TECHNICAL WHITEPAPER

THE NEXT-GENERATION STORAGE INTERFACE FOR THE RED HAT ENTERPRISE LINUX KERNEL VIRTUAL MACHINE: VIRTIO-SCSI

Paolo Bonzini
Red Hat Israel
pbonzini@redhat.com

Laura Novich
Red Hat Israel
lnovich@redhat.com

EXECUTIVE SUMMARY

This technical paper describes the features and benefits of a new storage interface for virtual machines. virtio-scsi is a virtual small computer system interface (SCSI) host bus adapter (HBA). The virtio-scsi HBA is the foundation of an alternative storage implementation for virtual machines, replacing virtio-blk and improving upon its capabilities with:

- the ability to connect to multiple storage devices
- a standard command set
- standard device naming to simplify migrations
- device pass-through

The new virtio-scsi HBA can be deployed on guest virtual machines running Red Hat® Enterprise Linux® 6.3, as well as other guests running the Linux kernel 3.4 and above. virtio-scsi can be used for both boot and data disks.

LEGAL NOTICE

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701

TABLE OF CONTENTS

VIRTIO-SCSI'S FOREBEAR: VIRTIO-BLK

The virtio-scsi HBA supersedes virtio-blk, a simple high-performance para-virtualized storage device. Inherent in virtio-blk's design, however, are the following limitations:

- virtio-blk has limited scope, which makes new command implementation complex. Each time a new command is developed, the virtio-blk driver has to be updated in every guest.
- virtio-blk maps PCI functions and storage devices 1:1, limiting scalability.
- virtio-blk is not a true SCSI device. This causes some applications to break when they are moved from physical to virtual machines.

VIRTIO-SCSI FEATURES AND BENEFITS

Virtio-scsi is a new para-virtualized SCSI controller device. It provides the same performance as virtio-blk, and adds the following immediate benefits:

- improved scalability—virtual machines can connect to more storage devices.
- a standard command set—virtio-scsi uses standard SCSI command sets, simplifying new feature addition.
- standard device naming—virtio-scsi disks use the same paths as a bare-metal system. This simplifies physical-to-virtual and virtual-to-virtual migration.
- SCSI device passthrough—virtio-scsi can present physical storage devices directly to guests.

Improved scalability

Virtual machines can only be supplied with hardware that is available on a physical machine. While the hardware (CPUs, interrupt controllers, timers, bus bridges, etc.) itself is virtualized, it adheres to the same characteristics and limitations as the same hardware on a physical machine. The Peripheral Component Interconnect (PCI) bus has a limitation of 32 slots, each of which can have up to 8 separate functions.

PCI SLOTS:

Note that PCI slots are also referred to as devices, and this document will use the term 'slots' to avoid confusion with the virtio HBA or storage devices.

Every function behaves as if it is a separate peripheral and is not affected by other functions operating within the same slot. Whenever devices are added (hot-plug) or removed (hot-unplug) from a running machine, all functions in the same slot have to be added or removed simultaneously. System administrators can configure 28 of these 32 slots per KVM guest. The remaining four slots are used (or reserved) for various pieces of virtual hardware, including the video card, and IDE and USB controllers.

virtio presents each disk or network card as a separate PCI function. Due to the limitations regarding hotplug, each PCI slot will usually hold a single virtio device, thus limiting the KVM guest to 28 virtio devices. The actual limit is usually lower, because a few of the slots have to be set aside for other virtual devices—one or more network interfaces, a balloon driver, and possibly a virtual serial channel.

This limitation is relevant mostly for storage devices, since virtual machines only need a small number of network interfaces and a single virtio-serial device. Currently, Red Hat Enterprise Linux supports combining virtio devices in a single, multi-function PCI device.

virtio-scsi solves this limitation by multiplexing numerous storage devices on a single controller. Each device on a virtio-scsi controller is represented as a logical unit, or LUN. The LUNs are grouped into targets. This limit is much larger—each device can have a maximum of 256 targets per controller and 16,384 logical units per target.

