



Red Hat Reference Architecture Series

Scaling Oracle 10g in a Red Hat® Enterprise Virtualization Environment

OLTP Workload
Oracle 10g
Red Hat® Enterprise Linux 5.3 Guest
Red Hat® Enterprise Linux 5.4 (with Integrated KVM Hypervisor)
HP ProLiant DL370 G6 (Intel Xeon W5580 - Nehalem)

Version 1.0
August 2009





Scaling Oracle 10g in a Red Hat® 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

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

© 2009 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

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.....	4
2	Red Hat Enterprise Virtualization (RHEV) - Overview.....	5
	2.1 Red Hat Enterprise Virtualization (RHEV) - Portfolio.....	5
	2.2 Kernel-based Virtualization Machine (KVM).....	7
	2.2.1 Traditional Hypervisor Model.....	8
	2.2.2 Linux as a Hypervisor.....	8
	2.2.3 A Minimal System.....	8
	2.2.4 KVM Summary.....	9
3	Test Configuration.....	10
	3.1 Hardware.....	10
	3.2 Software.....	10
	3.3 SAN.....	10
4	Test Methodology.....	12
	4.1 Workload.....	12
	4.2 Configuration & Workload.....	12
	4.3 Performance Test Plan.....	13
	4.4 Tuning & Optimizations.....	14
5	Test Results.....	15
	5.1 Scaling Multiple 2-vCPU Guests.....	16
	5.2 Scaling Multiple 4-vCPU Guests.....	18
	5.3 Scaling Multiple 8-vCPU Guests.....	20
	5.4 Scaling-Up the Memory and vCPUs in a Single Guest.....	22
	5.5 Consolidated Virtualization Efficiency.....	24
6	Conclusions.....	25
7	References.....	25
	Appendix A - Acronyms.....	25



1 Executive Summary

This paper describes the performance and scaling of Oracle running in Red Hat Enterprise Linux 5.3 guests on a Red Hat Enterprise Linux 5.4 host with the KVM hypervisor. The host 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 was a common Oracle Online Transaction Processing (OLTP) workload.

Scaling Up A Virtual Machine

First, the performance of the Oracle OLTP workload was measured by loading a single VM on the server, and assigning it two, four and eight vCPUs. The performance scales nearly linear 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 comprised of two, four or eight vCPUs, to a total of 16 vCPUs on an 8 core/16 hyper-thread Nehalem server. Results show that the addition of guests scaled well, each producing significant increases in total database transactions.

The data presented in this paper establishes that Red Hat Enterprise Linux 5.3 virtual machines using the KVM hypervisor on Intel Nehalem provide an effective production-ready platform for hosting multiple virtualized Oracle OLTP workloads. The combination of low virtualization overhead and the ability to both scale-up and scale-out contribute to the effectiveness of KVM for Oracle. 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 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 data center. 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 data center 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 (Kernel-based Virtual Machine), 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.

