



Red Hat Performance Briefs

Red Hat Enterprise Linux KVM Hypervisor I/O

**Red Hat and IBM Demonstrate Top Virtualization
Performance with KVM on IBM's x3850X5**

Khoa Huynh, Ph.D. - Linux Technology Center, IBM

Andrew Theurer - Linux Technology Center, IBM

Dor Laor - Senior Engineering Manager, Red Hat

Version 1.0
August 2012





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.

IBM, the IBM logo, ibm.com, ServerAID, System x, X-Architecture, and all other IBM products and services mentioned herein are trademarks of International Business Machines Corporation in the United States and other countries

UNIX is a registered trademark of The Open Group.

Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

© 2012 by Red Hat, Inc and IBM Corp. 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

Send feedback to refarch-feedback@redhat.com

Table of Contents

<u>1 Executive Summary.....</u>	<u>1</u>
<u>2 Introduction.....</u>	<u>2</u>
<u>2.1 The KVM hypervisor.....</u>	<u>2</u>
<u>2.2 Red Hat® Enterprise Linux® 6 Virtualization.....</u>	<u>3</u>
<u>2.3 IBM® System x® Servers.....</u>	<u>3</u>
<u>2.4 Motivation.....</u>	<u>4</u>
<u>3 Test Setup.....</u>	<u>5</u>
<u>3.1 Test Hardware.....</u>	<u>5</u>
<u>3.2 Workload.....</u>	<u>5</u>
<u>3.3 KVM Configuration.....</u>	<u>7</u>
<u>4 Results.....</u>	<u>8</u>
<u>4.1 Single VM.....</u>	<u>8</u>
<u>4.2 Multiple VMs.....</u>	<u>11</u>
<u>5 Summary.....</u>	<u>15</u>
<u>Appendix A: References.....</u>	<u>16</u>

1 Executive Summary

The Kernel-based Virtual Machine (KVM) hypervisor has earned a reputation as the highest performing hypervisor in virtualization benchmarks, holding the top seven results in SPECvirt_sc2010 [1] and recently achieving a leadership spot among x86-virtualized results on SAP's 2-tier SD benchmark [2]. One of the key ingredients to this success is the KVM hypervisor's ability to handle the high I/O rates required by enterprise workloads, such as databases, ERP (Enterprise Resource Planning) systems, and low-latency financial trading applications, that are running in virtual machines.

This paper describes the test environment which is comprised of an IBM® System x3850 X5 host server with QLogic® QLE 256x Host Bus Adapters (HBAs), Red Hat® Enterprise Linux® 6.3 hypervisor, and Red Hat Enterprise Linux 6.3 guests. The test results validate that a single KVM guest can handle more than 800,000 I/Os per second (IOPS) and four KVM guests running on the same host can support an aggregate I/O rate of more than 1.4 million IOPS – hitting the “bare-metal” limit of the test storage setup. These results show that KVM can achieve the highest storage I/O performance in a virtualized environment and is ready for the most demanding enterprise workloads.



2 Introduction

2.1 The KVM hypervisor

A hypervisor is a specialized operating system that runs virtual machines instead of physical applications. The Kernel-based Virtual Machine (KVM) project represents the next generation in open-source virtualization. KVM is fully integrated into the Linux operating system both as a host and a guest. Unlike other hypervisors, KVM does not make any distinction between running in either host or hypervisor mode. This duality in design has helped KVM to rapidly mature into a stable, high-performing hypervisor, positioned to outperform any other hypervisor available on the market today.

The first design principle includes the following:

- Leverage all hardware-assisted virtualization capabilities provided by Intel[®] Virtualization Technology (VT) and AMD[®] Secure Virtual Machine (SVM).
- Feature the latest hardware virtualization extensions, including:
 - Hardware nested paging (EPT/NPT)
 - Pause loop filtering
 - I/O off-load features, such as secure PCI pass-through using Intel VT-D or AMD I/O Memory Management Unit (IOMMU)
- Exploit hardware capabilities while keeping the KVM virtualization overhead to the absolute minimum

The second design principle includes the following:

- Leverage the Linux operating system
- Fulfill the many components required by a hypervisor, such as memory management, scheduler, I/O stack, and device drivers by reusing optimized, off-the-shelf Linux implementations

The Linux kernel, with its 20 years of development, is the industry leader in terms of performance and availability. The Linux process scheduler, for example, provides completely fair scheduling (CFS) that is optimized to manage complex workloads and NUMA systems, while offering low latency, high performance determinism, and fine-grained Service Level Agreement (SLA) for applications. By placing the KVM hypervisor directly into the Linux kernel, all of these services and advantages have a direct impact on the hypervisor performance.

2.2 Red Hat® Enterprise Linux® 6 Virtualization

Red Hat's unique approach to virtualization is easy to adopt as it is delivered as an integral part of the Red Hat Enterprise Linux platform. Using KVM technology, Red Hat's virtualization capabilities are integrated into Red Hat Enterprise Linux and leverage the latest in hardware virtualization that Intel and AMD processor platforms can provide. The modular design of Red Hat Enterprise Linux allows customers to choose when and where to use virtualization. For additional flexibility, customers can deploy both Red Hat Enterprise Linux and Microsoft® Windows® as fully supported guests within a Red Hat Enterprise Linux virtualized environment. Red Hat Enterprise Linux also supports multiple virtualization use cases, from hardware abstraction for existing software stacks and data center consolidation to virtualized clusters and private clouds.