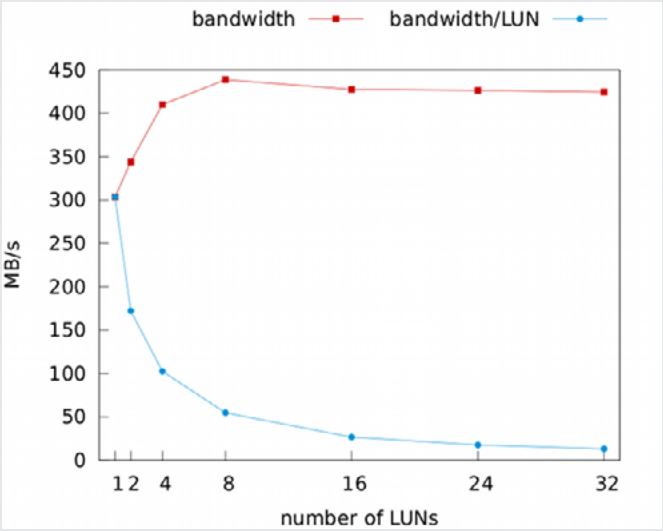

Figure 1. Maximizing LUN bandwidth

LUNs within a single target share resources (for example, using the same command queue(s) and interrupt(s)), so adding more units will always limit the bandwidth that is available to each LUN.

Figure 1, Maximizing LUN bandwidth, shows the result of benchmark testing using virtio-scsi with a varying number of LUNs (**Additional notes**, for a description of the benchmark configuration). The graph in Figure 1 shows that, even if new adapters cannot be added, subdividing I/O across multiple LUNs may increase the overall bandwidth available to the virtual machine. Note that this is only true up to a certain limit. Referring to Figure 1, for example, 8 LUNs were sufficient to maximize the available bandwidth.

The current implementation of virtio-scsi also requires that all LUNs within a given target share the same command queue. For this reason, spreading LUNs across multiple targets does not affect performance in Red Hat Enterprise Linux 6.3. Refer to **Additional notes**, for additional details.

The best way to resolve the trade-off depends on the applications that are running and the configuration of the virtual machine. The following setup tips should be considered:

- Disks with heavier I/O requirements can be put on separate controllers; these controllers can occupy a single multi-function PCI device, because hotplugging can be done at the SCSI level rather than at the PCI level.
- Additional disks can share a single controller, with each disk residing on a separate target.
- Add a serial number to each disk, so that the disks can be referenced with /dev/disk/by-id stable paths.

Standard command set

The virtio-blk device was designed to have a small and self-contained specification and to be easily implemented on different virtual machine monitors. To this end, the virtio-blk specification defines a small set of commands: “read”, “write”, and “flush cache”. Knowledge of these commands is shared by the virtual machine monitor (in the host) and the virtio-blk driver (in the guest). In order to add a new command to virtio-blk, it is necessary to update both the virtual machine monitor and the guest operating system.

virtio-scsi takes a different route. Its specification does not define commands for disks. Instead, it defines a transport protocol for these commands, and defers the actual definition of the commands to a set of industry-standard SCSI specifications: The most relevant are the SCSI Architecture Model (SAM), SCSI Primary Commands (SPC), and SCSI Block Commands (SBC) documents. This approach is shared with a number of other storage technologies, including but not limited to: Fibre Channel, the ATA Packet Interface (ATAPI), and USB mass storage devices.

Using a standardized SCSI command set keeps the virtio-scsi specification very stable. New features can be easily added to the virtualized storage stack, because the same code in the guest operating system will command both virtualized and physical disks. With virtio-scsi, when the guest operating system supports the functionality for physical disks, it will be able to detect the same capability in a virtio-scsi disk. After updating the virtual machine monitor, the guest will automatically detect the newly supported SCSI commands and use them.

Device naming

virtio-blk devices are represented by the guests with files whose names start with /dev/vd. This is unlike devices in a bare-metal system, whose names typically start with /dev/sd. Migration tools such as virt-p2v and virt-v2v know about this difference and can change the /etc/fstab and /boot/grub/device.map to accommodate it. In fact, /etc/fstab will usually refer to partitions or logical volumes by label or UUID so that this difference can be ignored.

Even so, references to /dev/sd* could be present in places not known to the tools mentioned above, thus creating problems in the migration of physical servers or with guests running under other virtualization technologies. As virtio-scsi disks use the same paths as a bare-metal system, this issue does not become a problem.

In addition to simple device names under /dev, both virtio-blk and virtio-scsi disks provide stable paths. These paths are independent of the sequence of hot-plug and hot-unplug operations leading to a particular disk being attached to the guest, and they are placed in subdirectories of /dev/disk. Stable paths are based on a serial number attributed to the device found within the virtual machine's configuration and are slightly different between virtio-blk and virtio-scsi.