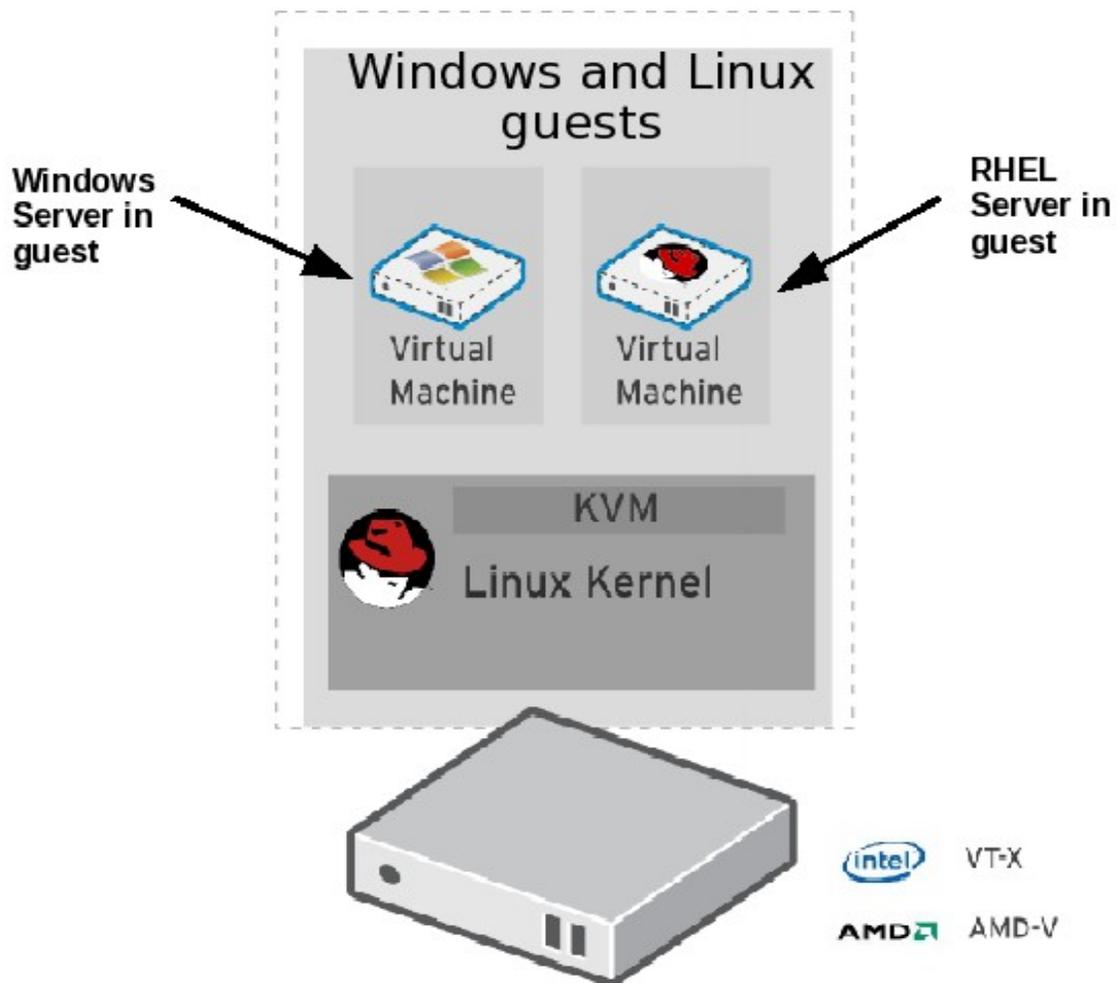


Figure 1: Red Hat Enterprise Virtualization Hypervisor

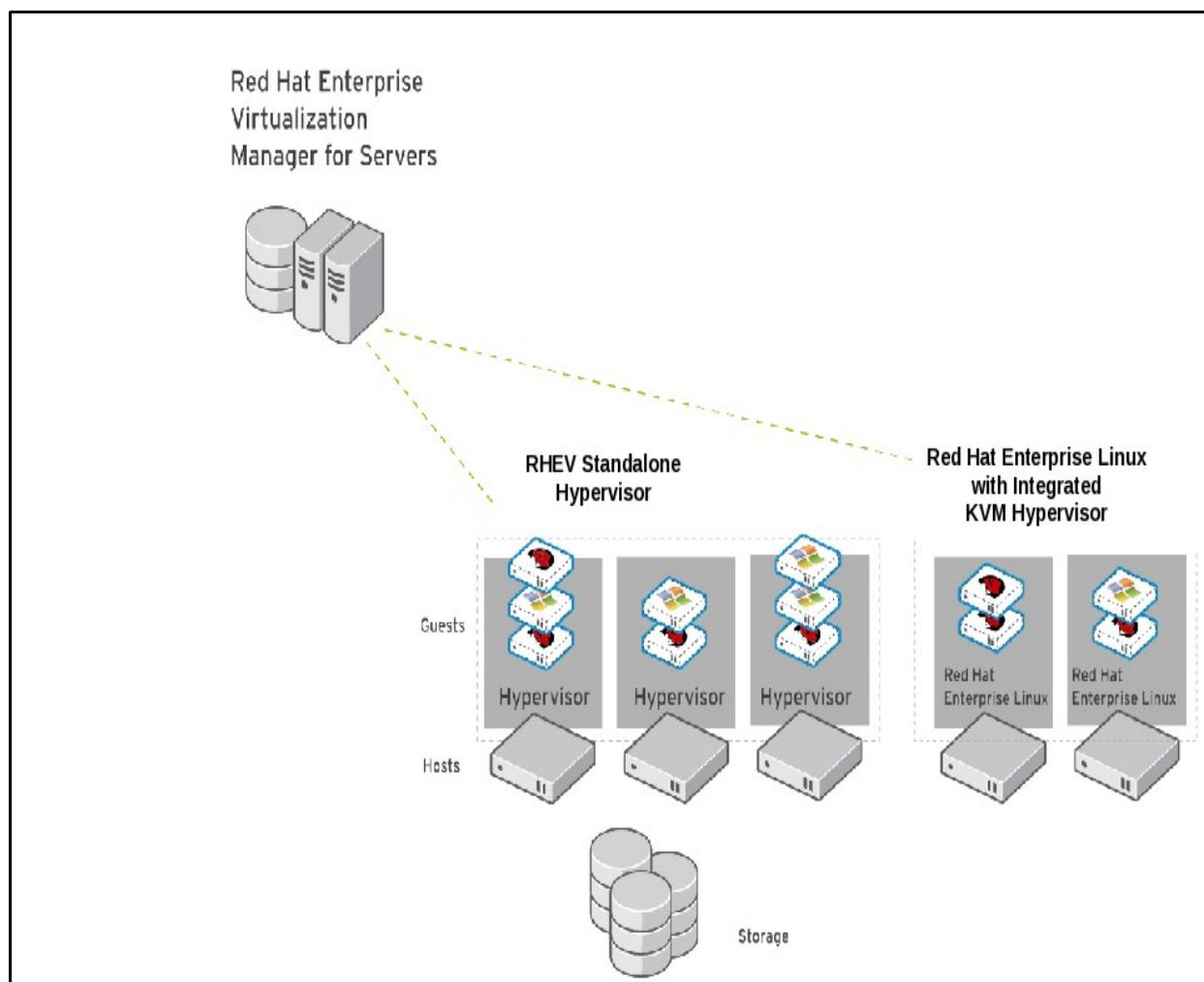


Figure 2: Red Hat Enterprise Virtualization Manager for Servers

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 operating systems) used to require complex hypervisors and previously incurred a performance cost for emulation and translation of I/O and memory requests.

Over the last couple of years, 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 the 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. SR-IOV (Single Root I/O Virtualization) allows specific PCI devices to be split into multiple virtual devices, providing 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 somewhat 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.

KVM turns a Linux kernel into a hypervisor. Red Hat Enterprise Linux 5.4 provides the first commercial-strength implementation of KVM, developed as part of the upstream Linux kernel.

2.2.1 Traditional Hypervisor Model

The traditional hypervisor model consists of a software layer that 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 specific, privileged guest.

Today's hardware, however is becoming increasingly complex. So-called “basic” scheduling operations must 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 Architecture (NUMA) characteristics of a system. While great effort is invested into adding these capabilities to hypervisors, Red Hat has 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 take advantage of all the fine-tuning work that has previously gone (and is presently going) into the kernel, and benefit by it in a virtualized environment. Using 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 thousand 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 (Direct Memory Access) 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 and 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 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 Test Configuration

3.1 Hardware

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

Table 1: Hardware

3.2 Software

Red Hat® Enterprise Linux 5.4	2.6.18-155.el5 kernel
KVM	kvm-83-80.el5
Oracle	v10.2.0.4

Table 2: Software

3.3 SAN

The hypervisor host utilized four one MSA2212fc and three MSA2324fc fibre channel storage arrays for this testing. The MSA2212fc array was used for each of the guest OS disks. The remaining three arrays were used to store workload data and logs. Additional details regarding the Storage Area Network (SAN) hardware are in **Table 3**.



(1) HP StorageWorks MSA2212fc Fibre Channel Storage Array	Storage Controller: Code Version: J200P19 Loader Code Version: 15.010 Memory Controller: Code Version: F300R21 Management Controller Code Version: W420R35 Loader Code Version: 12.013 Expander Controller: Code Version: 2042 CPLD Code Version: 27 Hardware Version: LCA 55
(3) HP StorageWorks MSA2324fc Fibre Channel Storage Array	Storage Controller: Code Version: M100R18 Loader Code Version: 19.006 Memory Controller: Code Version: F300R22 Management Controller Code Version: W440R20 Loader Code Version: 12.015 Expander Controller: Code Version: 1036 CPLD Code Version: 8 Hardware Version: 56
(1) HP StorageWorks 4/16 SAN Switch	Firmware: v5.3.0
(1) HP StorageWorks 8/40 SAN Switch	Firmware: v6.1.0a

Table 3: Storage Area Network

Device-mapper multipathing was used at the host to manage multiple paths to each LUN. Each virtual machine was allocated four 50GB LUNs from the host; one for its operating system, another for Oracle data files, and two for Oracle logging.



4 Test Methodology

4.1 Workload

An Oracle OLTP workload was chosen as it represents a common database implementation exercising both the memory and I/O sub-systems of the virtual machines. Tests were performed on each guest configuration in 15 minute exercises on an eight warehouse database.

