IBM and Red Hat Enterprise Linux on the new IBM PureFlex System

Performance tuning and best practices guide for LAMP web server stack workload scalability and consolidation on IBM PureFlex System

Mark A. Peloquin and Karl Rister
IBM Systems and Technology Group ISV Enablement

April 2012

Table of contents

Abstract 1

Introduction 1

Prerequisites 2

Hardware description 2

The LAMP stack 3

Dell DVD Store database test suite 3

Initial setup 3

- Bridged networking 4
- Storage pools 5
- Software dependencies 10

Guest virtual machines 10

- Creating a template guest virtual machine 10
- Cloning guest virtual machines 13

Installing the LAMP software Stack 15

- MySQL server 15
- Apache HTTP server 15
- PHP script language 15
- Obtaining the DVD Store v2.1 files 16

Test methodology 16

Optimizations 19

- Operating system optimizations 20
  - I/O performance 20
  - swappiness 22
  - Increase QLogic FC driver queue depth 22
  - Kernel Samepage Merging 23
- LAMP stack optimizations 24
  - MySQL 24
  - Configuration file 24
  - innodb_buffer_pool_size 24
  - table_open_cache 25
  - thread_concurrency 25
  - query_cache_size 25
  - max_connections 26
  - innodb_log_file_size 26
  - Apache HTTP Server 26
  - Prefork or worker multi-processing module (MPM) 26
  - MaxClients 27
  - ServerLimit 27
  - StartServers 27
  - MinSpareServers, MaxSpareServers 28

PHP accelerators 28
Test results 28
Summary 31
Resources 32
Trademarks and special notices 33

Abstract

This paper describes how to install, configure, and optimize the Linux, Apache web server, MySQL database, and PHP web scripting language (LAMP) workload called DVD-Store in a virtualized environment using Red Hat Enterprise Linux with kernel-based virtual machine (KVM) running on IBM PureFlex System technology. DVD Store is an e-commerce web server application that simulates the basic attributes and functions of a web portal to browse and purchase DVD movies.

This paper focuses on best practices and tuning recommendations to assist IT departments, system administrators, and users to properly size, configure, and rapidly deploy LAMP workloads (such as DVD Store) in scalable virtual machines consolidated onto a single IBM PureFlex System x node. The guidance provided enables customers to become operational and well optimized, realizing better time to value.

Another key element is to use the IBM Storwize V7000 Unified storage controller and IBM System Storage Easy Tier feature, an automated data relocation function that manages the migration of hot data to and from solid-state drives (SSDs). The DVD Store workload generates a large random I/O load on the storage subsystem. The EasyTier feature of the Storwize V7000 Unified storage controller enables high rate of input/output operations per second (IOPS) and lower latencies to be supported on fewer number of drive spindles.

Introduction

This guide is the result of an effort to optimally align application, servers, and storage into a consolidated hardware solution offered in the IBM® PureFlex™ System. The PureFlex System converges servers along with high speed interconnect fabrics (integrated 10Gb and Fibre switching) and advance System Management into a small form-factor chassis. The PureFlex System chassis supports upto 14 servers, known as nodes. Nodes come in two flavors, one with IBM POWER® processors technologies and the other support
IBM System x® with Intel® Xeon® processors E5 delivering the highest levels of compute capacity available today. The IBM PureFlex System provide clients the flexibility to choose the number of servers to meet their current needs, but also the flexibility and scalability to add more servers as their needs grow. By optimizing the performance at both the hardware and application levels, clients can take advantage of the benefits with relevant and deployable solutions. The IBM PureFlex System and the IBM Storwize® V7000 Unified storage system provide a reliable, flexible, and intelligent platform for any client solutions, backed by advanced and proven technologies and features.

This paper determines the scalability of the LAMP stack and the DVD Store e-commerce web server workload on the IBM Next Generation Platform using Red Hat Enterprise Linux® 6 with KVM virtualization. For more information about the DVD Store database test suite, refer to the URL: http://linux.dell.com/dvdstore/

This solution provides value to users by significantly reducing the time it takes to install, configure, and tune a scale-out deployment of DVD Store workloads on the IBM PureFlex System. To use the superior computational capacity offered in the PureFlex System node, multiple virtual machines each running the DVD Store workload will be consolidated and run concurrently on a single System x server using the advanced hypervisor and virtualization (KVM) technologies available in Red Hat Enterprise Linux v6.2.

The results shown here are in the form of scalability and performance measurements along with the best practices and tuning recommendations that are relevant to other LAMP-based solutions providing insights into sizing and configuration of virtualized web-hosting solutions.

Prerequisites

The main goal of this paper is illustrate the optimization of the LAMP software stack performance and scalability in a virtualized configuration using the IBM PureFlex System platform running Red Hat Enterprise Linux 6 with KVM virtualization technology. The techniques and optimizations presented assume the following preexisting conditions:

- IBM PureFlex System with an Intel compute node and network connectivity
- Connection of high capacity and high performance storage to the compute node as raw block devices for virtual machine guest storage. This paper presents a scenario with an IBM Storwize V7000 Unified Storage Server (refer to the URL: **ibm.com**/systems/storage/disk/storwize_v7000/) or IBM System Storage® DS3500 (refer to the URL: **ibm.com**/systems/storage/disk/ds3500/) connected through Fibre Channel (FC).
- Red Hat Enterprise Linux 6.2 is installed on the Intel compute node with the selected software set being the virtualization Host. Other profiles will work, but this paper specifically addresses the software selection and initial configuration that the virtualization host provides.
- Availability of the Red Hat Enterprise Linux 6.2 installation sources for creating virtual machine guests. Several options, such as ISO images or PXE installation sources exist, the steps provided in this paper assume ISO images.
- Accessibility to the Dell DVD Store application.
- A Linux-based management workstation with support for X Window System. A Red Hat Enterprise Linux 6.2 installed desktop should work just fine.
- Familiarity with *bash* or other command-line interface (CLI). The procedures outlined in this paper have GUI equivalents, but emphasis is placed on the CLI so that best practices put forth can be used in scripted automation.
- Assumes the Red Hat *yum* software package manager has been already configured by your IT organization to access Red Hat online repositories or those from a satellite server.

Hardware description

LAMP stack optimizations were conducted using IBM newest and most powerful modular Intel x86 node. This node came equipped with:

- One (of two) Intel Xeon processor E5-2680 operating at 2.70 GHz
- 48 GB of RAM (twelve 4 GB dims)
- Twp 300 GB 10k rpm SAS drives
- Two 10Gb Ethernet adapters (on-board)
- One QLogic 8Gb FC host bus adapter

The node was housed in an PureFlex System chassis which has:

- Capacity for up to 14 nodes
- One (of two) Chassis Management Module
- One (of two) 20 port 10Gb Ethernet switches
- One (of two) xx20 port 8Gb FC switches

- Dual 2 kilowatt redundant power supplies

An IBM Storwize V7000 Unified storage server was attached externally and configured with:

- 20 600 GB 10k rpm SAS drives
- IBM System Storage Easy Tier® feature using four 290 GB SAS SSDs

The LAMP stack

The open source LAMP application stack is commonly used and widely considered as a reliable foundation for web servers and web services.

LAMP is an acronym referring to the first letters of Linux (operating system), Apache (HTTP Server), MySQL (database) and PHP (scripting language that can also be Perl or Python). These components comprise the main software elements used to create viable general purpose web servers.

Despite the explosive growth in the Internet and new products that fuel that growth, the LAMP stack is still used in more websites than any other product available. In a time with so many products and so much commercialization, the reasons for LAMP to be still popular are, it is an open source, it is scalable, it is low in cost, it is secure and delivers good performance even on older and smaller systems.

Dell DVD Store database test suite

The workload selected to test the LAMP stack is called DVD Store developed by Dell Computers.

The Dell DVD Store is an open source simulation of an online e-commerce site with implementations in Microsoft® SQL Server, Oracle, MySQL, and PostgreSQL along with driver programs and web applications

The DVD Store application has been designed in such a way that the database component uses many advanced database features (such as transactions, stored procedures, triggers, referential integrity and so on) while keeping the database easy to install and understand.

This release includes data generation programs, shell scripts to build data for 10 MB, 1 GB, and 100 GB versions of the DVD Store, database build scripts and stored procedure, PHP web pages, and a C# driver program.

Over the years, DVD Store has had several revisions. In December, 2010, DVD Store v2.1 was released. This version is tested in this paper.

The driver program supplied in DVD Store reports throughput and response time metrics to the console driver system. These statistics make this test suite a reasonable choice to test and illustrate characteristics of the LAMP stack. These metrics are helpful to quantify the scalability and the effects that many of the potential changes to the component configuration files can have on the resulting behavior of the environment.

Initial setup

Before beginning to create guest virtual machines, there are a few setup steps that must be performed on the virtualization host to configure the network and storage resources and to install software dependencies. The processes outlined here for bridged networking and storage pool setup are only

representative and specific requirements of a given installation might require deviation from the presented configuration.