Beyond core virtualization, Red Hat Enterprise Linux offers leading support for advanced virtualized I/O capabilities through Single Root I/O Virtualization (SR-IOV) and N-Port ID Virtualization (NPV) standards. The *libvirt* toolkit – a standard virtualization management infrastructure – was developed by Red Hat and adopted by other operating systems. The *libvirt* toolkit provides a flexible interface for defining, managing, and monitoring virtual machines.

The Red Hat Enterprise Linux 6.3 release supports up to 160 virtual CPUs (vCPUs) per virtual machine, allowing even the largest workloads to be virtualized.

2.3 IBM® System x® Servers

IBM System x servers support Microsoft Windows, Linux, and hypervisors. System x servers are intelligent systems, designed to reduce costs and complexity for enterprise workloads. With the introduction of eX5 – IBM's 5th-generation industry-leading enterprise X-Architecture® servers – IBM engineers have redefined x86 servers by expanding their capabilities. A member of the eX5 server family, the x3850 X5 is a scalable, 4-socket, 4U, rack-optimized enterprise server that delivers the following benefits to enterprise customers:

- High memory capacity (up to 3TB, 3 times the memory capacity of other 4-socket x86 servers, using the industry-unique IBM MAX5 memory expansion unit)
- Processor scalability up to 8 sockets (up to 80 processor cores) by connecting two 4-socket x3850 X5 systems together and doubling all system resources (including up to 6TB of memory, using two MAX5 memory expansion units)
- The broadest range of network and storage support in the industry for ultimate flexibility and choice
- Support for IBM eXFlash solid-state storage technology for extreme storage I/O performance
- Integrated Emulex 10-GbE Virtual Fabric Adapter with capability for upgrades to Fiber Channel over Ethernet (FcoE)



- 5 chipset design enhancements, built on the latest X-Architecture blueprint
- Balanced systems for virtualization, database, and enterprise workloads
- Workload-optimized systems with customizable configurations for target workloads
- Greater performance and utilization at a lower total cost
- Mainframe-inspired reliability
- Simplified power and systems management with an energy-smart design and remote access

Figure 1 shows the front exterior of the IBM System x3850 X5 server.



Figure 1. IBM System x3850 X5 Server

2.4 Motivation

IBM and Red Hat customers drive enterprise workloads, such as databases, ERP systems, and low-latency financial trading applications. In the past, these workloads were seldom virtualized in production due to scaling and time-sensitive barriers, so they were unable to exploit the many benefits of virtualization, such as hardware abstraction, live migration, dynamic resource allocation, and more. Proving that KVM is able to sustain high I/O rates is very critical in enabling the migration of these workloads into the virtualized environment.

3 Test Setup

3.1 Test Hardware

To demonstrate how KVM can handle extremely high I/O rates, it was necessary to set up a storage back-end that was capable of delivering at least one million I/Os per second (IOPS). For a diagram of the test environment, refer to **Figure 2**.

The KVM host server was an IBM System x3850 X5 with four Intel Xeon® E7-4870 processors (sockets) and 256 GB of memory. Each E7-4870 processor had 10 cores running at 2.40 GHz. The x3850 X5 server had seven available PCI slots, each of which was fitted with QLogic® QLE 256x Host Bus Adapters (HBAs). Each of these adapters had two ports, each supporting full-duplex, 8-Gigabit-per-second data links.

Each QLogic HBA in the KVM host was directly connected to a unique fiber-channel SCSI target server. For more information about the fiber-channel SCSI target server, please see <http://linux-iscsi.org>. Four 15-GB RAM disks were configured in every SCSI target server, so that these RAM disks would appear as Logical Unit Numbers (LUNs) at the SCSI host (KVM host server). As a result, from the KVM host's perspective, the storage back-end had a total of 14 (7 PCI slots x 2 ports) PCI devices and 56 (14 PCI devices x 4 RAM disks) storage LUNs.

3.2 Workload

The Flexible I/O (FIO) benchmark (<http://linux.die.net/man/1/fio>) is used to generate disk I/O workloads and measure the resulting I/O rates, throughput, and latency data. This workload has the following FIO parameter settings:

- Direct I/Os
- Asynchronous I/Os (engine = libaio)
- Random reads and writes (50% reads, 50% writes)
- I/O request sizes = 512 bytes, 1KB, 2KB, 4KB, 8KB
- One job per storage LUN
- Queue depth = 32

Both random reads and writes were included in the workload for two reasons. First, a workload with both random reads and writes would be more realistic and similar to actual enterprise workloads, such as database applications. Secondly, having both reads and writes in the workload allowed the ability to fully exploit the full-duplex data links supported by the QLogic HBAs. A range of I/O request sizes, from 512 bytes to 8KB, was also considered as these are the typical I/O sizes used in many real-world applications.



The FIO workload was first run directly on the KVM host server to determine the maximum I/O rates that this test setup could support. These “bare-metal” results indicated that our storage setup could deliver up to 200,000 IOPS per SCSI target server, or a total of 1.4 million IOPS with all 7 SCSI target servers, using 8KB requests. This was determined to be sufficient for the KVM storage performance testing.