4.2 Configuration & Workload

The host is configured with dual Intel W5580 processors, each being a 3.2 GHz quad-core processor supporting Hyper-Threading Technology. While each thread is a CPU in Red Hat Enterprise Linux, two threads share the same processing power of each hyper-threaded core with hardware support. For guests with two virtual CPUs (vCPUs), a single core was allocated for each virtual machine using the `numactl` command. By the same token, two cores from the same processor were allocated for each 4-vCPU guest and a full processor was allocated for each 8-vCPU guest.

Demonstrating the scaling of KVM based virtualization meant several aspects of the workload (user count, SGA size) and guest configuration (vCPU count, memory) were scaled accordingly with the size of the guest. The database was held constant to demonstrate that results were the effect of scaling the guests and not the application. However, per guest factors such as the amount of system memory, the size of the Oracle System Global Area (SGA), and the number of Oracle users were increased with each vCPU. To that extent, an Oracle load of 10 users with a 2GB SGA was allocated per vCPU in each guest. For example, a 4-vCPU guest executed the OLTP workload with 10GB of system memory and 40 Oracle clients using an 8GB SGA.

The host system possessed a total 48 GB of memory. Even distribution of this memory among the vCPUs would allow for 3GB per vCPU, however, 2.5GB was allocated to each vCPU in order to leave memory for the hypervisor as well as guests that may have oversubscribed the processing power of the hypervisor.

Table 4 lists the totals used for each guest configuration.

VCPUs per Guest	Guest Memory	Oracle Users	Oracle SGA
1	2.5 GB	10	2 GB
2	5 GB	20	4 GB
4	10 GB	40	8 GB
6	15 GB	60	12 GB
8	20 GB	80	16 GB

Table 4: Guest/Workload Configurations



4.3 Performance Test Plan

Scale-out:

The scale-out data set highlights the results of scaling a number of concurrent 2-vCPU, 4-vCPU, or 8-vCPU guests executing the OLTP workload.

Scale-up:

The scale-up data set was collected by increased the number of vCPUs and guest memory while repeating the workload on a single guest.

Virtualization Efficiency:

Efficiency is shown by comparing the data when all the physical CPUs are allocated to executing the workload using the bare metal host (no virtualization), eight 2-vCPU guests, four 4-vCPU guests, and two 8-vCPU guests.



4.4 Tuning & Optimizations

The host OS installed was Red Hat Enterprise Linux 5.4 beta, made available via RHN. The primary purpose of this system is to provide a KVM hypervisor for guest virtual machines. Several processes deemed unnecessary for this purpose were disabled using the `chkconfig` command on the host as well as each guest.

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

Security Enhanced Linux (SELinux) was also disabled.

Each guest was started using the `qemu - kvm` command. By doing so, `numactl` could be used to specify CPU and memory locality, and the disk drive cache mechanism could be specified per device. The following example:

- creates a 2-vCPU guest (`-smp 2`)
- binds to two threads in a single core (`--physcpubind=1,9`)
- uses 5 GB of memory (`-m 5120`) from NUMA node 1 (`-m 1`)
- allocates four drives (`-drive`) with cache disabled (`cache=off`)
- starts the network (`-net`)

```
numactl -m 1 --physcpubind=1,9 /usr/libexec/qemu-kvm -M pc -m 5120 -smp 2 \
-name oltp5 -uuid 071940f4-aa42-4a22-8b1e-32a6e5530657 -monitor pty \
-pidfile /var/run/libvirt/qemu/oltp5.pid -boot c \
-drive file=/dev/mapper/oltp_os5,if=virtio,index=0,boot=on,cache=off \
-drive file=/dev/mapper/oltp5_data,if=virtio,index=1,cache=off \
-drive file=/dev/mapper/oltp5_log1,if=virtio,index=2,cache=off \
-drive file=/dev/mapper/oltp5_log2,if=virtio,index=3,cache=off \
-net nic,macaddr=54:52:00:52:12:1d,vlan=0,model=virtio \
-net tap,script=/kvm/qemu-ifup,vlan=0,ifname=qnet6 -serial pty \
-parallel none -usb -vnc 127.0.5.1:0 -k en-us
```

The previous example uses a script (`qemu-ifup`) used to start the network on each guest. The content of that script is simply:

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



5 Test Results

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

Hardware:

The most prominent hardware characteristics relevant to the tests in this paper include limited storage throughput and system architecture. The disk IO requirements of a single database instance may not be extreme but this quickly compounds as multiple systems are executed in parallel against a limited IO bandwidth on the hypervisor.

The system architecture includes hyper-threading technology which provides a boost in performance beyond eight cores. However, the performance of the two threads on any hyper threaded core is not expected to be equal that of two non-hyper threaded cores as Linux treats each processing thread as a separate CPU. By assigning two vCPUs to a complete core, the impact of hyper-threading is minimized.

The system architecture also includes NUMA, which allows faster access to nearby memory, albeit slower access to remote memory. This architecture has two NUMA nodes, one for each processor. Restricting a process to a single NUMA node allows cache sharing and memory access performance boosts.

Application:

The specific scaling, up (increased amounts of memory and CPU) or out (multiple instances of similar sized systems), can effect various applications in different ways. The added memory and CPU power of scaling up will typically help applications that do not contend with a limited resource, where scaling out may provided a multiple of the limited resource. Conversely, scaling out may not be suited for applications requiring a high degree of coordination for the application, which could 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:

As it is not completely running directly on physical hardware and requires the hypervisor layer (which consumes processing cycles), some performance cost is associated with any virtualized environment. The amount of overhead can vary depending on the efficiency of the hypervisor and of the assorted drivers used.



5.1 Scaling Multiple 2-vCPU Guests

This section presents the results obtained when running multiple 2-vCPU guests (each running an independent Oracle OLTP workload) on a two-socket, quad-core HP ProLiant DL370 G6 host having 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread.

Figure 3 is a schematic illustrating the configuration as multiple 2-vCPU guests are added.