Bridged networking

In order for the guest virtual machines to be able to communicate externally to the virtualization host, a networking bridge must be created. This can be done by completing the following steps[*]:

1. Change to the network scripts directory.

```
[root@host ~]# cd /etc/sysconfig/network-scripts
```

2. Identify the network interface to which you need to attach the bridge. Usually this is the interface that was configured during the installation of Red Hat Enterprise Linux on the host. In most situations, this can be determined by running the following command that shows the interface (eth0) over which the default route exists:

```
[root@host network-scripts]# ip route | grep default
default via 10.0.0.1 dev eth0
```

3. Modify the configuration file that corresponds to the interface identified in the previous step. In this case, `/etc/sysconfig/network-scripts/ifcfg-eth0`, by adding a line to attach the interface to a bridge named `br0`.

```
[root@host network-scripts]# cat ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="5C:F3:FC:6E:31:50"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
BRIDGE="br0"
```

4. Create a configuration file for the bridge named `br0` and add the following contents to it,

```
[root@host network-scripts]# cat ifcfg-br0
DEVICE="br0"
TYPE="Bridge"
BOOTPROTO="dhcp"
ONBOOT="yes"
DELAY="0"
```

5. Restart the network to activate the new configuration,

```
[root@host network-scripts]# service network restart
Shutting down interface eth0:  bridge br0 does not exist!
```

[*] Steps adapted from http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/sect-Virtualization_Host_Configuration_and_Guest_Installation_Guide-Network_Configuration-Network_Configuration-Bridged_networking_with_libvirt.html

```
                                                           [  OK  ]
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:                                [  OK  ]
Bringing up interface br0:
Determining IP information for br0... done.
                                                           [  OK  ]
```

6. Configure iptables and Linux firewalling to forward all the traffic across the newly created bridge.

```
[root@host network-scripts]# iptables -I FORWARD -m physdev --physdev-is-
bridged -j ACCEPT
[root@host network-scripts]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
[root@host network-scripts]# service iptables restart
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Setting chains to policy ACCEPT: nat mangle filte[  OK  ]
iptables: Unloading modules:                               [  OK  ]
iptables: Applying firewall rules:                         [  OK  ]
```

7. The bridge configuration can be validated by using the `brctl` command. The output looks as shown in this step.

```
[root@host network-scripts]# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.001a64f1ba66       no              eth0
virbr0          8000.52540007281d       yes             virbr0-nic
```

8. The `virbr0` bridge is created by Libvirt and is used to enable private network with guest virtual machines. This feature is useful in desktop configurations, but it is generally not used in a server configuration. If not using private networking with guest virtual machines, `virbr0` can be deactivated by running the following commands.

```
[root@host ~]# virsh net-destroy default
Network default destroyed

[root@host ~]# virsh net-autostart default --disable
Network default unmarked as autostarted
```

Storage pools

Libvirt, the default virtualization management infrastructure built into Red Hat Enterprise Linux, provides a storage pool that is part of the root file system with properties that can be viewed using the following command:

```
[root@host ~]# virsh pool-info default
Name:           default
UUID:           f907c161-9e5e-cc0b-b974-af2d65d1ff00
```

```
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       209.60 GB
Allocation:     5.07 GB
Available:      204.53 GB
```

The default pool is created the first time that a Libvirt management program is launched, which means that the command will have no output initially. Examples of commands that would create the default pool are `virt-manager` or `virt-install`. In most cases, where high performance I/O is a requirement, alternatives to storage in the compute node might need to be considered to achieve greater capacity and performance. If the storage performance and capacity requirements are low, using the storage that is part of the compute node can be done using the default pool. In order to configure the high performance storage provided by the Storwize V7000 Unified storage, proper block devices must first be identified. To do so, look at the contents of `/proc/partitions`:

```
[root@host ~]# cat /proc/partitions
major minor  #blocks  name

   8       64  291991552 sde
   8       65     512000 sde1
   8       66  291478528 sde2
   8       16 2340421632 sdb
   8       32 2340421632 sdc
   8       48 2340421632 sdd
   8        0 2340421632 sda
 253        0 2340421632 dm-0
 253        1 2340421632 dm-1
 253        2  223285248 dm-2
 253        3   68190208 dm-3
```

In this example, the Storwize V7000 Unified storage is identified by the following device names: `sda`, `sdb`, `sdc`, and `sdd`. The Storwize V7000 Unified storage is configured to export two equal size volumes to the compute node and the use of multipath means that four devices are seen. As multipath is being used, it is important to identify the multipath device names. To do so, run the following command:

```
[root@host ~]# multipath -ll
mpathc (360050768028201fa840000000000000c) dm-1 IBM,2145
size=2.2T features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| `- 0:0:0:1 sdb 8:16 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  `- 0:0:1:1 sdd 8:48 active ready running
mpathb (360050768028201fa840000000000000b) dm-0 IBM,2145
```

```
size=2.2T features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| `- 0:0:1:0 sdc 8:32 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  `- 0:0:0:0 sda 8:0  active ready running
```

This output identifies the two multipath devices as `mpathb` and `mpathc`. These devices can be referenced as `/dev/mapper/mpathb` or `/dev/mapper/mpathc`. Before creating a new storage pool, perform the following steps to use Logical Volume Manager (LVM) to abstract the raw block devices from the overlying layers. This allows for the future possibility of expanding the storage capacity by adding additional storage resources when the need arise.

1. As the first step to creating a LVM volume, initialize the block devices for use in LVM.

```
[root@host ~]# pvcreate /dev/mapper/mpathb /dev/mapper/mpathc
  Writing physical volume data to disk "/dev/mapper/mpathb"
  Physical volume "/dev/mapper/mpathb" successfully created
  Writing physical volume data to disk "/dev/mapper/mpathc"
  Physical volume "/dev/mapper/mpathc" successfully created
```

2. After the block device has been initialized, create a LVM volume group (VG, named `V7000` here) which contains the devices.

```
[root@host ~]# vgcreate V7000 /dev/mapper/mpathb /dev/mapper/mpathc
  Volume group "V7000" successfully created
```

3. Following creation of the VG, it is time to create the LVM Logical Volume (LV) itself. The `-l 100%FREE` argument is a simple way of telling LVM to allocate all of the available space to the LV. The `-i 2` option informs LVM to stripe the data across the two underlying block devices which can improve performance. The `-n volume` option specifies the LV name. The final argument, `V7000`, specifies the VG within which to create the LV:

```
[root@host ~]# lvcreate -l 100%FREE -i 2 -n volume V7000
  Using default stripesize 64.00 KiB
  Logical volume "volume" created
```

4. Verify the properties of the newly created LV using the following command:

```
[root@host ~]# lvdisplay --maps /dev/V7000/volume
  --- Logical volume ---
  LV Name                /dev/V7000/volume
  VG Name                V7000
  LV UUID                eesOIz-CoT8-X549-JYhq-Tm4d-nZ9Q-tqXSGr
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                4.36 TiB
  Current LE             1142782
```

```
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     512
  Block device           253:4

  --- Segments ---
  Logical extent 0 to 1142781:
    Type            striped
    Stripes         2
    Stripe size           64.00 KiB
    Stripe 0:
      Physical volume     /dev/mapper/mpathb
      Physical extents    0 to 571390
    Stripe 1:
      Physical volume     /dev/mapper/mpathc
      Physical extents    0 to 571390
```

In order to create a new storage pool from the LV, perform the following steps:

1. Format the LV with the `ext4` file system, which is the replacement for Linux venerable `ext3` file system. There are specific advantages of virtualization that using `ext4` provides and this is explained later in this paper:

```
[root@host ~]# mkfs.ext4 /dev/V7000/volume
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=16 blocks, Stripe width=32 blocks
292552704 inodes, 1170208768 blocks
58510438 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
35712 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
2654208,
      4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
```

```
        102400000, 214990848, 512000000, 550731776, 644972544

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

2. Before performing the next step, the mount point at which the storage pool will be located must be created. For this scenario `/V7000` is used:

```
[root@host ~]# mkdir /V7000
```

3. The next step is to define the storage pool using the `virsh` command, which is a CLI to the Libvirt management infrastructure

```
[root@host ~]# virsh pool-define-as V7000 fs - - /dev/V7000/volume V7000
/V7000
Pool V7000 defined
```

4. After defining the storage pool, it is necessary to tell Libvirt to autostart it each time the system is booted. If this step is not performed, the storage pool might need to be manually activated after each system boot.

```
[root@host ~]# virsh pool-autostart V7000
Pool V7000 marked as autostarted
```

5. At this point, the storage pool has been created and is properly configured, but it is not active. Run the following command to start the storage pool:

```
[root@host ~]# virsh pool-start V7000
Pool V7000 started
```

6. Verify the settings of the newly created storage pool.

```
[root@host ~]# virsh pool-info V7000
Name:           V7000
UUID:           ece7f152-f249-a624-ddf8-47862bcd978b
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       4.29 TB
Allocation:     192.05 MB
Available:      4.29 TB
```

7. If required, view the storage pool using the `df` command.