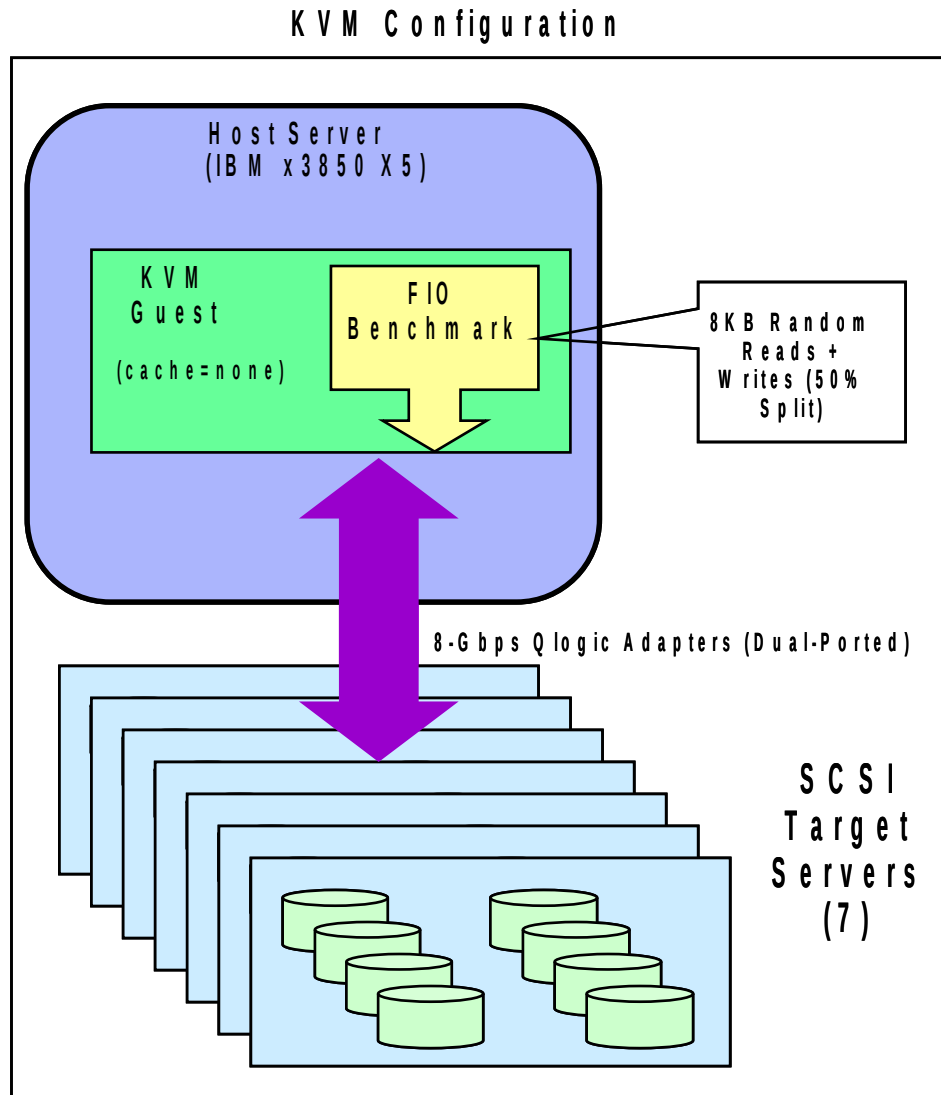


Figure 2. The test setup

3.3 KVM Configuration

In order to achieve the best possible I/O rates, the QLogic PCI physical functions were passed directly from the host to the KVM guest(s) using KVM's *PCI pass-through* capability. More specifically, the following configuration changes were made to the KVM host server:

1. Added `intel_iommu=on iommu=pt` to the kernel boot command line in `/boot/grub/grub.conf`
2. Rebooted host server
3. For each PCI device (port) that was needed to be passed directly to the KVM guest, the following was specified in the XML definition file for that guest:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x86' slot='0x00' function='0x0' />
  </source>
  <alias name='hostdev0' />
</hostdev>
```

In this example, *bus* represents the PCI bus address of the QLogic HBA and *function* represents the port on that HBA through which the storage LUNs were passed directly to the guest.

After the guest was defined from the XML libvirt file and finished booting up, the guest could see all the storage LUNs that were passed to it from the host.

Red Hat Enterprise Linux version 6.3 was used on all guests and on the host. By running the same FIO workload on the KVM host (“bare metal”) and guests, it would be able to demonstrate how well KVM can handle I/Os compared to bare metal, which represented the maximum performance that one could get out of the test setup and was therefore the control for this experiment.



4 Results

4.1 Single VM

In the single-guest configuration, the goal was to determine the maximum I/O rates that could be achieved with a single KVM guest. A very large guest was configured with 40 virtual CPUs to test this goal. Because our FIO workload was not very memory-intensive, 16GB of memory was configured for the guest.

The single-guest configuration details included:

- Host Server: IBM System x3850 X5
 - 4 Intel Xeon E7-4870 processors (40 cores @ 2.40 GHz), 256 GB memory (total)
 - Red Hat Enterprise Linux 6.3
- Storage:
 - 4 QLogic QLE 256x (8 Gbps, dual ported) connected to 4 SCSI target servers
 - 8 PCI devices and 32 LUNs (each SCSI target server was backed by RAM disks and provided 2 PCI devices and 8 LUNs)
- KVM Guest (Virtual Machine):
 - 40 vCPUs, 16 GB memory
 - 8 PCI devices, 32 LUNs