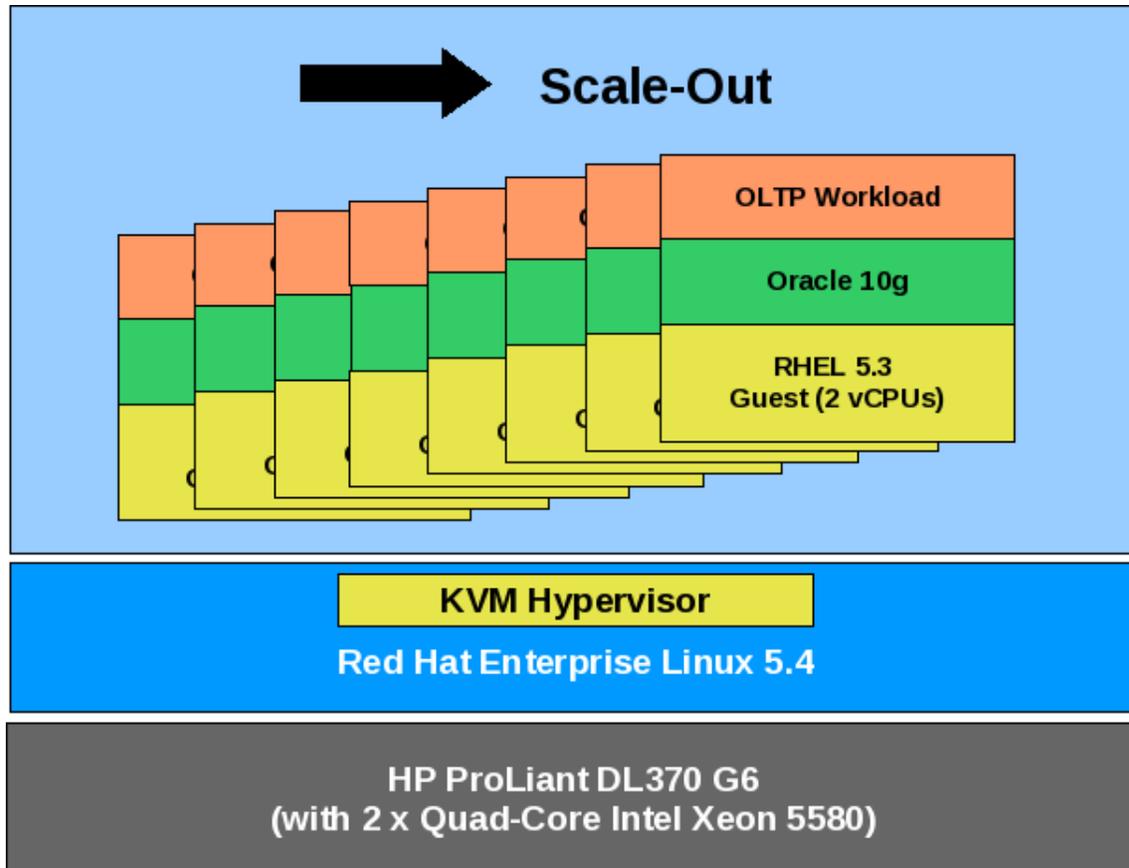


Figure 3: Scaling Multiple 2-vCPU Guests



Figure 4 graphs the scalability achieved by increasing the number of 2-vCPU RHEL guests from one to eight, running independent OLTP workloads. The throughput demonstrates good scaling. As guests are added the throughput per guest decreases slightly due to IO contention and virtualization overhead.

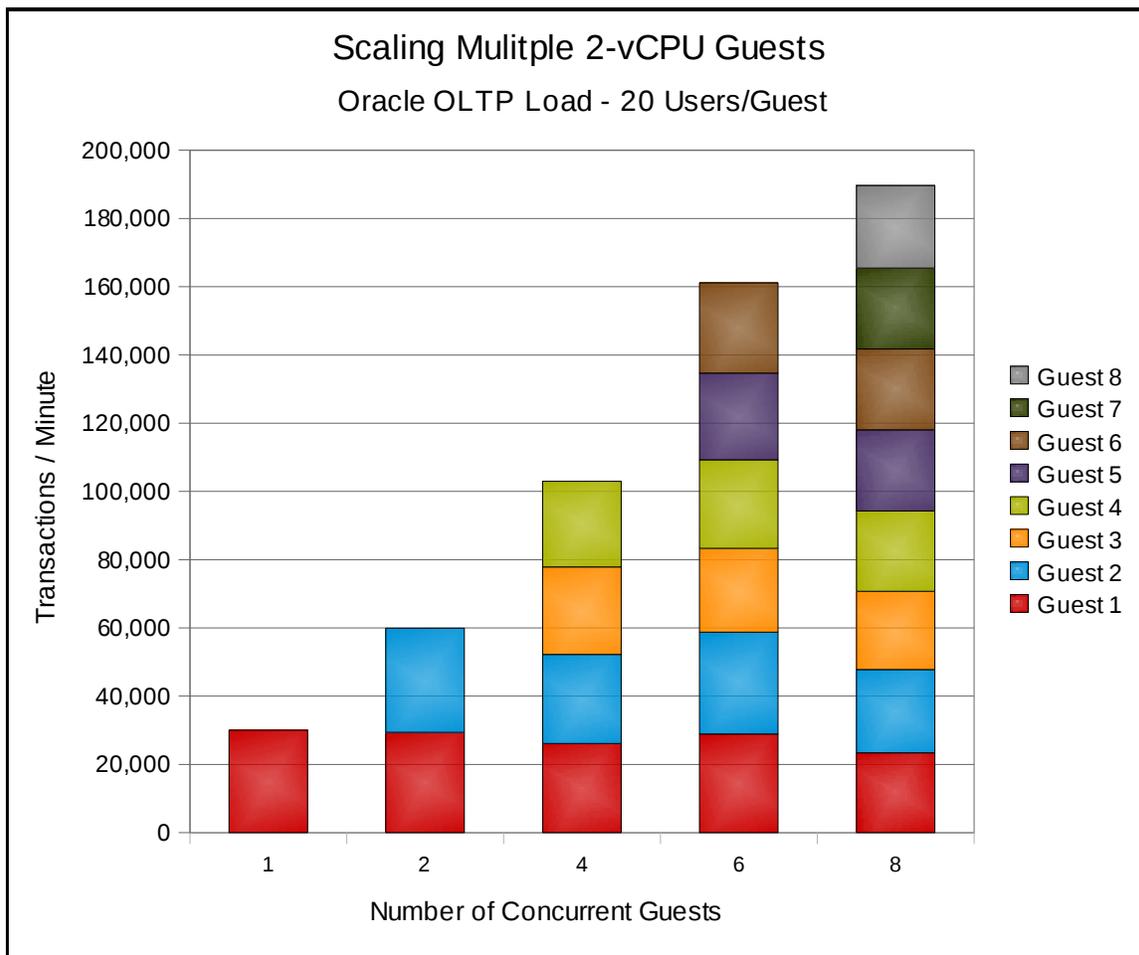


Figure 4: Results of Scaling Multiple 2-vCPU Guests



5.2 Scaling Multiple 4-vCPU Guests

This section presents the results obtained when running multiple 4-vCPU guests (each running an independent Oracle OLTP workload) on a two-socket, quad-core HP ProLiant DL370 G6 host having 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread.

Figure 5 illustrates the configuration as multiple 4-vCPU guests are added.