```
[root@host ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_host-lv_root
```

```
                      210G  5.2G  194G   3% /
tmpfs                  32G     0   32G   0% /dev/shm
/dev/sde1             485M   38M  423M   9% /boot
/dev/mapper/V7000-volume
                      4.3T  193M  4.1T   1% /V7000
```

Software dependencies

Some of the commands that are used later in this paper have software dependencies that are not met by the software set virtualization host. The required packages can be installed either by the `yum` utility or by using the `rpm` command in association with the available installation sources. There are several additional dependencies that are required for the packages that are needed which make `yum` an ideal method of installation as it can resolve dependencies automatically. The packages that need to be installed provide support for the GUI software on Linux called X Window System. To install the packages, run the following command:

```
[root@host ~]# yum install xorg-x11-xauth xorg-x11-server-Xorg
```

Enter your answer as *yes* when prompted to continue and the packages will be installed. The output is quite lengthy so it has been omitted from this example.

Guest virtual machines

After the initial networking and storage setup is complete, the virtualization environment is ready to have guest virtual machines installed. In order to be able to rapidly deploy many guest virtual machines on the system, a technique will be presented where a template guest virtual machine is created and this can be cloned each time a new guest virtual machine needs to be created.

Creating a template guest virtual machine

Templates are a useful mechanism for quickly deploying new guest virtual machines as needed. In the testing done for this paper, there were two types of guest virtual machines that were needed and each one could be created from its own template. For the purposes of brevity, the steps outlined here show how to create a single template from which many guest virtual machines can be created.

1. Connect to the virtualization host from the management workstation with X11 forwarding over the SSH protocol enabled. This allows for GUI applications to be launched on the local workstation from the remote virtualization host. This is required because when a guest virtual machine installation is started, a GUI console will be launched by the virtualization host. The following output containing `xauth` shows that the X11 forwarding is properly configured.

   ```
   root@mgmt:~# ssh -X root@host
   root@host's password:
   Last login: Fri Feb 17 10:05:34 2012 from mgmt
   /usr/bin/xauth:  creating new authority file /root/.Xauthority
   [root@host ~]#
   ```

2. The next step is to begin installation of the guest virtual machine template. This can be done using the `virt-install` command, which takes several arguments that can be used to customize the

configuration of the guest virtual machine. Several of these customizations are important performance optimizations that aides in increasing guest virtual machine performance, scalability, and density. Here is the command invocation and the resulting output which includes the launching of the guest virtual machine console in a `virt-viewer` window:

```
[root@host ~]# virt-install --name template --ram 2560 --vcpus 1 --cdrom
/iso-images/RHEL6.2.iso --os-variant rhel6 --disk
pool=V7000,size=100,cache=none,format=raw,bus=virtio,io=native --network
bridge=br0,model=virtio --cpu host


Starting install...
Allocating 'template.img'                               | 100 GB     00:00
Creating domain...                                      |   0 B      00:00
```

Figure 1: Red Hat Enterprise Linux 6.2 Installation in a guest virtual machine

3. Complete the installation of Red Hat Enterprise Linux 6 inside the virtual machine. There should be no special steps to perform as the installer ensures that the correct device drivers are installed and configured.

4. After the installation is complete, closing the `virt-viewer` window should result in the following final output from the `virt-install` command:

```
[root@host ~]# virt-install --name template --ram 2560 --vcpus 1 --cdrom
/iso-images/RHEL6.2.iso --os-variant rhel6 --disk
```

```
pool=V7000,size=100,cache=none,format=raw,bus=virtio,io=native --network
bridge=br0,model=virtio --cpu host


Starting install...
Allocating 'template.img'                                | 100 GB     00:00
Creating domain...                                       |   0 B      00:00
Guest installation complete... restarting guest.
```

5. Shut down the newly created guest virtual machine template.

```
[root@host ~]# virsh shutdown template
Domain template is being shutdown
```

It is important to note and understand the specific customizations that were performed with the arguments to the `virt-install` invocation. The recommendation of these customization settings are part of the results of extensive testing of Red Hat Enterprise Linux 6.2 on the IBM PureFlex System platform for this paper.

```
--ram 2048 --vcpus 4 --cdrom /iso-images/RHEL6.2.iso --os-variant rhel6
```

For the initial LAMP stack with DVD Store testing scenario used in this paper, all guest virtual machines running on the virtualization host are configured with one virtual processor and 2.5 gigabytes (GB) of memory. These settings are configured by the `--ram 2560` and `--vcpus 1` options. The `--cdrom` parameter instructs the hypervisor of the location of the ISO image that will be used for the installation. On completion of the installation process, the ISO image will be disconnected from the guest virtual machine and an empty CD-ROM device will remain. The `--os-variant rhel6` parameter informs the hypervisor about the type of operating system that will be running in the guest virtual machine. This is important for defining the interfaces that the hypervisor makes available. In general, more recent operating systems enable additional interfaces and this improves performance.

```
--disk pool=V7000,size=100,cache=none,format=raw,bus=virtio,io=native
```

Due to the disk I/O characteristics of the MySQL database component of the LAMP stack, the arguments to the `--disk` parameter are particularly important. The storage pool to use is configured by the `pool=V7000` parameter, and the `size=100` parameter sets the size of the virtual disk in GB. The size (100 GB) used in this testing was arbitrarily chosen and was much larger than required for the LAMP stack and the DVD Store workload; it must be adjusted on an as-needed basis. When the disk image is created it will be a sparse file; this means that it is allocated only the space that it uses and not the actual size that is requested. The sparse nature of the virtual disk results in significant time savings when creating the template guest virtual machine and it will be important later in the paper when creating many guest virtual machines by cloning the template. The `cache=none` setting instructs the hypervisor not to allow the virtualization host to cache I/O request data, which this testing found to be advantageous. Specifying `format=raw` instructs the hypervisor not to use an advanced disk image format which is generally good for performance. Setting `bus=virtio` tells the hypervisor to use a para-virtual device for the disk instead of an emulated one. Using para-virtual devices allows for virtualization-specific optimizations to be made to the device drivers and I/O model which helps increase performance and efficiency. The KVM hypervisor that is embedded into Red Hat Enterprise Linux uses a para-virtual device model called VirtIO. When performance is important in a virtualization environment, para-virtual devices should be used wherever possible. The last setting, `io=native`, instructs the hypervisor to use asynchronous I/O for performing I/O

on behalf of the guest virtual machine. Testing has shown that this generally improves performance but it is not the default setting, so it must be manually set.

```
--network bridge=br0,model=virtio
```

The network settings for a guest virtual machine are much simpler than what is required for disk I/O which results in only two parameters specified to the `--network` option. Specifying `bridge=br0` tells the hypervisor to connect the guest virtual machine's network device to the network bridge that was configured earlier in this paper. This connection allows for network traffic to flow to and from the guest virtual machine. The second parameter to set is `model=virtio`. As with the disk device, using the VirtIO para-virtual network device results in improved performance and efficiency when compared to the emulated device alternatives.

```
--cpu host
```

The last setting specified to `virt-install` with the `--cpu` option is used to inform the guest virtual machine of the advanced capabilities present in the Intel Xeon processors E5-2600 series in the IBM PureFlex System compute node. Specifying `host` tells the hypervisor to pass through all capabilities of the physical processor which can prevent the hypervisor from having to perform emulation if the guest virtual machine attempts to use those capabilities (for example, machine instructions from advanced processor extensions such as SSE). There are potential issues when specifying the `--cpu` parameters to a guest virtual machine. The testing performed for this paper with the LAMP stack and DVD Store shows that these settings provide substantial performance gains which result in improved scalability and higher levels of consolidation density. The potential issue with specifying the `--cpu` parameter arises when an environment contains heterogeneous hardware configurations as guest virtual machine migration between virtualization hosts with different processor types might not be possible. For example, a guest virtual machine configured to use the Westmere processor type cannot run on a virtualization host with Nehalem processors, but a guest virtual machine configured to use the Nehalem processor type can run on a virtualization host with Westmere processors as it is compatible with earlier processor versions. However, if all the virtualization hosts that a guest virtual machine can potentially run on have the same hardware configuration, then this is not an issue. Due to the launch of the Intel Xeon processors E5-2600 series, the version of Libvirt in Red Hat Enterprise Linux 6.2 recognizes the processors as the Westmere type.

Cloning guest virtual machines

After the template guest virtual machine has been created, it is possible to rapidly deploy guest virtual machines by cloning it. The command used to clone the template is `virt-clone`, and it can be invoked as shown here:

```
[root@host ~]# virt-clone --original template --name guest-vm-1 --file
/V7000/guest-vm-1.img
Allocating 'guest-vm-1.img'                              | 100 GB     02:25


