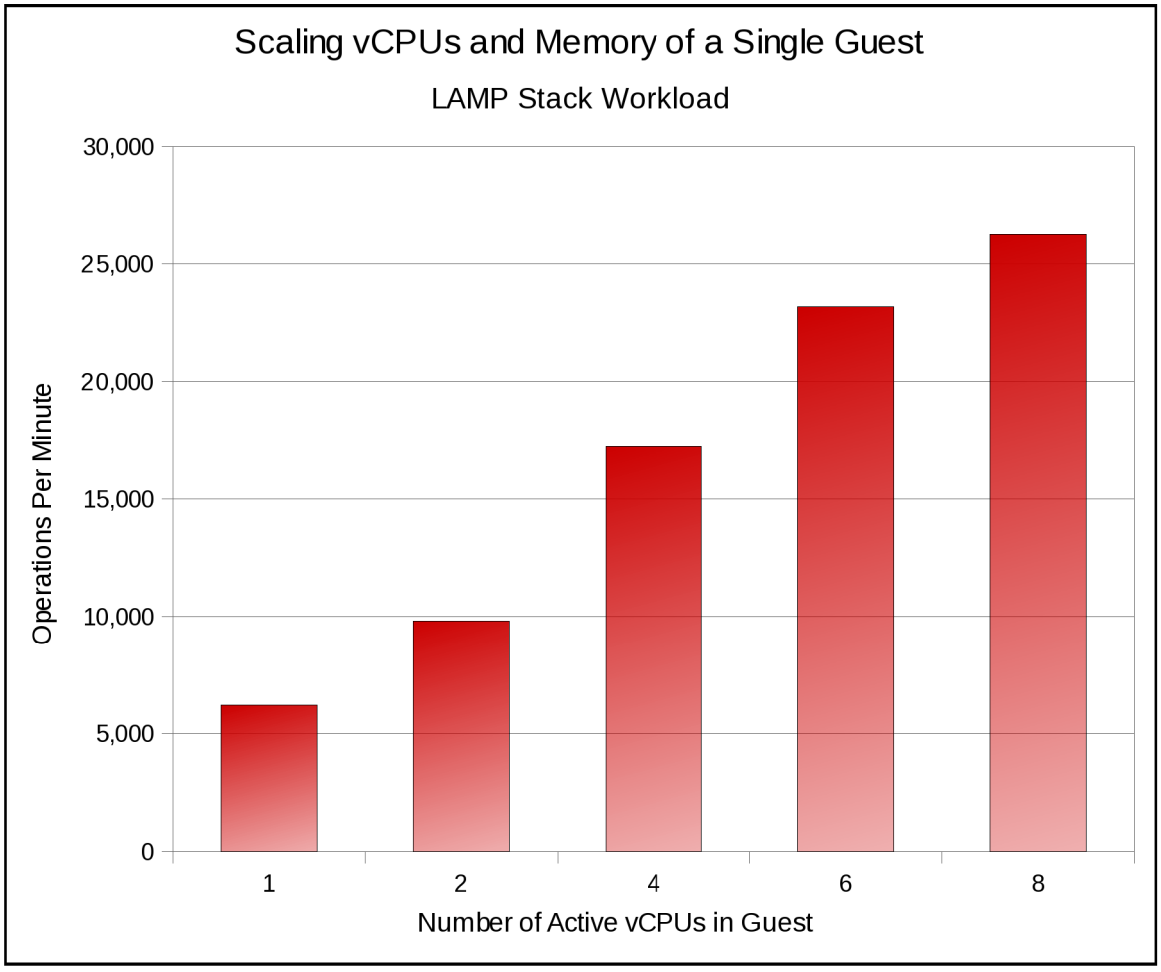


Figure 12: Results of Scaling Memory and Number of vCPUs in a Guest

6.5 Virtualization Efficiency in Consolidation Scenarios

Figure 13 compares the throughput performance of an eight-core (16 hyper-thread) bare-metal configuration to various virtual machine configurations totaling sixteen vCPUs. In the virtual environment, this test was run with 8 x 2-vCPU guests, 4 x 4-vCPU guests, and 2 x 8-vCPU guests. The virtualization results achieving better throughput as compared to bare metal is indicative that LAMP is a better candidate for scaling-out versus scaling-up. The four instances of 4-vCPUs appears to identify a sweet spot. The limited guest memory, limited storage bandwidth, and virtualization overhead affected the eight instances of 2-vCPUs, while the application scaling issues seen in the bare metal result affected the two instances of 8-vCPUS.