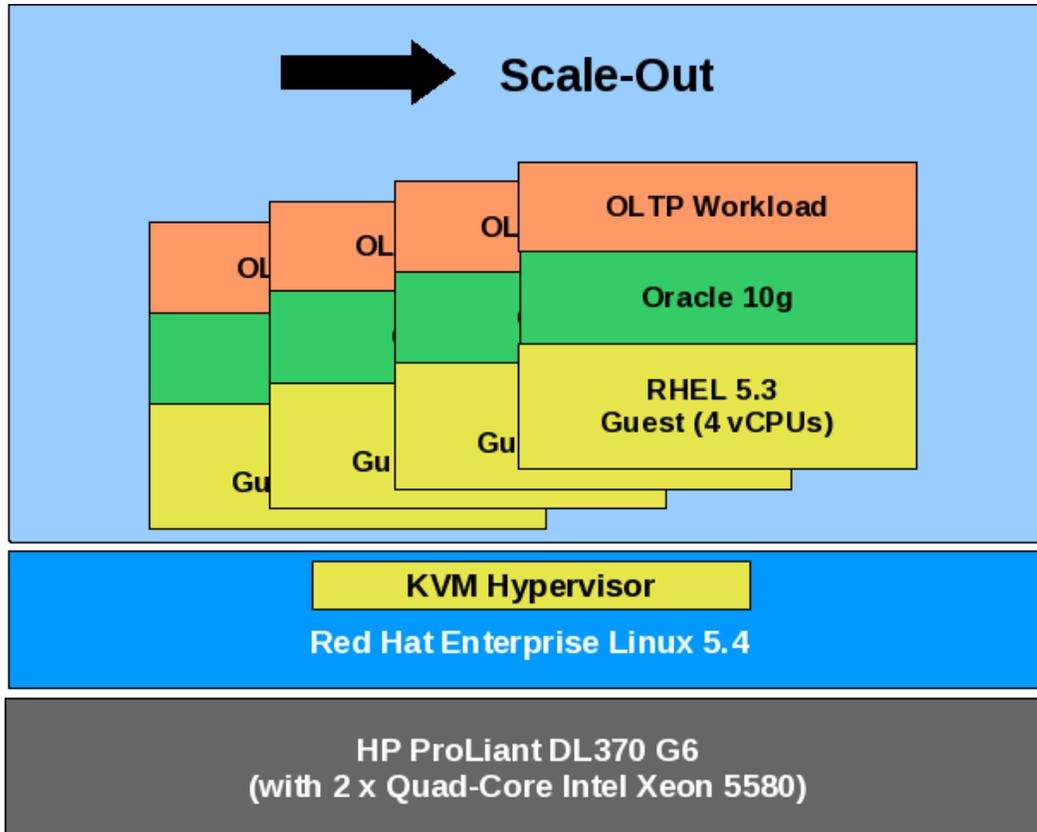


Figure 5: Scaling Multiple 4-vCPU Guests



Figure 6 graphs the scalability achieved by increasing the number of 4-vCPU RHEL guests running the independent OLTP workloads. The throughput demonstrates good scaling. As guests are added the throughput per guest decreases slightly due to IO contention and virtualization overhead.

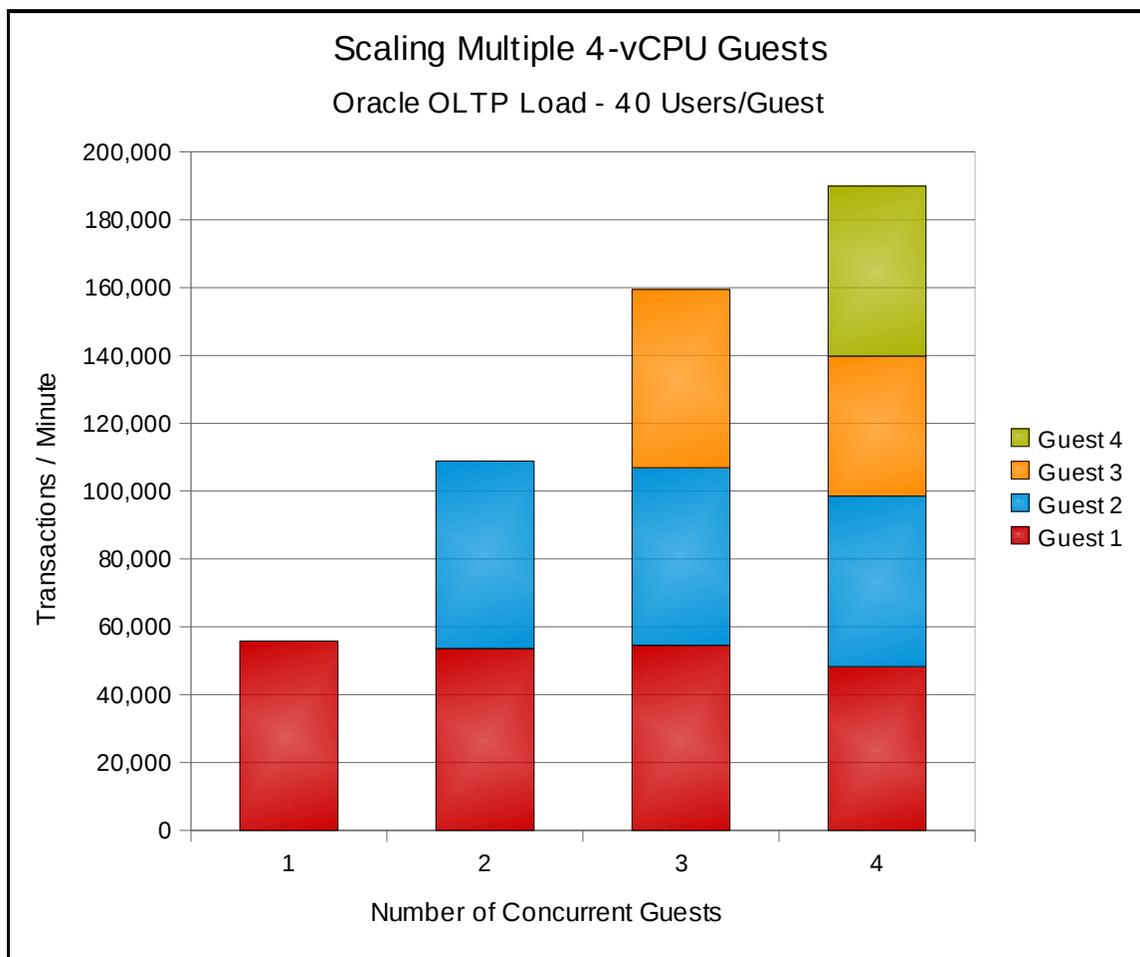


Figure 6: Results of Scaling Multiple 4-vCPU Guests



5.3 Scaling Multiple 8-vCPU Guests

This section presents the results obtained when running one and two 8-vCPU guests (each running an independent Oracle OLTP workload) on a two-socket, quad-core HP ProLiant DL370 G6 host having 8 cores = 16 hyper-threads. Note: 1 vCPU = 1 hyper-thread.