 - Red Hat Enterprise Linux 6.3
- FIO Workload
 - Random reads and writes (50% reads, 50% writes)
 - 1 job per LUN
 - Direct I/Os
 - Engine = libaio
 - Queue depth = 32

A single KVM guest supports a maximum of 8 PCI devices, so during this test, four QLogic HBAs with 8 PCI devices and 32 storage LUNs were passed to the guest. The guest's large footprint (its virtual CPUs and memory) actually spanned two of the four NUMA nodes on the host server, so the guest had its own NUMA characteristics, which were specified in its XML libvirt definition file.

In addition, caching efficiency was leveraged in the host processors by binding FIO workload jobs to specific virtual CPUs in the guest and the interrupt-coalescing capability for the QLogic HBAs was used for optimal performance.

Figure 3 shows the number of I/Os per second (IOPS) and the average latencies for random reads and writes across several I/O request sizes. Using a single guest, KVM was able to achieve about **800,000 IOPS** for 8KB random I/O requests and **more than 900,000 IOPS** for random requests that were 4KB or less. It should be noted that VMware® recently indicated that it could achieve 300,000 IOPS for a single virtual machine (VM) running on a vSphere™ 5.0 host [3], using a different operating system in the VM and a different storage backend.

The KVM I/O rates were actually limited by the physical bandwidth of the storage setup. The average latencies for random reads and writes were very consistent up to the 4KB I/O size – at about 0.9 milliseconds (ms) for reads and 1.35 ms for writes. The average latencies for 8 KB requests were a little higher – 1.1ms for reads and 1.5 ms for writes – due to larger data transfer times. This data indicates that KVM can provide very *deterministic* performance for block I/O.

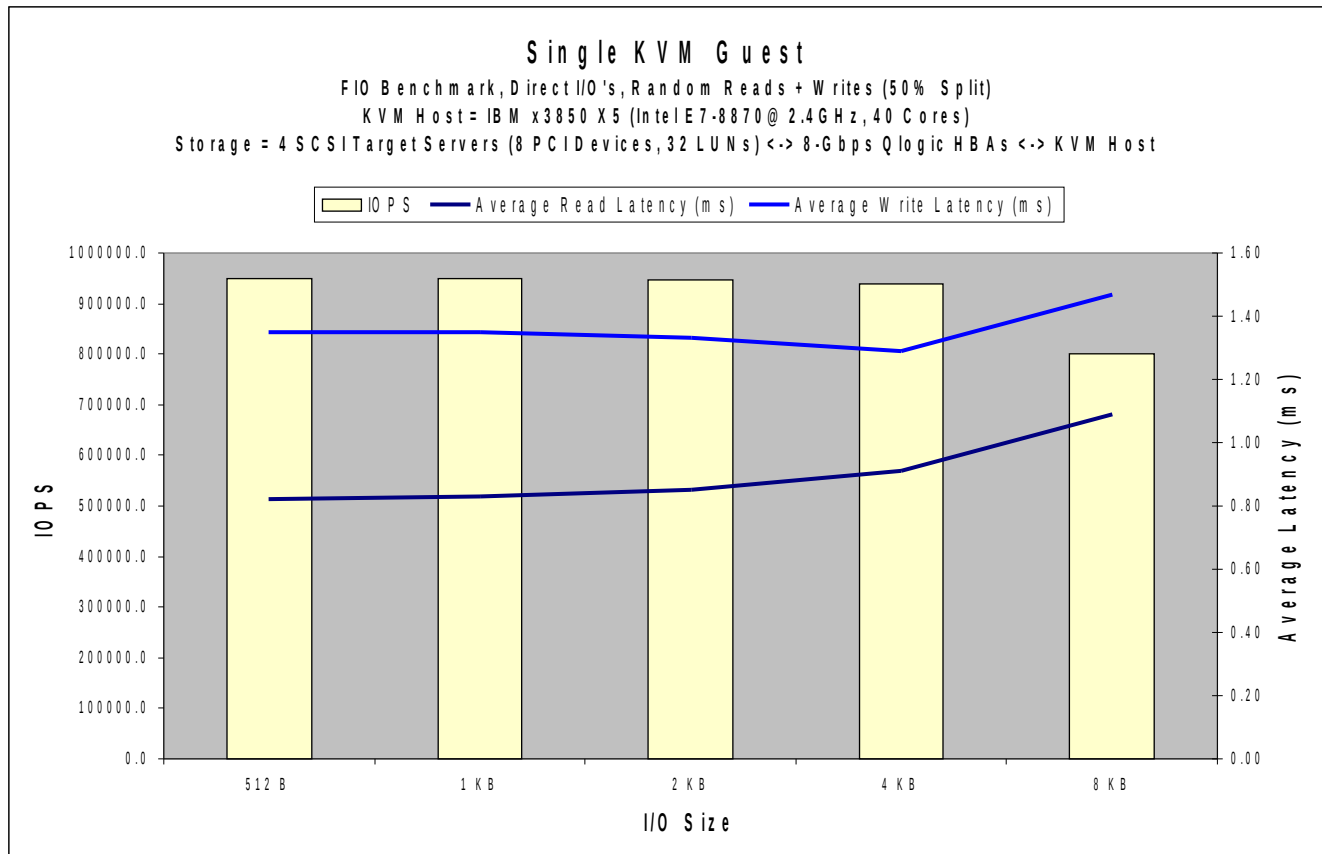


Figure 3. I/O rates and average latencies for a single KVM guest