Clone 'guest-vm-1' created successfully.
```

This example uses three parameters to `virt-clone`, but additional parameters can be used for further customization (such as setting a specific Media Access Control (MAC) address for the network device). The `--original` parameter specifies the guest virtual machine name of the template. Specifying the `--name` parameter defines the name of the newly created guest virtual machine and the `--file` parameter specifies the name and location of the newly created disk image.

In this example, the 100 GB disk image from the template was cloned in two minutes and 25 seconds which is much faster than the creation of the original template, which took about 15 minutes. This represents an 84% reduction in provisioning time for a single guest, allowing six guest virtual machines to be created in the same time as one standard installation. The time it takes to perform a clone operation is dependent on three different attributes: storage I/O performance; disk image size; and the amount of content in the disk image. Due to the sparse nature of the template image, the clone operation does not have to read all 100 GB of data from the template image. As `ext4` is used as the host's file system, the new image created as part of the clone operation will be preallocated (not sparse) but still be created without having to write out every data block which decreases the time to complete the clone operation. Using preallocated images is generally good for performance and `ext4` allows that without having to spend significant amounts of time doing the allocations. One downside to these optimizations is that it makes the progress indicator that `virt-clone` provides very inaccurate, in fact the closer the cloning operation comes to completion, the more inaccurate the progress indicator becomes. One aspect of the progress indicator that is fairly accurate is the amount of data (which can be used to gauge actual progress) that has been copied. As the template image is a sparse file, it is possible to determine the amount of data to copy before the clone operation by using the following command:

```
[root@host ~]# qemu-img info /V7000/template.img
image: /V7000/template.img
file format: raw
virtual size: 100G (107374182400 bytes)
disk size: 6.3G
```

Here, the utility correctly reports that the disk image is 100 GB in virtual size but that only 6.3 GB has actually been used. The used data is all that has to be cloned because of the `ext4` optimizations.

As the cloning of a guest virtual machine can be done entirely from the CLI, it is possible to automate the creation of many guest virtual machines. This can be done using scripts or using the CLI as in the following example:

```
[root@host ~]# for i in `seq 1 5`; do virt-clone --original template --name
guest-vm-$i --file /V7000/guest-vm-$i.img; done
```

In this example, five guest virtual machines will be created in sequence. After running the command, it is possible to check their existence by running a specific `virsh` command:

```
[root@host ~]# virsh list --all
 Id Name                 State
----------------------------------
  - guest-vm-1           shut off
  - guest-vm-2           shut off
  - guest-vm-3           shut off
  - guest-vm-4           shut off
  - guest-vm-5           shut off
  - template              shut off
```

After guest virtual machines have been cloned from the template, they must be customized with any modifications that were not generic enough to commit to the template. In the testing performed for this

paper, customization of the various SPECjEnterprise2010 components was done to make the individual guest virtual machines ready for testing. To start and stop a guest virtual machine, use the following commands:

```
[root@host ~]# virsh start guest-vm-1
Domain guest-vm-1 started

[root@host ~]# virsh shutdown guest-vm-1
Domain guest-vm-1 is being shutdown
```

Installing the LAMP software Stack

For the purposes of this paper, the most traditional definition of LAMP was followed using MySQL for the database and PHP as the scripting language.

Before attempting to install and configure DVD Store, you need to install the other LAMP components that the test suite requires.

Fortunately, all LAMP software stack components likely to be required by most LAMP applications are prebuilt and distributed as part of Red Hat Enterprise Linux 6. The following is a list of the specific Red Hat Package Manager packages along with the `yum` commands to install the LAMP stack components that is required to run the DVD Store application.

MySQL server

1. RHEL 6 includes MySQL server v5.1.52. To install the required MySQL packages, issue the following command:

   ```
   [root@host ~]# yum install mysql-client mysql-devel mysql-shared-common
   mysql-shared mysql-shared-compat mysql-embedded mysql-test mysql-server
   perl-DBI perl-DBD perl-DBD-mysql
   ```

2. Use the following command to enable it to start at boot:

   ```
   [root@host ~]# chkconfig –level 35 mysqld on
   ```

Apache HTTP server

3. Apache server v2.2.15 is available. If you fulfilled the prerequisites in this paper or if you have installed the virtualization host software selection, then Apache should already be loaded on your system and steps 3 and 4 can be skipped. If Apache needs to be installed, run the following command:

   ```
   [root@host ~]# yum install httpd
   ```

4. To configure Apache to start it at boot, run the following command:

   ```
   [root@host ~]# chkconfig --level 35 httpd on
   ```

PHP script language

5. PHP v5.5.3 is the included version. To install PHP, run the following command:

   ```
   [root@host ~]# yum install php php-common-php-pdo php-mysql
   ```

The basic LAMP stack components are now loaded on the system. At this stage, you might have a reasonably complete LAMP stack installed that is more than capable to support most of the web applications. If a specific LAMP application requires additional packages, they should be identified in the application's documentation, called out during the install, or possibly supplied with the install files.

Obtaining the DVD Store v2.1 files

This is the only software component used in this paper that is not distributed as part of Red Hat Enterprise Linux 6.

Even though this test suite supports several database servers, only MySQL is used and discussed.

To download the applicable files for this version of DVD Store, run the following commands:

```
# wget http://linux.dell.com/dvdstore/ds21.tar.gz
# wget http://linux.dell.com/dvdstore/ds21_mysql.tar.gz
# wget http://linux.dell.com/dvdstore/readme.txt
```

The main readme.txt file, along with additional readme.txt files within the DVD Store tar files, describe installation, setup, configuration instructions, and details on how to run the test suite.

The LAMP stack is required to be installed to complete many of the setup and configuration steps called out in the readme files.

Test methodology

Workload properties

The LAMP application used to evaluate LAMP stack scalability and performance on the PureFlex System is DVD Store. The DVD Store application creates an online store where users can browse for and purchase DVDs. DVD Store is a three-tier test suite. First is the driver tier, which represents users and simulates user activities. It exercises four activities that are supported by the application: *Logon, New Customer, Browse*, and *Purchase*. These activities represent different transaction types. The second tier is the application tier. This tier reads and runs different PHP scripts (one for each transaction type) stored on the website. The PHP scripts accept user-specified information, both account and product related, and communicates and interacts with the third tier which is the MySQL database. Responses are received from the database by the application (second) tier, which then produces HTML pages that are sent back to the simulating user browser sessions in the driver (first) tier.

The DVD Store Release 2 is available from the URL: http://linux.dell.com/dvdstore. The application includes support for a back-end database component, a PHP web application layer, and driver programs to simulate users. The goal in designing the database component as well as the middle-tier application was to use many advanced database features (such as transactions, stored procedures, triggers, referential integrity and so on) while keeping the database easy to install and understand. The DVD Store

Release 2 workload can be used to test databases or as a stress tool for any purpose. The code is licensed under the GNU General Public License (GPL).[1]

The DVD Store application supports standard database sizes of 10 MB (small), 1 GB (medium), 100 GB (large), and also custom sizes.

The driver program produces real-time metrics to the console that can provide insight about how well the LAMP stack functions on the test system. Of the metrics displayed, this subset is particularly useful:

- *Elapsed time* since the driver started the session with web store
- Number of *overall purchases* (running total)
- Number of *orders per minute* (running average)
- *Maximum response time* in the last 100 transactions
- *Average response time* from the start of the session

The *overall purchases* and *the orders per minute* results both generally describe the amounts of work generated by the driver program. The number of *overall purchases* shows the total amount of work that has been completed during the run while the *order per minute* represents the average load that is sustained.

The response time metrics provide an indication of the user experience and can also be known as quality of service (QoS). Response time is the duration between the moment a request is submitted by a user and the moment a response is received. It is the round-trip or request-to-response delay. For DVD Store, QoS refers to how quickly the website responds to a user. The DVD Store driver reports response times in two ways. First, there is *maximum response time* which shows the worst or longest delay of the last 100 responses received. This is useful to show what is happening to the system in the last few seconds. Few workloads have a flat load characteristic for the life of a run. Ebbs and flows are normal and expected. However, the magnitude and frequency of the changes must be understood to know if a problem might be unfolding or about to unfold. The second way it is reported is the *average response time* which provides the typical delay for all responses received. This is a good method to detect and visualize trends and to get a general sense of how (under or over) loaded a system might be.

Even though the DVD Store driver does report response times, there are no defined criteria or indication of acceptable values. For the purpose of this paper, a reasonable quality of service value is selected. As reasonable is an overly vague and subjective reference, it was decided to use a more recognized standard as the acceptable criteria of QoS. Several industry standard benchmarks define acceptable levels of QoS as 95%, where all response times must be 2 seconds or less and 99%, where all response time must be 4 seconds or less. That was the pass / fail criteria used in this paper.

These metrics are specifically tied to DVD Store test suite, but in general, must have some degree of relevance to other workloads. Most LAMP workloads deal with users and information exchange, and therefore have concerns about (response) time and throughput which translate into user experience, QoS, work completed, and system load rates.

This last one, load rates, is very important. Workloads need to be evaluated not only at typical or target loads, but also for peak periods of traffic. IT departments, web and system administrators should have

[1] The DVD Store readme.txt

some informed notions of the expected work load levels and the goals to achieve an acceptable experience by those visiting the website.

In this paper, the DVD Store web application is used to show how well the platform, operating system, and the LAMP stack function together. The metrics produced by the DVD Store driver tier are used to compare and contrast the LAMP stack scalability. Scalability can occur in two ways, known as scale-up and scale-out. Scale-up dedicates more resources to a single instance of the stack. Scale-out, using KVM virtualization, uses multiple instances of a stack. In both cases, scaling (up or out) is employed to provide the improved opportunity to support greater load and accomplish more work.

DVD Store test configurations:

The goal of this study is to determine the scaling configuration of the LAMP stack that allows the PureFlex System using a Storwize V7000 Unified storage server running Red Hat Enterprise Linux 6 to complete the greatest amount of work in a given period of time while meeting QoS targets. For the measure of work, the *order per minute* result is used and *average response time* is used for QoS.

The DVD Store driver supports many parameters. However, for testing purposes only those values specified in Table 1 were changed from the default values.

Parameter name	Assigned value
n_threads	256 (for baremetal) 50 x number of virtual processors
db_size	10 GB x number of virtual processors
think_time	0.25 seconds
run_time	10 minutes
warmup_time	1 minute
ramp_time	10 users/second
pct_newcustomers	0

Table 1: DVD Store driver parameters

To provide a more complete picture of the abilities of scaling and consolidating the LAMP stack in a KVM-virtualized environment, testing was performed in an environment where virtualization is not active, known as running bare-metal. Virtualization does not come free and there is additional resource utilization associated with using it. Bare-metal comparisons can be useful to illustrate and quantify the additional resource consumption incurred when using KVM.

Based on preliminary bare-metal testing, it was decided that the DVD Store driver would simulate 16 users for each virtual processors used in all the virtual machines included in the test. Similarly, the memory and the database were sized at 2.5 GB and 10 GB respectively per virtual processor. As additional virtual processors were added to guest virtual machine configurations, the number of users, memory, and database sizes increased by 16, 2.5 GB, and 10 GB, respectively. Testing began with 16 users, 2.5 GB of memory, and 10 GB database for a guest virtual machine with one virtual processor and increasing up to 256 users with the largest guest virtual machine having eight virtual processors and 80 GB database. (Refer to Table 2.)

Number of (virtual) processors	Memory (GB)	Database size (GB)	Number of virtual machines tested
1	2.5	10	1,2,4,8,16
2	5	20	1,2,4,8
4	10	40	1,2,4
8	20	80	1,2
16	40	160	1 (bare-metal)

Table 2: Tested configurations

8 GB of memory was reserved for the bare-metal / host environment. This amount of memory is larger than what is typically necessary but kept the math easy to follow.

Every attempt was made to run this environment in as close to the way users and customers would be expected to operate. For this reason, all the default services enabled by Red Hat Enterprise Linux 6 were left running, unless explicitly noted in the paper.

After starting the test on the guest virtual machine, it was noted that the response times were very low. After careful investigation, it was discovered that 16 users per virtual processor was not the appropriate load level for the virtualized environments. After considerable comparison testing and analysis, it was found that while bare-metal only required 16 users per hardware thread on an average to get the best throughput and response results. Virtual machine needed to be driven to 50 users per virtual processor to get the best throughout. And at that load level, the response time were well below the QoS targets of 2 seconds or less. So, for the purposes of show-the-best throughput scalability, the guest virtual machines tested in this paper used a load of 50 users per virtual processor.

Optimizations

To achieve the highest levels of performance and scalability of the guest virtual machines, several aspects of the entire software stack were investigated and optimized. The areas that were analyzed are the Red Hat Enterprise Linux operating system, the device drivers for the PureFlex System platform and the LAMP stack components. The operating system can be further divided into the traditional base OS components/subsystems and the KVM-virtualization environments.

The process of determining all the optimizations was in part drawn from the past experience with Red Hat Enterprise Linux and from the analysis of data captured while running DVD Store. Rather than provide a

sequential timeline of the issues discovered, all the steps tried to isolate the root cause of each issue that would be lengthy and cumbersome to read, and only the resulting set of optimizations will be listed and discussed.

Operating system optimizations

The DVD Store application is highly transactional and has a large dependency on the ability to obtain, modify, and update information stored in the MySQL database. The system load that the database places on the operating system and storage subsystem is significant. The capacity to support high levels of I/O quickly and efficiently is reflected by the maximum numbers of users that the system can sustain while maintaining QoS / response time targets.

I/O performance

Two initial concepts were focused on to improve upon the performance of the storage system: the Linux I/O scheduler and write barrier enablement. The I/O scheduler is a feature of the Linux kernel which determines the order in which I/O requests are submitted to the hardware. The default scheduler is called `CFQ`, but there are additional schedulers such as `deadline` and `anticipatory`. The different schedulers have algorithms that are optimized for different workload characteristics, such as favoring read performance over write performance or one application's I/O over another. Even though `CFQ` is the default, `deadline` is known to perform significantly better under many conditions, and therefore, it was set as the active scheduler. Write barriers are another Linux kernel feature that serves to ensure that I/O requests are submitted in the proper order and are committed to persistent storage before handling future requests*. However, when using enterprise class storage, such as the Storwize V7000 Unified used in this testing, write barriers are not necessary due to redundancy features of the hardware such as battery backed, mirrored caches. As the use of the Storwize V7000 Unified storage means that write barriers are not needed, they were disabled in addition to the changing of the I/O scheduler to deadline.

The improvements were immediately visible in the metrics reported by the DVD Store driver program and also the I/O data collected during the workload run looked noticeably better. For this initial experimentation, the changes to the system were made manually, but for real world use, a proper production consumable procedure is required. For this purpose, Red Hat Enterprise Linux provides a tuning infrastructure with profiles that can easily be deployed on a system. There is even a predefined profile that is a perfect match for the optimizations that need to be made to improve storage I/O performance. To install the tuning infrastructure, the following command was used:

```
[root@host ~]# yum install tuned
```

The next step is to activate the desired profile which is called `enterprise-storage` (Note: if no additional steps are performed the profile called `default` is used, which does alter system behavior). Administration of the tuning infrastructure is done used the `tuned-adm` utility. To activate the `enterprise-storage` profile, run the following command:

```
[root@host ~]# tuned-adm profile enterprise-storage
```

---

*Refer to the URL: http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/writebarr.html for additional information on write barriers.

```
Switching to profile 'enterprise-storage'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf:                                [  OK  ]
Calling '/etc/ktune.d/tunedadm.sh start':                  [  OK  ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 dm-1 dm-2 dm-3 dm-4 sda sd[  OK  ]d sde
Starting tuned:                                            [  OK  ]
```

The current configuration of the tuning infrastructure and a list of available profiles can be obtained by running the following commands:

```
[root@host ~]# tuned-adm active
Current active profile: enterprise-storage
Service tuned: enabled, running
Service ktune: enabled, running