Figure 7 is a schematic illustrating the configuration as a second 8-vCPU guest is added.

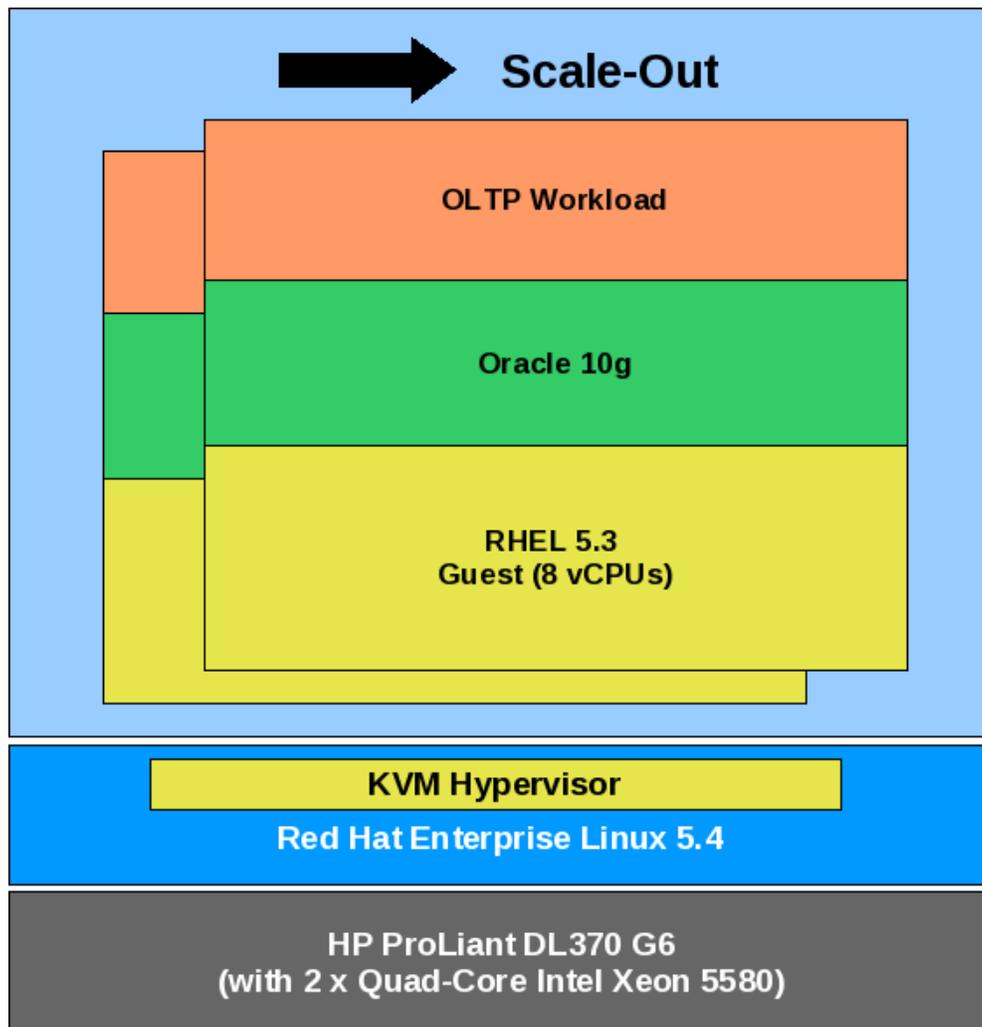


Figure 7: Scaling Multiple 8-vCPU Guests



Figure 8 graphs the scalability achieved by increasing the number of 8-vCPU RHEL guests running independent OLTP workloads. The throughput demonstrates excellent (near-linear) scaling.

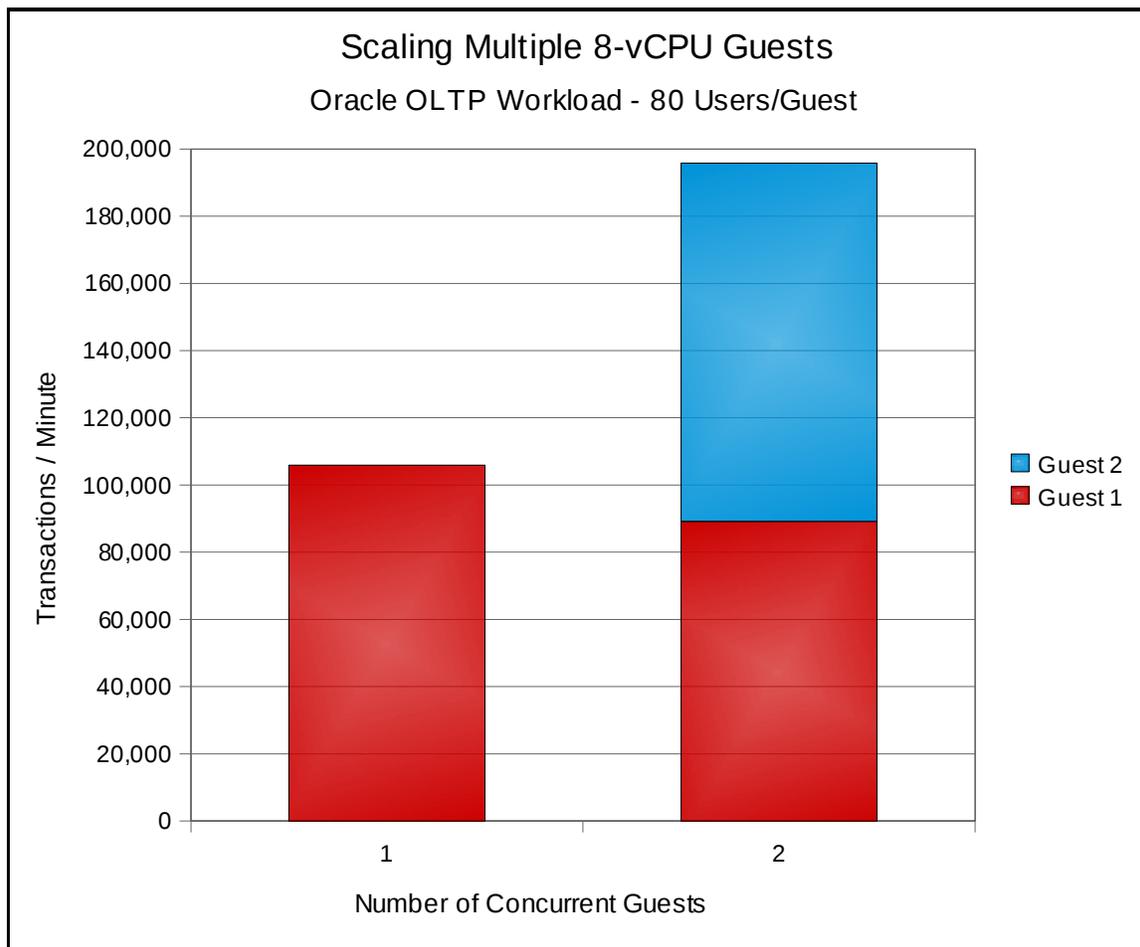


Figure 8: Results of One & Two 8-vCPU Guests



5.4 Scaling-Up the Memory and vCPUs in a Single Guest

This section presents the results obtained when running an Oracle OLTP workload on a single guest with increasing amounts of memory and vCPUs.