Figure 4 shows the block I/O throughput that a single KVM guest can handle for different request sizes. Given the consistent I/O rates that KVM was able to support across different I/O sizes, the block I/O throughput scaled very well with the I/O size, from 463 MB/s for 512-byte requests to more than 6 GB/s for 8KB requests.

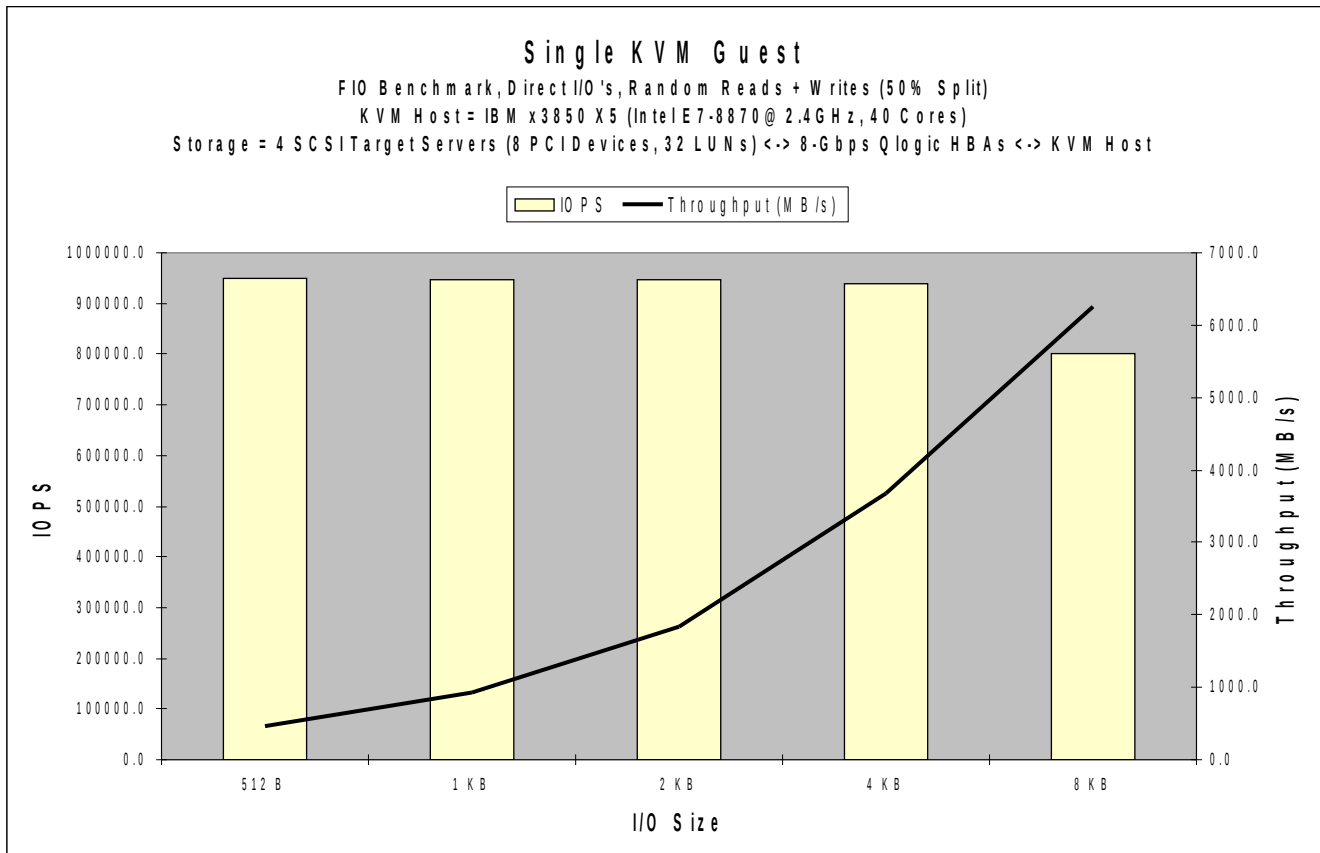


Figure 4. I/O throughput across various I/O request sizes for a single KVM guest

4.2 Multiple VMs

With multiple KVM guests, the goal was to determine the highest I/O rates that KVM could support with as few guests as possible. Because the IBM System x3850 X5 host server was a NUMA server with 4 nodes, one KVM guest was configured on each node for a total of 4 guests. This approach leverages all hardware resources on the host server. More specifically, the test was set up in the multi-guest configuration and had the following characteristics:

- Host Server: IBM System x3850 X5
 - 4 Intel Xeon E7-4870 processors (40 cores @ 2.40 GHz), 256 GB memory (total)
 - Red Hat Enterprise Linux 6.3
- Storage:
 - 7 QLogic QLE 256x (8 Gbps, dual ported) HBAs connected to 7 SCSI target servers
 - 14 PCI devices (ports) and 56 LUNs (each SCSI target server was backed by RAM disks and provided 2 PCI devices and 8 LUNs)
- KVM Guests (Virtual Machines) – Up to 4 KVM guests with each guest having:
 - 20 vCPUs (10 cores with 2 threads), 16 GB memory
 - 3 to 4 PCI devices, 12 to 16 LUNs
 - Red Hat Enterprise Linux 6.3
- FIO Workload
 - Random reads and writes (50% reads, 50% writes)
 - 1 job per LUN
 - Direct I/Os
 - Engine = libaio
 - Queue depth = 32

To leverage the caching efficiency and data locality, each KVM guest was bound to a single NUMA node on the host. Because there were 7 QLogic HBAs configured on the host server (host server had 7 available PCI slots), the ports and storage LUNs on these HBAs were divided among the 4 guests as follows:

- 2 QLogic HBAs with a total of 4 ports and 16 storage LUNs were passed to Guest 1
- 2 QLogic HBAs with a total of 4 ports and 16 storage LUNs were passed to Guest 2
- 1.5 QLogic HBAs with a total of 3 ports and 12 storage LUNs were passed to Guest 3
- 1.5 QLogic HBAs with a total of 3 ports and 12 storage LUNs were passed to Guest 4

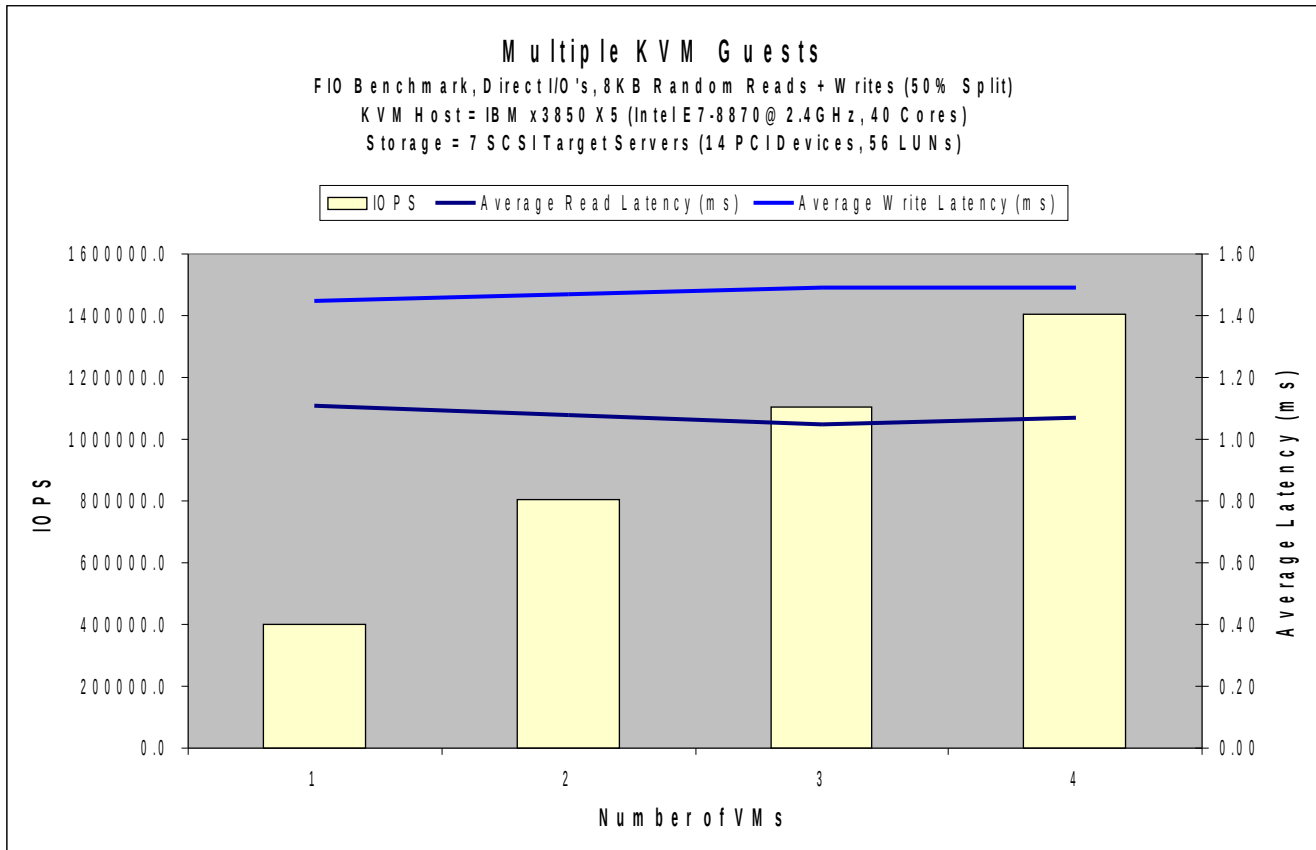


Figure 5. Aggregate I/O rates and average latencies for 8KB requests

Figure 5 shows the aggregate I/O rates for 8KB random read and write requests that KVM was able to support for one to four guests. During the test, KVM was able to sustain approximately 400,000 IOPS with one guest and **1.4 million IOPS with four guests**. These results are about the same as what could be obtained by running the same workload directly on the host server (bare metal). In other words, *these KVM results were actually limited by the test storage setup, not by KVM*. With a faster storage setup, higher I/O rates with KVM can potentially be achieved. It should be noted that VMware recently indicated that it could achieve an aggregate 1 million IOPS with 6 VMs running on the same vSphere 5.0 host [3], using a different operating system in the VMs and a different storage backend.

As more KVM guests were configured on the host server, the aggregate number of I/Os per second scaled very well while the *average latency for each I/O request stayed relatively constant*. The average latency was about 1.10 ms for 8 KB random reads and just under 1.50 ms for 8 KB random writes, regardless of the number of guests running on the same host, as shown in **Figure 5**.

Figure 6 shows that the average latencies also remained constant across different I/O request sizes from 512 bytes to 4 KB – with only a slight uptick at 8 KB. These results demonstrate very *deterministic* block I/O performance for KVM.

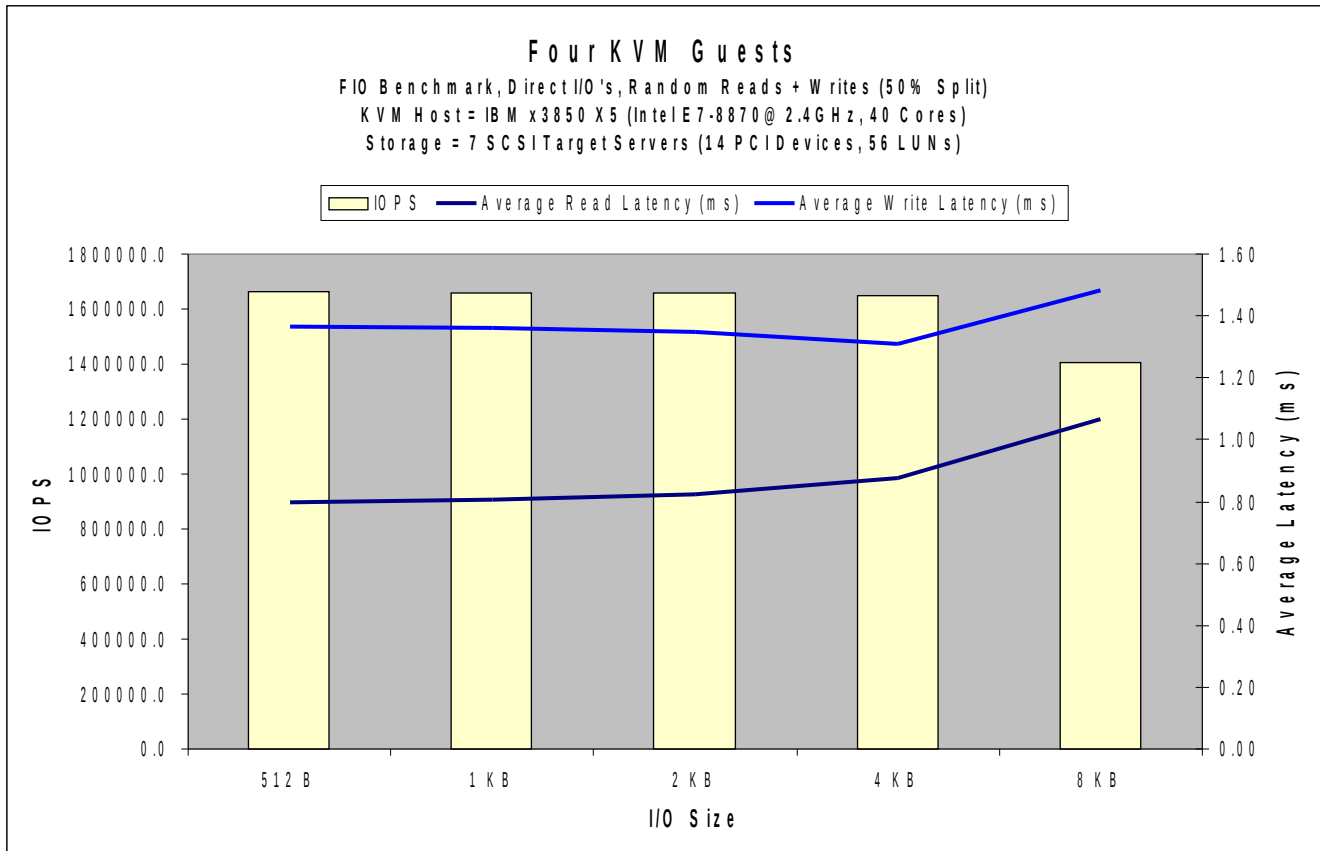


Figure 6. Aggregate I/O rates and average latencies for different I/O sizes