SCSI device passthrough

For virtual disks that are backed by a whole LUN in the host, it can be desirable to let the guest send SCSI commands directly to the LUN (known as 'passthrough'). virtio-blk provides the basic ability to do so, with the following limitations:

- The guest kernel must use the virtio-blk protocol (rather than SCSI command passthrough) for I/O and configuration.
- Some programs will refuse to send SCSI commands to devices without the /dev/sd or /dev/sr prefix.
- It is not available on Windows systems.

The virtio-blk protocol includes a configuration mechanism, whereby the Linux kernel obtains information on the characteristics of the virtio-blk disk (the number and size of logical blocks in the disk, for example). This configuration mechanism is always active and the information it provides may be inconsistent with the information returned by SCSI passthrough.

virtio-scsi, on the other hand has the following characteristics:

- It natively accepts the SCSI command set.
- Disks are presented to the guest on a SCSI bus.
- It does not suffer from any of limitations mentioned above.

Table 1, Change disk type description to LUN, shows the simple change in the libvirt XML that enables passthrough. This procedure works for both virtio-blk and virtio-scsi, but the effect of this change is very different in each case. For virtio-blk, it will simply enable SCSI command passthrough and will not otherwise affect the configuration of the disk.

CHANGE DISK TYPE DESCRIPTION TO LUN

Original XML	Edited XML
<disk type='block' device='disk'>	<disk type='block' device='lun'>
<driver name='qemu' type='raw'/>	<driver name='qemu' type='raw'/>
<source dev='/dev/sda'/>	<source dev='/dev/sda'/>
<target dev='sda' bus='scsi'/>	<target dev='sda' bus='scsi'/>
<address type='drive'	<address type='drive'
controller='0' bus='0'	controller='0' bus='0'
target='3' unit='0'/>	target='3' unit='0'/>
</disk>	</disk>

When SCSI disks have their disk type set as lun, the guest operating system will receive the disk's configuration settings directly from the host's /dev/sda disk. This includes the disk's worldwide name (WWN) and serial number, so that the host and guest will see the same stable path. Capacity and logical block size are also the same as that visible on the host; hence the guest will be able to use disks with 4096-byte sectors, sometimes called advanced format disks.

At this time, libvirt only supports configuring block devices (disks and CD-ROM drives) on a SCSI bus and this limitation also applies to SCSI device passthrough. Red Hat Enterprise Linux may extend passthrough support in the future to other devices, including tapes.

ADVANCED FORMAT LIMITATIONS:

The same limitations on advanced format disks apply to guests and bare-metal systems. For example, Red Hat Enterprise Linux 6 only supports their usage for data disks, and not for boot disks.

SUMMARY

The virtio-scsi para-virtualized HBA, available as a technology preview in Red Hat Enterprise Linux 6.3, introduces a new virtual machine storage architecture.

Designed to replace virtio-blk, virtio-scsi retains virtio-blk's performance advantages while improving storage scalability, allowing access to multiple storage devices through a single controller, and enabling reuse of the guest operating system's SCSI stack.

Finally, virtio-scsi provides a technological base for the smooth introduction of new features in the future.

ADDITIONAL NOTES

Data for **Figure 1, Maximizing LUN bandwidth**, was obtained with the fio benchmarking tool, which is available from Fedora Extra Packages for Enterprise Linux (EPEL). The benchmark was run on an Intel Xeon CPU (model E5-2620 Sandy Bridge) running at 2 GHz, on a 2-socket, 12-core, 24-thread machine, with 32 GB of memory. The guest was configured with 4 virtual CPUs and 8 GB of RAM. The boot disk in this configuration was a virtio-blk disk; the virtio-scsi controller hosted 32 LUNs, all 256 MiB large and residing on a tmpfs file system on the host.

fio was configured as follows:

```
[global]
rw=read
bsrange=4k-64k
ioengine=libaio
direct=1
iodepth=4
loops=20
```

```
# Up to 32 jobs, one for each logical unit:
[lun0]
filename=/dev/disk/by-path/...-0:0:0:0
[lun1]
filename=/dev/disk/by-path/...-0:0:0:1
...
[lun31]
filename=/dev/disk/by-pathl/...-0:0:0:31
```


SALES AND INQUIRIES

NORTH AMERICA
1-888-REDHAT1
www.redhat.com

EUROPE, MIDDLE EAST AND AFRICA
00800 7334 2835
www.europe.redhat.com
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
www.apac.redhat.com
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
latammktg@redhat.com

www.redhat.com
#9847847_1012