Figure 9 illustrates the configuration as vCPUs and memory are added.

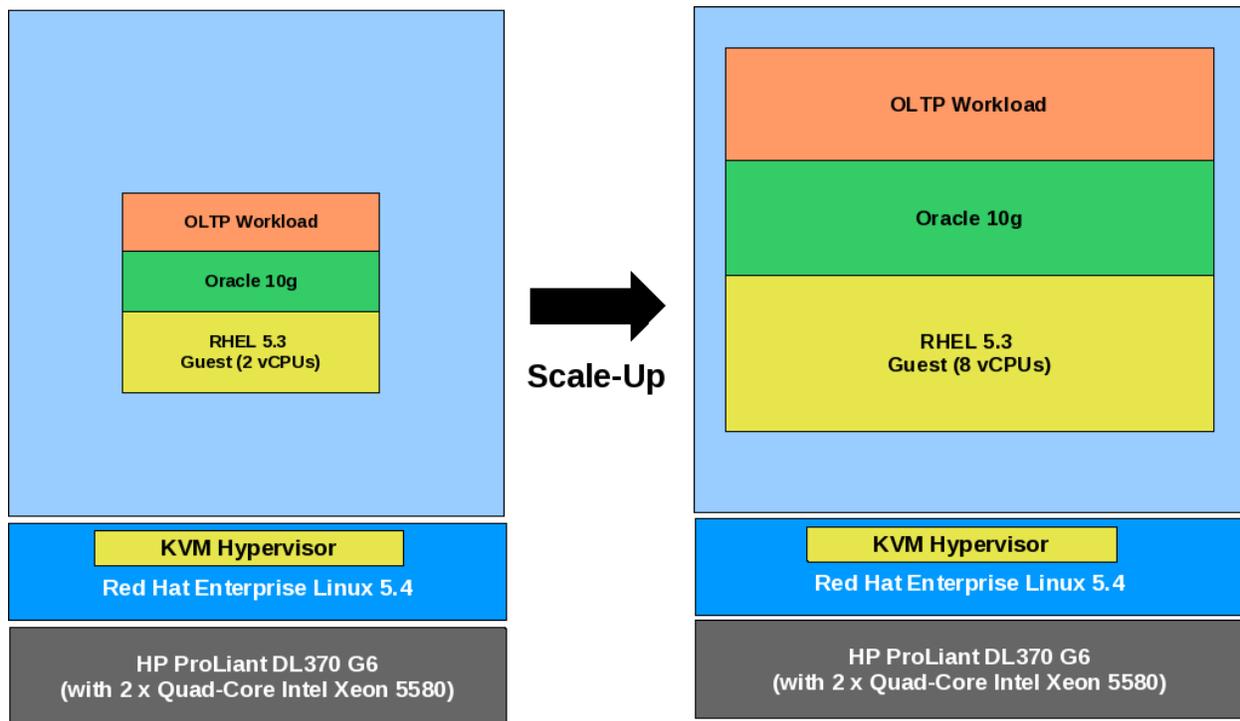


Figure 9: Scaling the Memory and vCPUs in a Single Guest



Figure 10 graphs the results when the OLTP Workload was executed on a guest with 2, 4, 6, and 8 vCPUs with 2.5GB of memory for each vCPU. The throughput demonstrates good scaling. As vCPUs are added, the throughput per vCPU decreases slightly due to IO contention, distributed lock management and virtualization overhead.