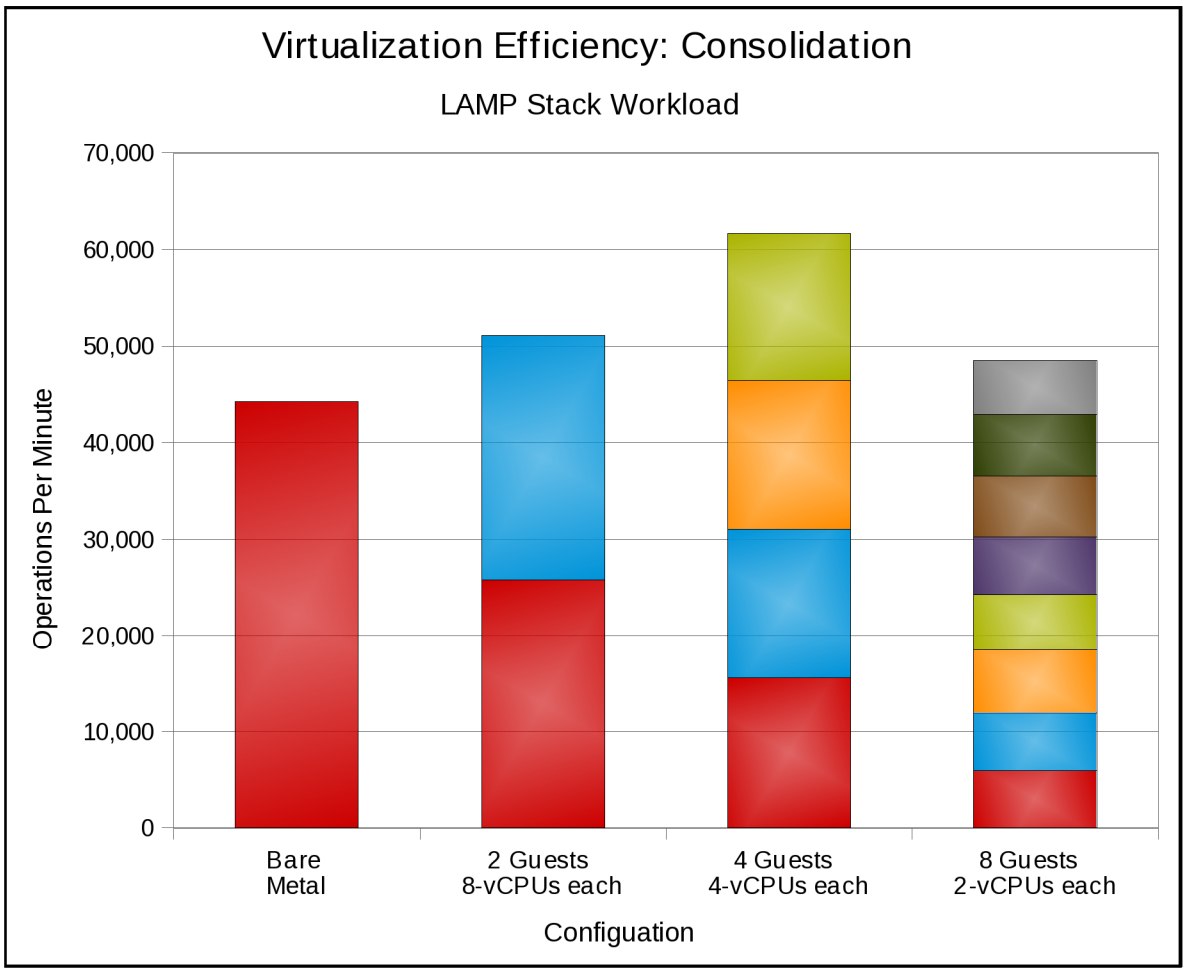


Figure 13: Results of Various LAMP Configurations which Fully Subscribe Available CPUs

7 Conclusions

This paper describes the performance and scaling of the industry-standard LAMP web application stack running in Red Hat Enterprise Linux 5.4 guests on a Red Hat Enterprise Linux 5.4 host with the KVM hypervisor. The host system was deployed on an HP ProLiant DL370 G6 server equipped with 48 GB of RAM and comprising dual sockets each with a 3.2 GHz Intel Xeon W5580 Nehalem processor with support for hyper-threading technology. Total of 8 cores and 16 hyper-threads.

The data presented in this paper clearly establishes that Red Hat Enterprise Linux 5.4 virtual machines using the KVM hypervisor on a HP ProLiant 370 provide an effective production-ready platform for hosting multiple virtualized LAMP web application stacks. The combination of low virtualization overhead and the ability to both scale-up and scale-out contribute to the effectiveness of KVM for LAMP web application stack. The number of actual users and throughput supported in any specific customer situation will, of course, depend on the specifics of the customer application used and the intensity of user activity. However, the results demonstrate that in a heavily virtualized environment, good throughput was retained even as the number and size of guests/virtual-machines was increased up until the physical server was fully subscribed.

8 References

1.Qumranet White paper: *KVM – Kernel-based Virtualization Machine*
2. Red Hat Reference Architecture: *Deploying the LAMP Stack on Red Hat Enterprise Linux 5*