Thanks to the deterministic latencies and I/O rates across different I/O request sizes and numbers of KVM guests, the aggregate I/O throughput scaled very well with both I/O size and number of KVM guests, as shown in **Figure 7**. In fact, with 8 KB request size, the aggregate throughput for 4 KVM guests reached almost 11 GB/sec.

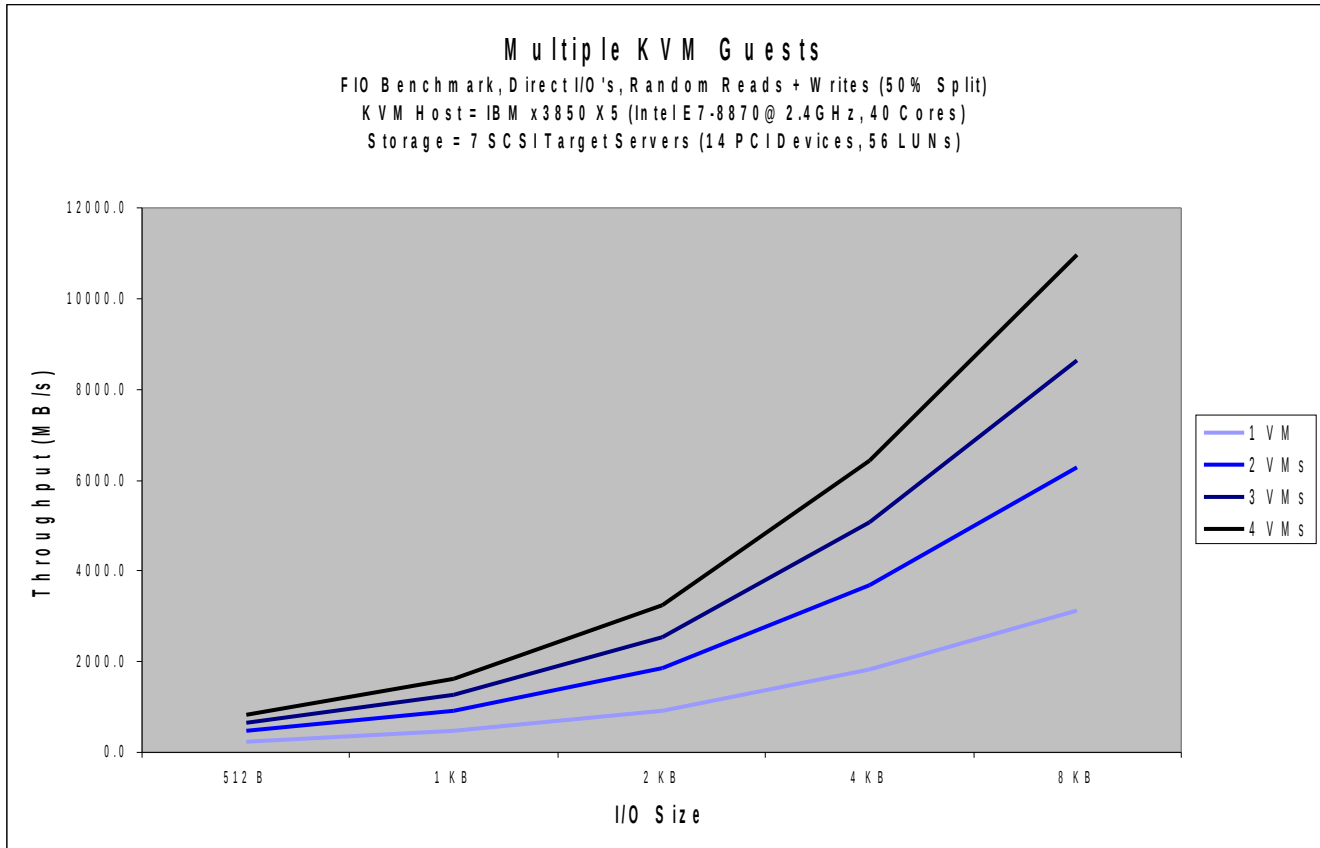


Figure 7. Aggregate I/O rates and average latencies with 4 KVM guests

5 Summary

Leveraging new silicon capabilities, together with tight integration into the Linux kernel, has allowed the Kernel-based Virtual Machine (KVM) to outperform any other hypervisor solution. With KVM, the block I/O performance can scale very well with the number of KVM guests and I/O request sizes. This paper has shown that a single KVM guest can handle more than 800,000 I/Os per second (IOPS) for 8KB random I/O requests and up to 950,000 IOPS for smaller requests. Four KVM guests running on the same host server can support more than 1.4 million IOPS for 8KB random I/O requests and 1.6 million IOPS for smaller requests – hitting the “bare-metal” limit of the test storage setup.

As a result, it has been proven that KVM is ready for enterprise workloads that demand extremely high I/O rates, very low latencies, and deterministic storage I/O performance. On-going improvements to the para-virtualization block drivers will enable similar levels of performance for virtio-blk and virtio-scsi.



Appendix A: References

- [1] SPECvirt_sc2010 Results Published by SPEC,
http://www.spec.org/virt_sc2010/results/specvirt_sc2010_perf.html
- [2] SAP SD Standard Benchmark Application Results, Two-Tier Internet Configuration,
<http://www.sap.com/solutions/benchmark/sd2tier.epx>
- [3] VMware Inc., “Achieving a Million I/O Operations per Second from a Single VMware vSphere 5.0 Host”, <http://www.vmware.com/resources/techresources/10211>, August 26th, 2011