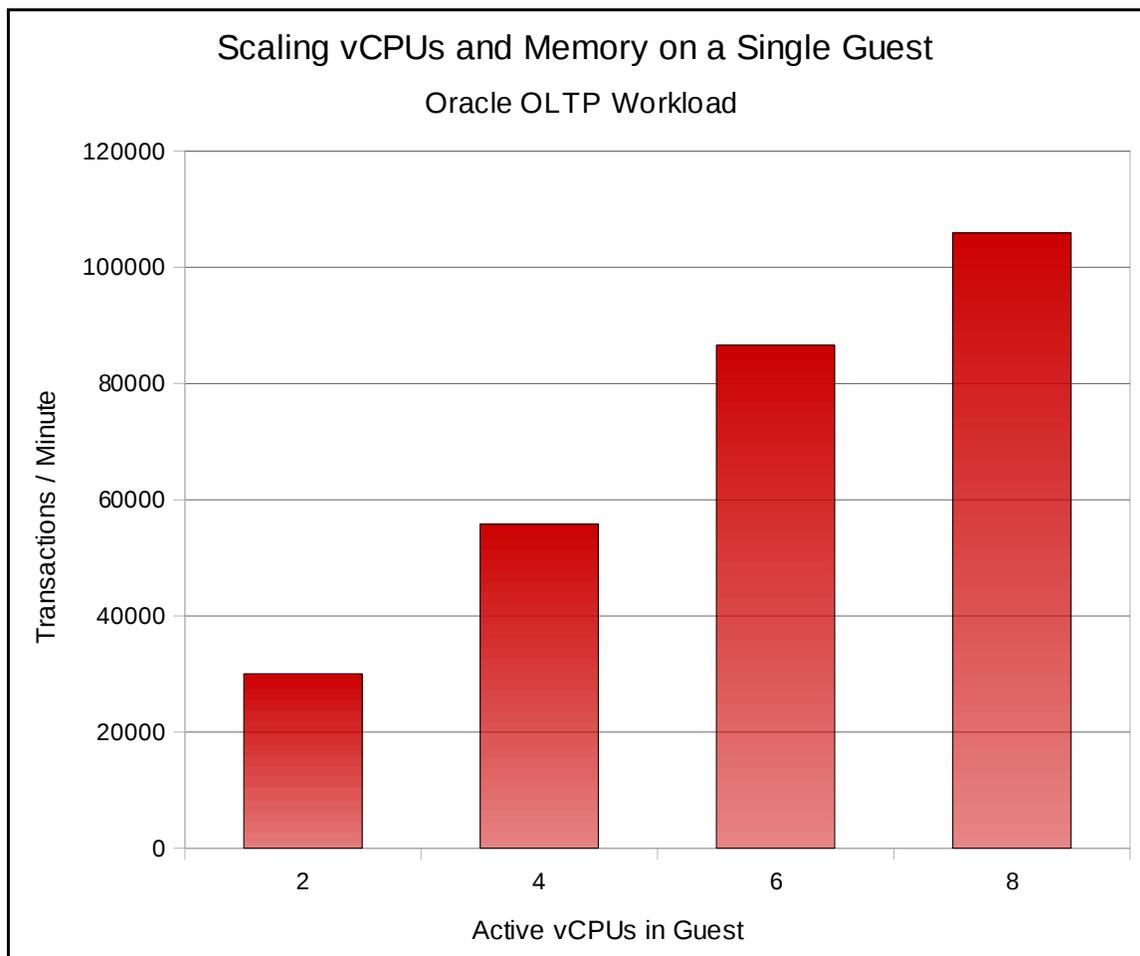


Figure 10: Results of Scaling the Memory and vCPUs in a Single Guest



5.5 Consolidated Virtualization Efficiency

Figure 11 compares the throughput performance of an eight-core (16 hyper-thread) bare-metal configuration to various virtual machine configurations totaling 16 vCPUs. In the virtual environment, this test was run with eight 2-vCPU guests, four 4-vCPU guests, and two 8-vCPU guests.

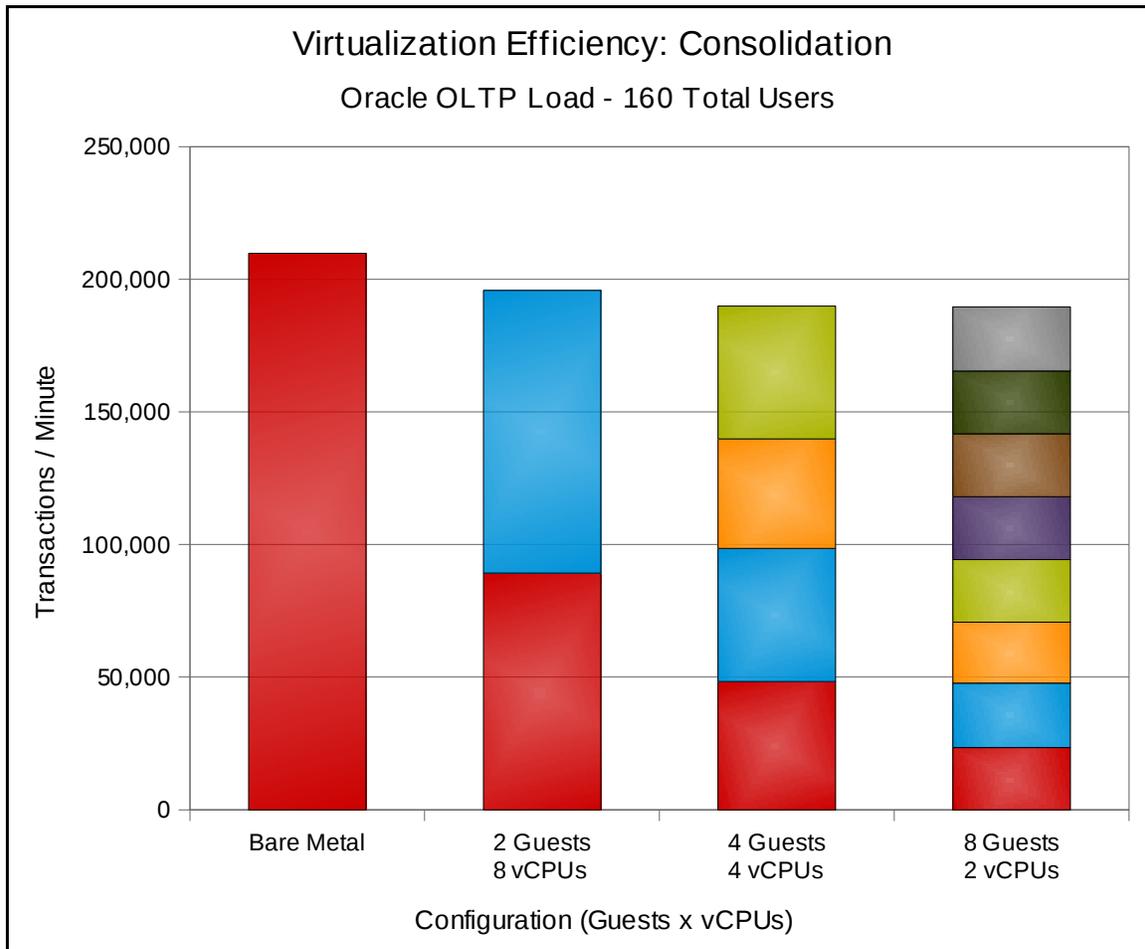


Figure 11: Virtualization Efficiency

The results indicate that comparable performance in virtualized environments can be achieved by scaling the number of guests.

In order to supply sufficient storage to execute the OLTP workload on every guest, each was allocated a single LUN for housing Oracle data files and two for logging. For the non virtualized (bare metal) testing, a second LUN was LVM striped with the first for the data files.



6 Conclusions

This paper describes the performance and scaling of the Oracle OLTP workload running in Red Hat Enterprise Linux 5.3 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 comprised of 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 data presented in this paper clearly establishes that Red Hat Enterprise Linux 5.3 virtual machines using the KVM hypervisor on a HP ProLiant DL370 provide an effective production-ready platform for hosting multiple virtualized Oracle OLTP workloads. The combination of low virtualization overhead and the ability to both scale-up and scale-out contribute to the effectiveness of KVM for Oracle. 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 until the physical server was fully subscribed.

7 References

1. Qumranet White paper: KVM – Kernel-based Virtualization Machine

Appendix A - Acronyms

Acronyms referenced within this document are listed below.

ABI	Application Binary Interface
API	Application Programming Interface
CPU	Central Processing Unit
DMA	Direct Memory Access
EPT	Extended Page Tables
HA	High-Availability
I/O	Input/Output
IOMMU	Input/Output Memory Management Unit
IP	Internet Protocol
KVM	Kernel-based Virtualization Machine
NUMA	Non Uniform Memory Architecture



OLTP	Online Transaction Processing
OS	Operating System
PCI	Personal Computer Interface
RHEL	Red Hat Enterprise Linux
RHEV	Red Hat Enterprise Virtualization
RVI	Rapid Virtualization Indexing
SAN	Storage Area Network
SELinux	Security Enhanced Linux
SGA	System Global Area
SR-IOV	Single Root I/O Virtualization
SVM	Secure Virtual Machine
TPM	Transactions Per Minute
vCPU	Virtual Central Processing Unit
VMM	Virtual Machine Monitor
VT	Virtualization Technology
VT-d	Virtualization Technology for Directed I/O