[root@host ~]# tuned-adm list
Available profiles:
- server-powersave
- enterprise-storage
- spindown-disk
- latency-performance
- default
- laptop-ac-powersave
- laptop-battery-powersave
- throughput-performance
- desktop-powersave
Current active profile: enterprise-storage
```

After activating the `enterprise-storage` profile using `tuned-adm`, the testing was restarted to measure the performance gains compared to the initial result (which is the sum of the *orders per minute* metric from each guest virtual machine). With this optimization, the result was improved from the initial runs by 20% to 25%.

While the increase made by this optimization is impressive, the improvements to the load times of the DVD Store database are similarly compelling. Unlike the measurement phase of the workload which runs for a fixed amount of time, the time spent loading the database is determined purely by the performance of the system. The faster it loads, the lesser time it takes. For the purpose of this optimization effort, the faster the database loads, the more DVD Store testing could be done which directly translates to greater productivity. Using `tuned-adm` to activate the `enterprise-storage` profile resulted in an 83% decrease in the load time of the 160 GB database from 2.5 hours down to 22 minutes.

When `tuned-adm enterprise-storage` is used, the system is able to use the hardware resources more efficiently than in the baseline, driving higher levels of I/O throughput and processor utilization. These results might not be the typical benefits seen on every LAMP workload, but they do indicate workloads with similar behavior to the database load portion of this test using `tuned-adm enterprise-storage` that can make a significant impact.

It should be noted that the `tuned-adm enterprise-storage` profile has more benefits than just configuring the deadline I/O scheduler and disabling write barriers. The profile contains additional settings that change several behaviors of the Linux kernel running in the hypervisor, such as power management, processor scheduling, and memory management.

swappiness

Another option that prior experience has shown can be beneficial is the Linux kernel *swappiness* setting. This value dictates a general behavior regarding how aggressively the operating system can attempt to reclaim memory from the page cache. The setting is available in the /proc file system allowing users the ability to tailor behavior to their needs. To tune, simply echo a value from 0 to 100 in to `/proc/sys/vm/swappiness`. The higher the number the more the system will swap. The lower the value the less it will swap. For systems where performance is important, setting this value to 0 will minimize the possibility to swap.

Note: Setting to 0 will not disable swapping, but it just reduces the proactive tendency to do so. If the system falls under memory pressure, swapping will occur. Avoiding the default behavior to proactively swap eliminates one potential source of unnecessary disk I/O that can leech performance from your system.

Increase QLogic FC driver queue depth

For the QLogic 8Gb FC host bus adapter (HBA) available in the PureFlex System compute nodes, the Linux device driver (qla2xxx) defaults to using an I/O request queue depth of 32. This setting allows the device driver to hold up to 32 requests waiting to be sent to the disk storage attached to the HBA. When a high-performance storage subsystem, such as the Storwize V7000 Unified storage server is attached to a PureFlex System, allowing greater queue depths can be beneficial. With workloads that have a high rate of I/O requests, such as the DVD Store database tested in this paper, changing the queue depth to a value of 64 has been shown to improve the I/O rate by approximately 25% (refer to Figure 2).

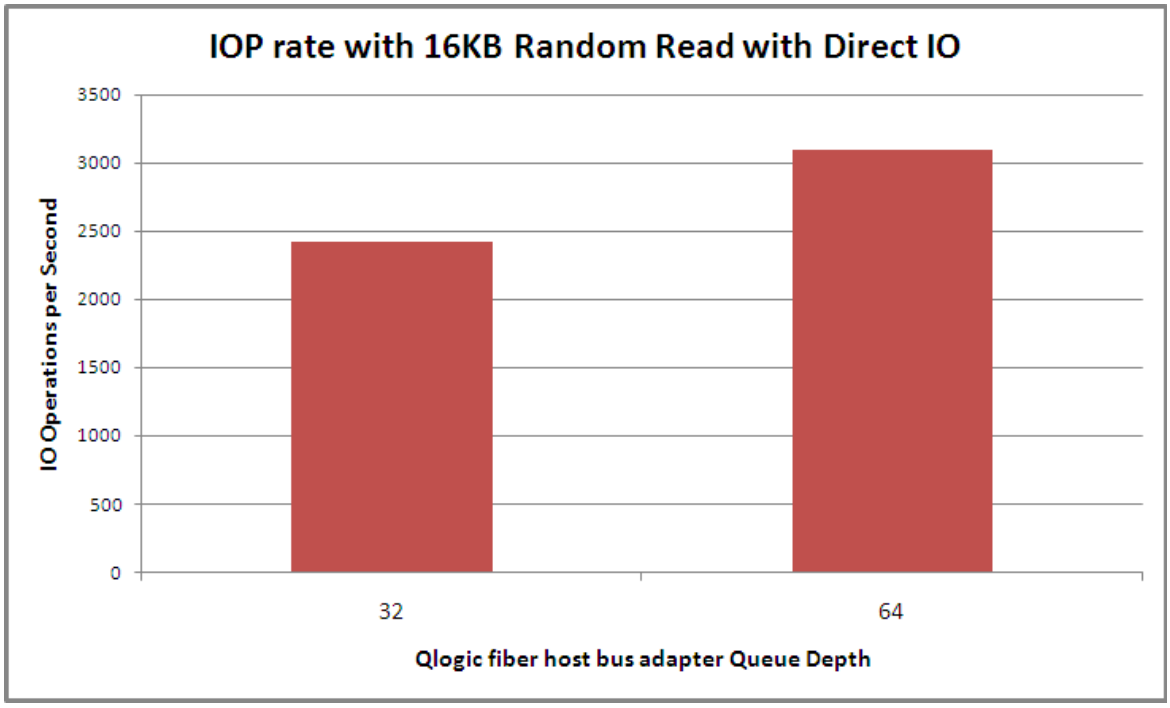

Figure 2: Comparing queue depth with IOP rate

To override the default queue depth value, the optional parameter `ql2xmaxqdepth=` must be specified at the time the driver is loaded. The PureFlex System used in this test was equipped with internal storage where Red Hat Enterprise Linux 6 was installed and booted. This makes it a simple matter to unload and reload the qla2xxx driver with the additional queue depth parameter with the following command.

```
[root@host ~]# modprobe qla2xxxx ql2xmaxqdepth=64
```

Kernel Samepage Merging

Another action that was taken was to disable Kernel Samepage Merging (KSM). Based on earlier experience, where processor utilization is a critical resource and in the absence of a memory over-commitment scenario, allowing KSM to run serves to introduce additional work in the form of breaking down transparent huge pages and consuming processor cycles when scanning memory. As there is no need for memory over-commitment for the purposes of this paper, the service which runs KSM was disabled, which can be done by running the following commands:

```
[root@host ~]# chkconfig ksm off
[root@host ~]# service ksm stop
Stopping ksm:                                              [  OK  ]

[root@host ~]# chkconfig ksmtuned off
[root@host ~]# service ksmtuned stop
Stopping ksmtuned:                                         [  OK  ]
```

LAMP stack optimizations

MySQL

This section explains the optimizations done to the MySQL LAMP stack component, which enabled the DVD Store application to perform with greater throughput and improve response times. Wherever applicable, the differences between bare-metal and guest virtual machines are noted.

Configuration file

The MySQL database server supports a large assortment of (over 250) variables to set and control configuration and runtime behavior. In this paper, following the DVD Store install instructions the default MySQL configuration was switched to the configuration for huge systems. To install this configuration file, run the following command.

```
[root@host ~]# cp /usr/share/doc/mysql-server-5.1.12/my-huge.cnf
/etc/my.cnf
```

The next time you restart `mysql`, the contents of this file will be used. The huge configuration file has a number of variables set to larger default value that are considered more applicable to systems with more than 1 GB of memory, which will likely apply to most servers.

innodb_buffer_pool_size

The next recommendation for MySQL is to increase the value of the `innodb_buffer_pool_size` variable. The comments in the configuration file indicate setting a value up to 80% of the available memory is fine, but provide warning about going above this. In this paper, 80% was used for bare-metal and every guest virtual machine size (5 GB to 40 GB) except the smallest which with only 2.5 GB was set to 60% or 1536 MB, leaving 1 GB for rest of the software stack.

The InnoDB buffer pool essentially becomes a cache, storing recently used data. This helps substantially when there is new or modified data that needs to be committed to the database. Storing the writes in memory for a period of time provide benefits. It allows more opportunity for a larger number of I/O requests to be merged when the data is submitted to storage, potentially reducing the number of I/O operations to disk. This change also enables frequently updated tables and fields to be written less often resulting is less continuous load on the storage system.

Similar improvements can be realized for guest virtual machines using a similar size to innodb buffer pool size ratio.

An unexpected side effect occurred when this value was changed. After this change, a gradual degrade in reported throughput from the driver program was observed. This continued for roughly 6 to 7 minutes into the run, at which point the throughput could be seen slowly returning. However, as the length of the runs was only 10 minutes and the throughput metric reported is a running average, there was insufficient runtime for the throughput to recover. Further analysis indicated that significant time was being spent in a background process used to defragment and coalesce free memory as part of transparent huge pages support. When this memory defragmenter was disabled, the drop in performance was no longer seen. As this impacted the reported throughput for the majority of the test duration, it was disabled using the following command:

```
[root@host ~]# echo “never” >
/sys/kernel/mm/redhat_transparent_hugepage/defrag
```

This was originally found when testing the bare-metal configuration that used a huge 40 GB of memory for the buffer pool, but was also seen when testing guest virtual machines configured with 5 GB or more memory.

table_open_cache

`table_open_cache` is another variable that can improve performance. The table open cache sets the number of table definitions that can be stored in the definition cache. If you use a large number of tables, you can create a large table definition cache to speed up opening of tables. The table definition cache takes less space and does not use file descriptors, unlike the normal table cache.[2] In this paper it was observed using the `mysql` command and the information seen with `show global status` that with DVD Store application at high (user) load, the default value of 512 was too small causing MySQL to have to close and later reopen tables over and over. This value was increased to 1536 until the number of times required to open the table reached a steady state.

thread_concurrency

Better concurrency (more parallelism) leads to better exploitation of the multicore Intel processors available in the PureFlex System compute node. MySQL uses the `innodb_thread_concurrency` variable to regulate concurrency. Using `innodb_thread_concurency,` the InnoDB engine tries to keep the number of operating system threads concurrently inside `InnoDB` less than or equal to the limit given by this variable. After the number of threads reaches this limit, additional threads are placed into a wait state within a first-in first-out (FIFO) queue for running. Threads waiting for locks are not counted in the number of concurrently running threads. [3]

The correct value for this variable is dependent on environment and workload. You will need to try a range of different values to determine what value works for your applications. A recommended value is two times the number of processors plus the number of disks.[3]

The range of this variable is 0 to 1000. A value of 20 or higher is interpreted as infinite concurrency before MySQL 5.1.12. From 5.1.12 onwards, you can disable thread concurrency check by setting the value to 0. Disabling thread concurrency check enables InnoDB to create as many threads as it needs.[3]

The default value is 8. For the testing in this paper, this value was effectively changed to 20 to minimize the thread wait states and permit the greatest level of concurrency.

query_cache_size

This variable defines the amount of memory allocated for caching query results. The default value is 0, which disables the query cache. The permissible values are multiples of 1024; other values are rounded down to the nearest multiple.[4]

---

[2] http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html#sysvar_table_cache
[3] Refer to the URL: http://dev.mysql.com/doc/refman/5.1/en/innodb-parameters.html#sysvar_innodb_thread_concurrency
[4] Refer to the URL: http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html#sysvar_query_cache_size

The huge configuration file sets this variable at 32 MB of memory. For the DVD Store workload, it was found that disabling the cache produced noticeably better performance. However, it must be noted that DVD Store is a very simple workload with few query types. A more robust and complex workload should benefit from using the cache.

max_connections

This variable defines the maximum permitted number of simultaneous client connections. By default, this is 151. For high-performance servers, such as the PureFlex System compute node, with one or two processors with a large number of processor cores, the default value will likely be too small. In this paper, the bare-metal configuration (with 16 hardware threads) used a value of 1024 to permit support of 800 simultaneous users. For the guest virtual machines, the size was scaled with the number of virtual processors assigned to a guest virtual machine.

Note: The values for the Apache HTTP server must be adjusted equally to allow an increased number of connections to reach the database.

innodb_log_file_size

This variable defines the size in bytes of each log file in a log group. The combined size of log files must be less than 4 GB. The default value is 5 MB. Sensible values range from 1 MB to 1/*N*-th of the size of the buffer pool, where *N* is the number of log files in the group. The larger the value, the lesser checkpoint flush activity is needed in the buffer pool, saving disk I/O. But larger log files also mean that recovery is slower in case of a crash.[5]

This value defaults to 100 MB. From information collected using `mysql` and the `show global status`, it was seen that even with a larger `innodb_buffer_pool_size` value, that data was being flushed to the disk before the buffer pool became full. In order to eliminate or at least reduce this behavior, increasing the value of `innodb_log_file_size` reduced the frequency of flushes, thereby reducing some of the load on the storage subsystem and this improved the DVD Store results. In this testing, the value was set to 1/40th of the `innodb_log_file_size` and this significantly reduced the frequency of log file flushes.

Apache HTTP Server

This section contains the optimizations done to the Apache LAMP stack component which enabled the DVD Store application to perform with greater throughput and improve response times. Wherever applicable, the differences between bare-metal and guest virtual machines are noted.

Prefork or worker multi-processing module (MPM)

Apache offer two MPM modules to choose from, the prefork and the worker.

The prefork multi-processing module (MPM) implements a nonthreaded, preforking web server that handles requests in a manner similar to Apache 1.3. It is appropriate for sites that need to avoid

[5] http://dev.mysql.com/doc/refman/5.1/en/innodb-parameters.html#sysvar_innodb_log_file_size

threading for compatibility with nonthread-safe libraries. It is also the best MPM for isolating each request, so that a problem with a single request will not affect any other.[6]

Most important is that `MaxClients` be big enough to handle as many simultaneous requests as you expect to receive, but small enough to assure that there is enough physical RAM for all processes.[5]

The worker MPM implements a hybrid multiprocess multithreaded server. By using threads to serve requests, it is able to serve a large number of requests with less system resources than a process-based server. Yet it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.[7]

The prefork module is the default. It uses processes to handle new requests. The worker module uses threads, which is more efficient using system resources (that is, memory) and is considered to be more scalable. However, threads running within the same process have some security implications. For this reason, the prefork module was used in this paper.

MaxClients

This parameter defines the maximum number of connections that can be processed simultaneously. In order to use the concurrency changes made to MySQL, it is necessary to adjust the Apache configuration accordingly. In the testing for this paper, the bare-metal configuration used a value of 1024. For the guest virtual machines, this value was scaled to the number of virtual processors assigned. The value was set according to the formula of 64 x number of virtual processors. When this value was below the default of 256, the default value was used.

ServerLimit

This directive sets the maximum configured value for `MaxClients` for the lifetime of the Apache process.[8]

Special care must be taken when using this directive. If `ServerLimit` is set to a value much higher than necessary, extra, unused shared memory will be allocated. If both `ServerLimit` and `MaxClients` are set to values higher than the system can handle, Apache may not start or the system may become unstable.[9]

With the `prefork` MPM, use this directive only if you need to set `MaxClients` higher than 256 (default). Do not set the value of this directive any higher than what you might want to set for `MaxClients`.[8]

In the paper, this value was set equal to `MaxClients`.

StartServers

This value sets the number of server processes created when Apache starts. Although this value is dynamically adjusted with load, in this paper, `StartServers` was set to the same value as

---

[6] http://httpd.apache.org/docs/2.0/mod/prefork.html
[7] http://httpd.apache.org/docs/2.0/mod/worker.html
[8] http://httpd.apache.org/docs/2.0/mod/mpm_common.html#serverlimit
[9] http://httpd.apache.org/docs/2.0/mod/mpm_common.html#serverlimit

`ServerLimit`, and so Apache was not required to perform any runtime allocations when a burst of user traffic occurs.

MinSpareServers, MaxSpareServers

These two directives determine the upper and lower threshold for the spare processes held in reserve to hold sudden changes in server load. Following the same line of thought for `StartServers`, `MinSpareServers` was set to the same value as `MaxSpareServers` to avoid runtime allocations. The value that was used for both was 256.

PHP accelerators

The only optimization that was done for PHP was to install PHP accelerators. A PHP accelerator is a PHP extension designed to improve the performance of applications written in the PHP programming language.[10]

Most PHP accelerators work by caching the compiled byte code of PHP scripts to avoid consuming additional processor resources when parsing and compiling source code on each request (some or even most of which may never be ran). To further improve performance, the cached code is stored in the shared memory and directly ran from there, minimizing the amount of slow disk reads and memory copying at runtime.[1]

A popular PHP accelerator APC v3.1.3p1 is prebuilt and distributed in Red Hat Enterprise Linux 6. To install APC v3.1.3p1, get the following package:

```
[root@host ~]# yum install php-pecl-apc
```

As it turns out, a PHP accelerator was performance neutral for the DVD Store application. However, there are sufficient studies and other examples showing that significant improvement can be realized. The use of a PHP accelerator is strongly encouraged when PHP scripts are used.

Test results

There are many elements that can affect workload performance and scalability. The system hardware (internal and external), the operating system, installed software components, and applications play a role in determining the overall behavior and capabilities any workload can achieve.

The DVD Store workload has a large disk I/O component stemming from high-transaction rates generated by the thin and uncomplicated application stack. With only four basic transaction types contained in the supplied PHP scripts in the application, simulated user requests from the driver tier waste little time pouring through application tier (web server) before being funneled down to the database tier. The resulting flood of queries and updates to the database generates enormous demands on the storage subsystem. Without an advanced, high-performance storage architecture, such as that offered in the Storwize V7000 Unified storage with the Easy Tier feature, and with the enormous compute potential of the latest addition of the Intel processor family, it might not have been possible to show the level of

[10] See http://en.wikipedia.org/wiki/PHP_accelerator

performance that this PureFlex System running Red Hat Enterprise Linux 6 with KVM virtualization is capable of and how well it can run and scale virtual machines with the LAMP stack.

Figure 3 compares the differences in the DVD Store throughput performance (order per minute) when a single guest virtual machine has additional processors and memory added. The guest virtual machine continues to scale up with additional resources even in a large eight virtual processor configuration, demonstrating the advanced and efficient virtualization capabilities available in KVM in Red Hat Enterprise Linux 6.

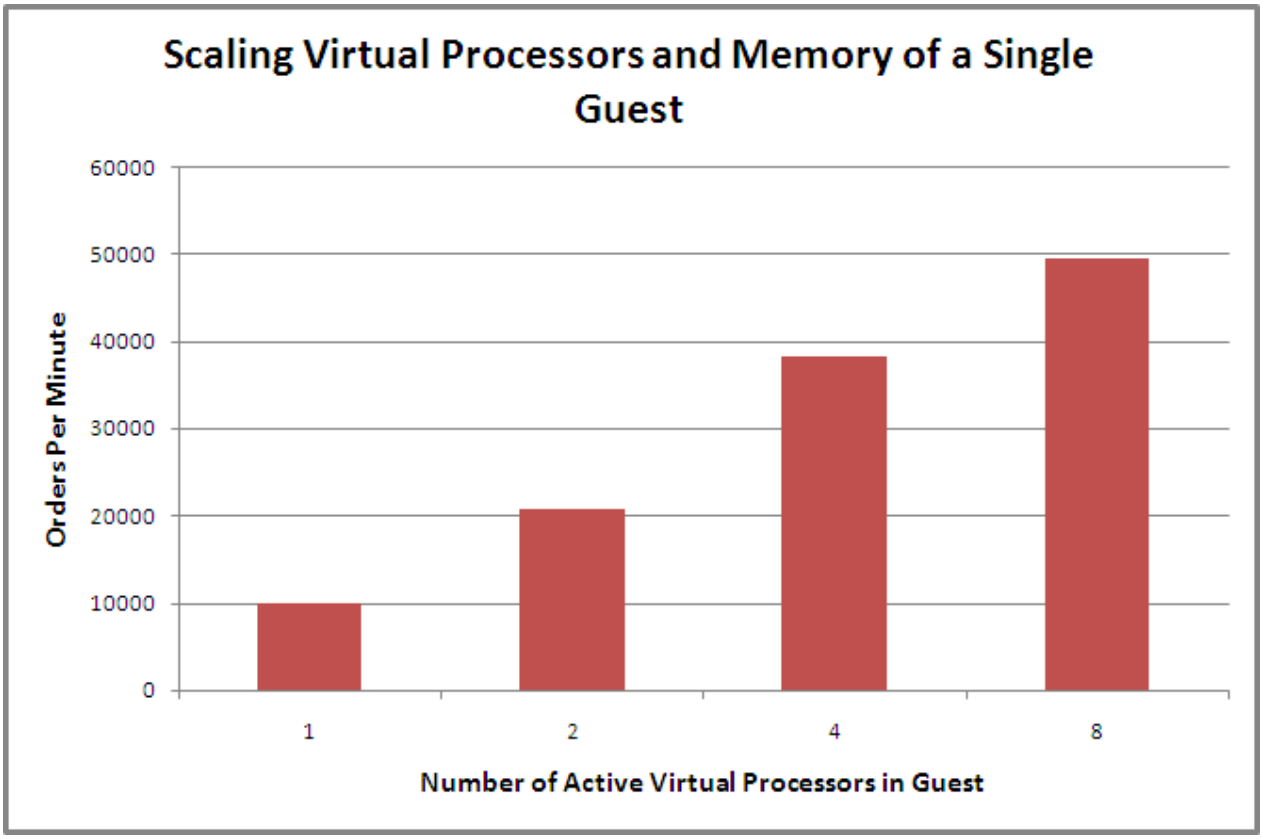

Figure 3: Scaling virtual processors and memory in one virtual machine

Here are the results showing both scale-out and scale-up capabilities of LAMP stack and the DVD Store application. Figure 4 compares the performance (in throughput reported by DVD Store) of the bare-metal configuration using all 8 cores (16 hardware threads) versus guest virtual machine configurations which also use all processor cores (or hardware threads) for virtual processors. Virtual machine configurations were tested that consist of two 8-virtual processor guests, four 4-virtual processor guests, Eight 2-virtual processor guests, and sixteen 1-virtual processor guests. The results clearly show that the virtualized configurations in all but the 16 guest virtual machine configuration achieved higher throughput than the bare-metal configuration. The eight 2-virtual processor configuration produced the best result providing better hardware utilization and delivering more than double the workload throughput than bare-metal. It also illustrates the best overall tradeoff of LAMP stack scaling limits when using higher processor counts and the impact of virtualization when using larger virtual machine counts seen in the 16 guest result.

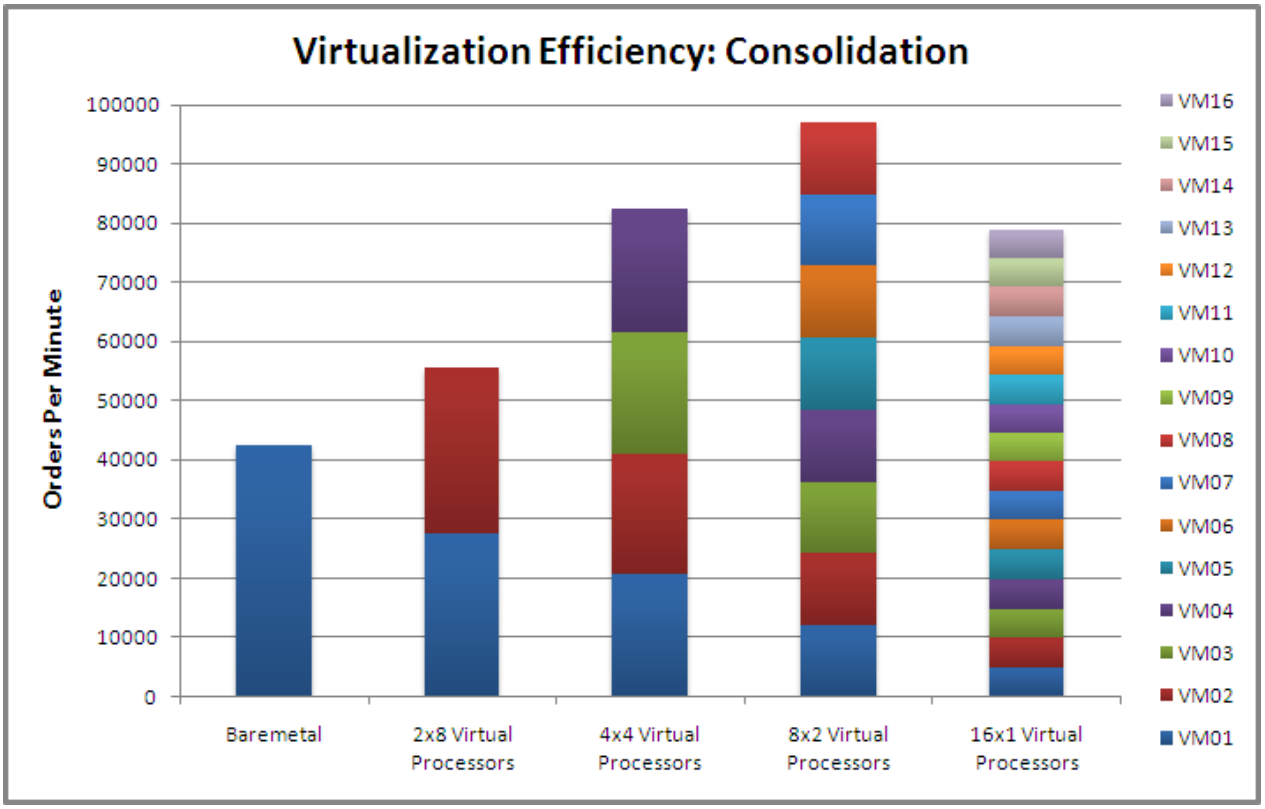

Figure 4: Aggregate throughput of various LAMP configurations using all the available processors

Another noteworthy detail although not shown in any figures was the response times reported by the DVD Store driver and collected during the test runs. Driving a user count of 50 users per virtual processor, it was observed that average response times never even came close to the QoS level of 2 seconds or less defined at the beginning of this paper. In only a few sample periods in as many guest virtual machines did the average response time ever even reach 1 second.

Summary

The DVD Store workload performance, both throughput and QoS, is a result of several factors.

The PureFlex System has the capacity to perform enormous amounts of work. This capacity is derived from the convergence of high performance components available in the PureFlex System architecture. The PureFlex System compute node utilizes the latest Intel processor technology providing the highest levels of compute resources, memory bandwidth and I/O capabilities. Coupled with integrated high-speed network and Fibre Channel switch technologies in the PureFlex System chassis, the PureFlex System expandable architecture is designed to support the most demanding workloads.

The DVD Store database with its extremely random-access pattern and burst of high transaction rates creates a heavy load on the storage subsystem. The demands are considerable and when combined with the need for low latencies become difficult and challenging for smaller storage configurations to satisfy. These workload characteristics make the capabilities of the storage subsystem a key factor in determining the overall performance. The results achieved in this paper would not have been possible without an advanced high-performance storage architecture, such as that offered in the Storwize V7000 Unified storage server. Leveraging the Easy Tier feature, the frequently accessed portions of the application's data was transparently migrated to SSDs providing enhanced performance. Easy Tier enabled a storage subsystem configured with too few spinning drives to excel, delivering extremely high IOPS and outstanding latencies.

Red Hat Enterprise Linux 6.2 offers many new features and refinements that deliver added value and functionality. The KVM virtualization support provided in Red Hat Linux 6.2 has reached new levels of performance, scalability, efficiency, and manageability.

The workload throughput and QoS and the overall system capacities described in this paper for DVD Store application can be attributed to five main (hardware and software) elements. If any of these elements were lacking, the workload results would have been diminished. The five elements are: the high performance of the PureFlex System platform, the virtualization efficiency of the Intel Xeon processor E5 series, the advanced architecture and high performance of the Storwize V7000 Unified storage server, the enterprise ready / feature-rich Red Hat Enterprise Linux with the integrated, highly scalable and very efficient KVM virtualization capabilities.

Resources

The following websites provide useful references to supplement the information contained in this paper:

- IBM Systems on PartnerWorld
  ibm.com/partnerworld/systems

- IBM Redbooks
  ibm.com/redbooks

- Wikipedia: PHP Accelerators
  http://en.wikipedia.org/wiki/PHP_accelerator

- MySQL 5.1 Reference Manual
  http://dev.mysql.com/doc/refman/5.1/en

- Apache MPM Common Directives
  http://httpd.apache.org/docs/2.0/mod/mpm_common.html

- Scaling the LAMP Stack in a Red Hat Enterprise Linux Virtualization Environment
  http://www.redhat.com/resourcelibrary/reference-architectures/doc034r3-lamp-scaling

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is

presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.